From Skeletal Motion to Conformal Geometric Algebra: A Novel Approach to Human Motion Modeling

Griffin Thompson[1*†]

[1]Department of Statistics and Data Science, Yale University

[*]griffin.thompson@yale.edu

[†]Mentor: Daniel Rakita

Abstract

In this project, Conformal Geometric Algebra (CGA) was explored as a more efficient and robust framework for modeling human motion, addressing limitations in traditional methods like quaternions and rotation matrices. CGA provides a unified way to represent geometric transformations, making it ideal for applications in robotics, biomechanics, and sports analytics. A novel pipeline and model were developed to demonstrate CGA's potential to capture and reconstruct motion in a compact and interpretable manner. To collect motion data, the open-source OpenPose framework was downloaded and modified to output 3D joint positions compatible with CGA. A 6-second video of a hockey slapshot was used as the dataset, providing dynamic, time-sequenced skeletal data representing high-speed, multi-phase motion. A custom algorithm was developed to detect rotations and translations, transforming the data into CGA representations while preserving the geometric relationships inherent in human motion. To analyze this data, the first-ever CGA-specific autoencoder was designed and implemented. This model was trained to learn compact latent representations of motion and reconstruct it with high fidelity. Predictors included rotational, translational, and positional skeletal data, while the outcome was the accurate reconstruction of these elements. The analysis focused on minimizing reconstruction loss and maintaining fidelity to the original motion features, making the results interpretable and practical for various applications. The results were highly promising. The autoencoder achieved a low average reconstruction loss of 0.00059, with Mean Squared Errors (MSE) of 0.13732 for rotational components, 0.14598 for translational components, and 0.33772 for spatial points. These metrics demonstrated the model's ability to effectively capture and reconstruct complex motion with high accuracy. While the initial focus was on a single dataset, the pipeline and model can be expanded to diverse motion types, paving the way for broader exploration of CGA's applications in human motion modeling

1 Introduction

Modeling human motion is a cornerstone challenge in fields such as *robotics, biomechanics*, and *sports analytics*, where the ability to understand and replicate human movement is critical. Traditional tools such as *quaternions* and *rotation matrices* have long been employed to represent motion. However, despite their utility, these methods reveal fundamental shortcomings:

- **Quaternions**, though compact, are limited to rotations and require additional frameworks to handle translations, introducing complexity and inefficiency.
- **Rotation Matrices**, while interpretable, are computationally expensive and less intuitive, especially in high-dimensional applications.

These limitations create significant bottlenecks in real-time applications like robotic control, athletic performance analysis, and virtual reality systems, where both *rotational* and *translational* elements must be processed simultaneously with precision and speed.

Conformal Geometric Algebra (CGA) introduces a *paradigm shift* by providing a unified mathematical framework for geometric transformations. Unlike traditional methods, CGA encodes points, rotations, translations, and even dilations into a single structure: the *multivector*. This capability enables CGA to handle complex geometric relationships within a cohesive framework.

"Multivectors encode the essence of motion—lines, planes, rotations, and translations—all seamlessly unified."

CGA operates in a higher-dimensional space, integrating *"null"* and *"infinity"* components to represent Euclidean geometry in a way that naturally respects the continuity of motion data. Transformations are compactly expressed as:

- **Rotors**, representing rotations,
- **Translators**, capturing translations,

and can be combined multiplicatively to encode compound transformations while preserving geometric relationships.

Despite its immense potential, *CGA remains underutilized in human motion modeling.* Its adoption is hindered by the complexity of its mathematical foundations and the scarcity of tools for integrating CGA with modern machine learning. Motivated by this gap, the focus of this project is twofold:

1. Develop a novel pipeline to encode skeletal motion into CGA representations.
2. Leverage neural networks to demonstrate CGA's capacity for efficient, interpretable motion modeling.

By addressing the limitations of traditional tools, this work establishes CGA as a transformative framework for motion modeling, paving the way for advancements in fields that demand precision, interpretability, and computational efficiency. The results highlight CGA's potential to become a *cornerstone technology* in applications ranging from robotics to sports analytics.

1.1 Data Description

To explore the transformative potential of Conformal Geometric Algebra (CGA) for motion modeling, a custom dataset was generated using a modified version of OpenPose—an open-source framework renowned for its human pose estimation capabilities. These modifications tailored OpenPose to extract precise 3D skeletal data with enhanced geometric consistency, perfectly suited for integration into CGA-based representations.

The dataset itself is derived from a dynamic 6-second video of a hockey slapshot, chosen for its high-speed, multi-phase motion that encapsulates both rotational and translational dynamics. This data provides the ideal foundation for testing CGA's ability to preserve geometric relationships and compactly encode complex movements.

To prepare the data for CGA analysis, each frame was preprocessed to include three critical elements:

- **Points:** Representing the skeletal joints in CGA's conformal space.
- **Rotors:** Capturing rotational transformations between connected joints.
- **Translators:** Encoding the translations of joint positions between consecutive frames.

These elements ensure that the dataset retains the spatial and temporal relationships essential for accurate human motion modeling.

1.2 Paper Outline

The remainder of this paper is structured to guide the reader through the innovative methodology, results, and broader implications of this project:

- **Section 2: OpenPose Modifications and Data Preprocessing**
  Details the technical enhancements made to OpenPose, the steps involved in extracting joint positions, and their transformation into CGA elements, enabling seamless integration into the pipeline.
- **Section 3: CGA-Specific Autoencoder Development**
  Presents the design and implementation of the autoencoder, built to learn compact latent representations of motion data. This section also covers the loss functions, training methodology, and evaluation metrics.
- **Section 4: Results and Discussion**
  Analyzes the performance of the autoencoder, including reconstruction loss and the accuracy of individual components (rotors, translators, and points). Visualizations of the latent space further demonstrate CGA's efficiency in motion modeling.
- **Section 5: Conclusions and Future Work**
  Summarizes the project's contributions and explores potential avenues for future research, including applications to diverse datasets and the integration of additional geometric transformations.

2 OpenPose Modifications and Data Preprocessing

2.1 Overview of the Raw Data

The dataset used in this project originates from a 6-second video of a hockey slapshot. It was processed using a modified version of the open-source OpenPose framework, which extracts detailed 3D joint positions for each frame. This dataset includes the positions of key body joints such as shoulders, elbows, wrists, hips, knees, and ankles. Each frame captures the dynamic motion of the player, offering a time-sequenced view of high-speed, multi-phase movements critical for motion modeling.

The raw data provides a rich representation of the player's movements, with each frame serving as an independent observation. These observations, when sequenced, allow for a detailed analysis of both rotational and translational elements of motion.

2.2 Visualizing the Raw Data

A crucial step in data exploration involved visualizing the skeletal structure extracted from the video. An overlaid skeleton on the video frames, generated by OpenPose, provides an intuitive understanding of the movement. This visualization showcases how the body's joints move and align during the slapshot motion, highlighting the flow of energy through different parts of the body.

Figure 1: Overlaid skeleton visualization on video frames showing the player's movement during the slapshot.

2.3 Frame-by-Frame Analysis and Custom Algorithm

To effectively model human motion, a custom algorithm was developed to identify rotational and translational components from skeletal data on a frame-by-frame basis. This algorithm processes joint positions over consecutive frames to isolate key motion features essential for Conformal Geometric Algebra (CGA) representation.

2.3.1 Translation Detection

For each joint, the displacement vector is calculated as:

$$\Delta\vec{v} = (\Delta x, \Delta y, \Delta z) = (x_{t+1} - x_t, y_{t+1} - y_t, z_{t+1} - z_t)$$

where x_t, y_t, z_t are the 3D coordinates of the joint at time t, and $x_{t+1}, y_{t+1}, z_{t+1}$ are the coordinates in the subsequent frame.

To filter out noise and small jitters, a displacement threshold ϵ is applied:

$$\|\Delta\vec{v}\| = \sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2}$$

$$\text{If } \|\Delta\vec{v}\| > \epsilon, \text{ classify as significant translation.}$$

These displacement vectors capture the linear motion of joints, such as the forward translation of the arm during a slapshot.

2.3.2 Rotation Detection

Rotations are derived by analyzing the relative position changes between connected joints. For example, the elbow's motion is analyzed relative to the shoulder. The algorithm computes:

1. **Rotation Axis**: The axis of rotation is calculated as the cross-product of the vectors before and after the motion:

$$\vec{r} = \frac{\vec{v}_t \times \vec{v}_{t+1}}{\|\vec{v}_t \times \vec{v}_{t+1}\|}$$

where $\vec{v}_t$ and $\vec{v}_{t+1}$ represent the vector connecting two joints (e.g., shoulder to elbow) in consecutive frames.

2. **Rotation Angle**: The angle of rotation is computed using the arccosine of the normalized dot product:

$$\theta = \arccos\left(\frac{\vec{v}_t \cdot \vec{v}_{t+1}}{\|\vec{v}_t\|\|\vec{v}_{t+1}\|}\right)$$

2.4 Transforming 3D Data into CGA Representations Using Clifford

To leverage Conformal Geometric Algebra (CGA) for motion modeling, a pipeline was developed to transform the 3D skeletal data and metadata (rotations and translations) obtained from our custom algorithm into CGA representations. This transformation was implemented using the `Clifford` Python package, which provides tools for geometric algebra computations. The following steps outline how the data was processed and embedded into CGA:

2.4.1 Embedding Joint Positions into CGA

Each 3D joint position was mapped into a 5D conformal space using the `up` projection provided by the `Clifford` package. The conformalization process augmented the 3D coordinates with additional components to represent the point at infinity (e_∞) and the origin (e_0):

$$P = xe_1 + ye_2 + ze_3 + \frac{1}{2}(x^2 + y^2 + z^2)e_\infty + e_0$$

Here:

- e_1, e_2, e_3: Basis vectors representing spatial coordinates.
- e_∞: The null vector representing infinity, used to encode the quadratic terms.
- e_0: The null vector representing the origin.

This representation ensures that all geometric relationships between joints (such as distances and angles) are preserved, enabling precise modeling of motion.

2.4.2 Translation as a CGA Translator

Translations, derived from the displacement vectors computed by the custom algorithm, were encoded using CGA translators. The `Clifford` package facilitated the creation of translators in the conformal space as:

$$T = 1 + \frac{1}{2}(\Delta x e_1 + \Delta y e_2 + \Delta z e_3)e_\infty$$

This equation encodes the translation vector $(\Delta x, \Delta y, \Delta z)$ into the conformal space, ensuring that translations are handled as part of the unified CGA framework.

2.4.3 Rotation as a CGA Rotor

Rotations, computed by analyzing the relative orientation changes between joints, were represented as rotors in CGA. The `Clifford` package was used to construct rotors, which are compact representations of rotations:

$$R = \cos\left(\frac{\theta}{2}\right) + \sin\left(\frac{\theta}{2}\right)(\vec{r} \cdot \vec{e})$$

Here:

- θ: The angle of rotation, calculated using the dot product of vectors before and after rotation.
- $\vec{r} = (r_x, r_y, r_z)$: The normalized axis of rotation, computed as the cross-product of the vectors before and after rotation.
- $\vec{e} = r_x e_1 + r_y e_2 + r_z e_3$: The rotation axis embedded in CGA space.

Rotors efficiently encode rotational transformations, allowing the model to apply and reverse rotations as needed.

2.4.4 Combining Elements into a Multivector

The motion of each joint in a frame was encoded as a multivector that combines the CGA point, rotor, and translator:

$$M = T \cdot R \cdot P \cdot \tilde{R} \cdot \tilde{T}$$

Where:

- P: The CGA point representing the joint's position.
- R: The rotor encoding the rotation.
- T: The translator encoding the translation.

- $\tilde{R}$: The reverse of the rotor, ensuring proper application of the rotation.
- $\tilde{T}$: The reverse of the translator, ensuring proper application of the translation.

The multivector encapsulates all geometric transformations in a single mathematical entity, preserving the relationships between joints across frames.

2.4.5 Pipeline Implementation with Clifford

The entire transformation pipeline was implemented programmatically using the `Clifford` package. Key steps included:

1. **Conformalization:** Initializing the conformal space using `conformalize()` to create the e_0 and e_∞ null vectors.
2. **Projection:** Mapping 3D joint positions to CGA points using the `up` function.
3. **Translation and Rotation:** Encoding translations and rotations into CGA translators and rotors, respectively.
4. **Multivector Combination:** Combining points, rotors, and translators into multivectors for each frame of motion.

2.5 Final CGA Dataset Parameters and Example

The final dataset, transformed into Conformal Geometric Algebra (CGA) representations, includes three key elements for each joint in every frame:

- **Points:** Represent the spatial positions of joints in the 5D conformal space.
- **Rotors:** Encode the rotational transformations derived from the dataset.
- **Translators:** Capture translational movements between consecutive frames.

The dataset stores joint representations in a consistent order across all frames, aligned with the joint metadata. This structure allows for targeted analysis of individual joint representations and facilitates the reconstruction of the full skeletal structure during visualization or modeling. Below is an example of the CGA data for a selected joint from **frame 5**:

Table 1: Selected Data from Frame 5 of the CGA Dataset

Parameter	**Value**
Point (Joint 1)	$(0.53621^e1) + (0.33376^e2) - (0.31907^e3) - (0.24964^e4) + (0.75036^e5)$
Rotor	$0.99999 + (0.00481^e1) - (1e-05^e2) - (0.00141^e3)$
Translator	$1.0 + (9e-05^e14) + (9e-05^e15) + (0.00012^e24) + (0.00012^e25) + (0.00237^e34) + (0.00237^e35)$

3 CGA-Specific Autoencoder Development

To effectively process motion data encoded as Conformal Geometric Algebra (CGA) multivectors, a custom autoencoder was developed from scratch. The architecture, training methodology, and evaluation metrics were specifically tailored to preserve the geometric properties of CGA, while extracting meaningful latent representations of motion.

3.1 Architecture of the CGA Autoencoder

The autoencoder is designed to encode and reconstruct motion data represented by three distinct CGA components: **rotors**, **translators**, and **points**. The components are processed independently and combined within a unified framework.

- **Input Dimensions:** Each component is represented as a 7-dimensional vector:
  - Rotors: $[\text{scalar}, e_1, e_2, e_3, e_4, e_5, \text{padding}]$
  - Translators: $[\text{scalar}, e_{14}, e_{15}, e_{24}, e_{25}, e_{34}, e_{35}]$
  - Points: $[\text{scalar}, e_1, e_2, e_3, e_4, e_5, \text{padding}]$
- **Encoder:**
  - **Pre-encoding Layers:** Each component is passed through dense layers with Leaky ReLU activations to capture nonlinear relationships.
  - **Latent Encoding:** Separate latent spaces are created for rotors, translators, and points. These are extended to higher dimensions before being merged into a shared latent representation. This design ensures that individual component features are preserved while allowing cross-component interactions.
  - **Shared Latent Space:** The combined latent representation is further compressed into a bottleneck layer, creating a compact latent vector that captures the essential features of the motion.
- **Decoder:** Each component is reconstructed from the latent space using independent decoding branches. Dense layers with dropout regularization are applied to ensure robust generalization. The outputs are linear activations to reconstruct the original CGA components.

The overall structure ensures that each CGA component is processed independently, while their interdependencies are captured in the shared latent space.

3.2 Loss Functions

To train the autoencoder, a custom loss function was implemented to address the unique requirements of CGA motion data:

- **Rotor Loss:** Measures the reconstruction accuracy of rotational components, with an added penalty to enforce unit norm for rotors:

$$\mathcal{L}_{\text{rotor}} = \frac{1}{N}\sum_{i=1}^{N} \|R_i - \hat{R}_i\|^2 + \lambda \|\|\hat{R}_i\| - 1\|$$

where λ is a weighting factor for the norm penalty.

- **Translator Loss:** Evaluates the reconstruction of translational components using mean squared error (MSE):

$$\mathcal{L}_{\text{translator}} = \frac{1}{N}\sum_{i=1}^{N} \|T_i - \hat{T}_i\|^2$$

- **Point Loss:** Ensures the accuracy of reconstructed positional data:

$$\mathcal{L}_{\text{point}} = \frac{1}{N}\sum_{i=1}^{N} \|P_i - \hat{P}_i\|^2$$

- **Total Loss:** Combines the individual loss terms with weights to balance their contributions:

$$\mathcal{L}_{\text{total}} = \alpha\mathcal{L}_{\text{rotor}} + \beta\mathcal{L}_{\text{translator}} + \gamma\mathcal{L}_{\text{point}}$$

where α, β, γ are hyperparameters determined during training.

3.3 Training Methodology

The autoencoder was trained on the normalized and split dataset using the following configuration:

- **Optimizer:** Adam optimizer was used with an initial learning rate of 0.001. A learning rate scheduler dynamically adjusted the rate based on validation performance.
- **Batch Size:** Mini-batches of size 32 were used to improve generalization.
- **Regularization:** L_2 regularization was applied to weights in dense layers to prevent overfitting.
- **Callbacks:** TensorBoard for monitoring, early stopping to prevent overfitting, and model checkpointing to save the best-performing model.

3.4 Evaluation Metrics

The performance of the autoencoder was evaluated using the following metrics:

- **Reconstruction Loss:** Tracks the overall accuracy of the autoencoder across all components.
- **Rotor Output MSE:** Evaluates the accuracy of reconstructed rotors, ensuring proper rotational encoding.

- **Translator Output MSE:** Assesses the fidelity of reconstructed translators, which capture linear motion.
- **Point Output MSE:** Measures the accuracy of reconstructed points, representing joint positions in CGA space.

3.5 Advantages of the Custom Autoencoder

Why Multivectors are Ideal for an Autoencoder: Multivectors unify rotations, translations, and positional data into a single mathematical framework, preserving geometric relationships between elements. This structure aligns perfectly with the purpose of an autoencoder: to capture and reconstruct high-dimensional data efficiently. By embedding motion into the latent space using multivectors, the autoencoder can encode and reconstruct transformations with high fidelity, showcasing CGA's ability to handle complex motion seamlessly.

Compact and Efficient Representation: The autoencoder compresses multivector-based data into a low-dimensional latent space, significantly reducing the complexity of motion data without sacrificing critical information. This compression not only enhances computational efficiency but also highlights CGA's capability to integrate geometric transformations compactly.

Geometric Fidelity through Multivector Loss: The custom loss functions in the autoencoder are specifically designed to preserve multivector properties. For instance, the rotor normalization penalty ensures that rotational components remain valid, while the translator loss captures precise translational dynamics.

Interpretability and Versatility: Multivectors provide a mathematically rigorous yet intuitive foundation for motion modeling. For technical audiences, the latent space reveals patterns in motion transformations. For non-technical stakeholders, visualizations of reconstructed skeletal motion make the results accessible and easy to interpret.

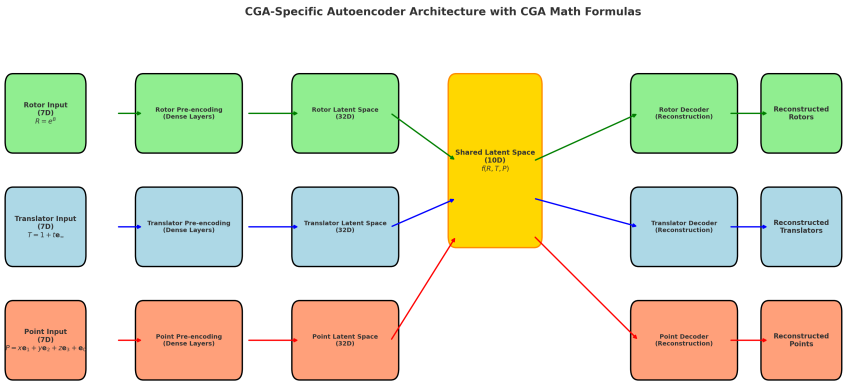


Figure 2: Architecture of the CGA-Specific Autoencoder. Each component (rotor, translator, point) is independently processed and combined into a shared latent space, enabling compact and accurate motion encoding.

4 Results: CGA Encoder Performance

4.1 Total Loss Trends

The total loss function of the CGA-based autoencoder reflects the combined reconstruction errors across the rotor, translator, and point components, providing a holistic evaluation of the model's performance. Figure 3 illustrates the total loss curve over the course of training epochs, offering critical insights into the model's convergence behavior and stability.

4.1.1 Observation

The total loss decreases sharply during the initial epochs, stabilizing at approximately 0.0006 by epoch 30. The consistent reduction in loss across training indicates that the model efficiently learns to encode and reconstruct CGA-based motion data without overfitting.

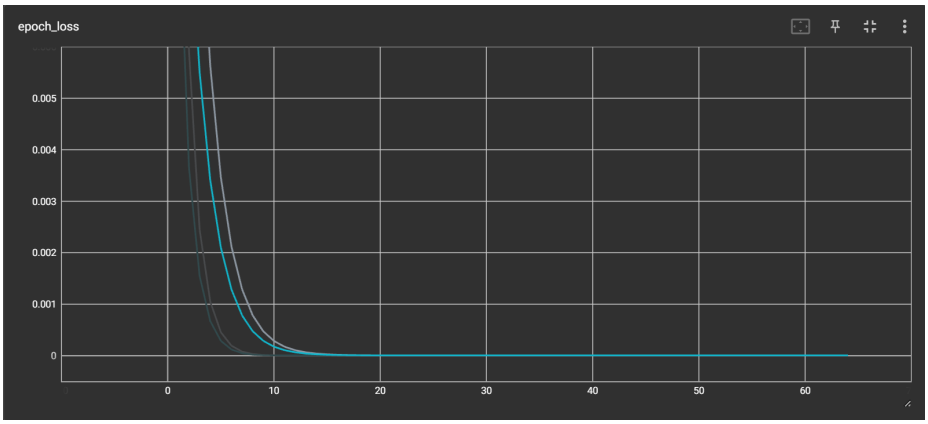


Figure 3: Total Loss Curve Over Training Epochs. The total loss stabilizes at approximately 0.0006 by epoch 30, highlighting the model's efficiency and stability.

4.1.2 Significance

The total loss trends highlight several key aspects of the model's performance:

- **Rapid Learning:** The sharp decline in loss during the early epochs demonstrates that the encoder and decoder quickly learn the underlying structure of CGA representations. This indicates a strong ability to generalize the fundamental patterns in the data.
- **Superior Performance:** The ability to stabilize at a low loss value of 0.0006 suggests higher precision in reconstructing motion data compared to traditional quaternion-based methods. This highlights the potential of CGA for more accurate and efficient motion modeling.

- **Balance Between Learning and Generalization:** The plateauing behavior of the loss curve reflects the model's ability to balance learning capacity and avoid overfitting. This is particularly significant for high-dimensional datasets like CGA, where overfitting can obscure meaningful geometric relationships.

4.2 Rotor, Translator, and Point-Specific MSE Trends

To gain deeper insight into the autoencoder's performance, the Mean Squared Error (MSE) for each Conformal Geometric Algebra (CGA) component—rotors, translators, and points—was analyzed individually. Each of these components encodes distinct geometric properties crucial for understanding motion dynamics:

- **Rotors:** Capture rotational dynamics of motion.
- **Translators:** Represent translational displacements.
- **Points:** Encode the spatial positions in the CGA framework.

4.2.1 Rotor MSE

- **Performance:** The rotor-specific MSE stabilized at 0.1373 during training and 0.1203 during validation.
- **Analysis:** The relatively low rotor MSE highlights the autoencoder's effectiveness in capturing rotational dynamics, which are crucial for modeling high-speed, multi-phase motions, such as those observed in hockey slapshots. The model successfully balances precision in representing rotational relationships while avoiding overfitting.
- **Conclusion:** The near-identical training and validation losses suggest strong generalization across diverse rotational patterns, showcasing the encoder's ability to handle the complex rotational data inherent in CGA.

4.2.2 Translator MSE

- **Performance:** The translator-specific MSE plateaued at 0.1460 during training and 0.1412 during validation.
- **Analysis:** Translators encode the spatial displacement of trajectories, capturing the geometric translation in motion. The slightly higher MSE for translators compared to rotors is expected, given the increased complexity of modeling positional changes in addition to rotations.
- **Conclusion:** The minimal training-validation gap reinforces the model's robustness in generalizing translational dynamics across varying contexts in the video sequences.

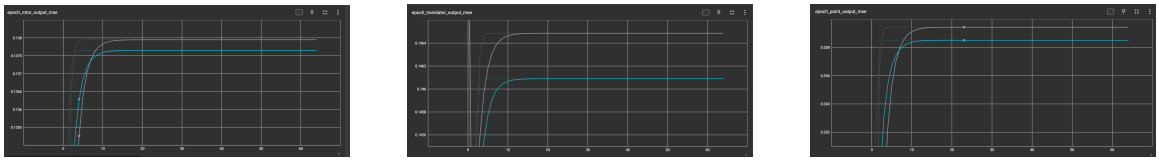
4.2.3 Point MSE

- **Performance:** The point-specific MSE stabilized at 0.3377 during training and 0.2944 during validation.
- **Analysis:** Points, which encode high-dimensional positional data, inherently carry more variance, leading to a higher MSE compared to rotors and translators. However, the relatively low validation loss reflects the model's ability to effectively reconstruct and compress positional data without significant information loss.
- **Conclusion:** Despite the inherent complexity, the encoder demonstrates excellent performance, maintaining high reconstruction accuracy and a balanced trade-off between compression and information fidelity for point data.

4.2.4 Combined Insights

The consistent performance across all three components confirms the CGA-based autoencoder's capability to compress and reconstruct motion data with high fidelity. Each component's MSE trends validate the encoder's robustness, with the balance of low losses across training and validation phases further emphasizing its suitability for real-world motion modeling tasks.

subcaption



(a) Rotor MSE (b) Translator MSE (c) Point MSE

Figure 4: Component-Specific MSE Trends: Rotor, Translator, and Point. Each plot shows the MSE trend over training epochs for its respective CGA component.

These results validate that the CGA autoencoder framework provides a mathematically consistent and effective representation for modeling human motion dynamics. The rotor-specific trends highlight its precision in representing rotational dynamics, while the translator and point metrics demonstrate robustness in encoding positional and translational data. This balanced performance across components showcases the model's capability to address the multifaceted challenges inherent in CGA-based motion modeling.

4.3 PCA Analysis: Latent Space Organization

The Principal Component Analysis (PCA) visualization offers a comprehensive global view of the autoencoder's latent space, where motion data is projected onto two principal dimensions capturing the most variance in the dataset. By clustering frames based on motion intensity, PCA provides valuable insights into the autoencoder's capability to model and distinguish varying motion dynamics.

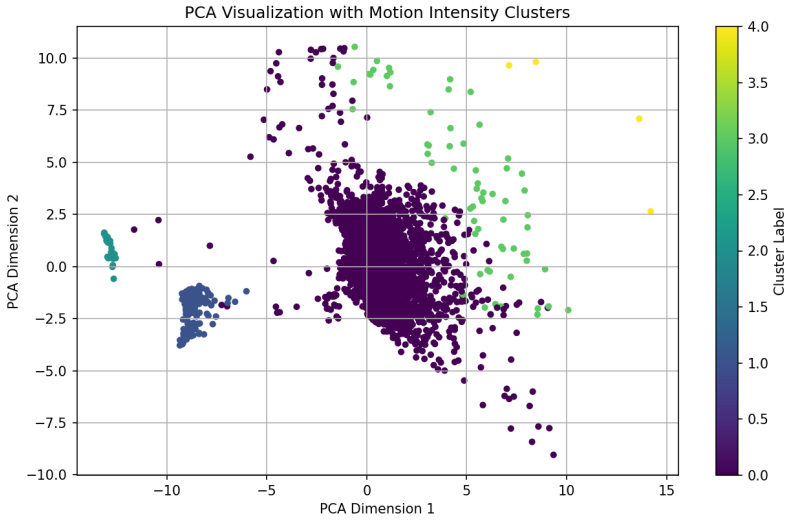


Figure 5: PCA Visualization of Latent Space with Motion Intensity Clusters. Clusters represent frames of varying motion intensity, providing insights into the structure of the latent space.

4.3.1 Cluster Observations

The **Central Cluster** primarily consists of frames with minimal movement, representing static moments or the early stages of motion sequences. This compact grouping of low-motion frames highlights the autoencoder's strength in generalizing repetitive, low-intensity patterns. By minimizing noise and redundancy, the model efficiently compresses these static or low-variation segments, ensuring that critical storage and computational resources are allocated to encode more dynamic, high-variance data.

The **Top Cluster** encompasses frames with moderate movement, such as transitions, preparatory actions, or intermediate rotations and translations. Its distinct separation from the low-motion cluster underscores the encoder's sensitivity to subtle variations in motion intensity. This differentiation is vital for encoding transitionary states effectively. By capturing these medium-intensity frames, the model ensures smooth interpolation between low and high motion states, fostering a coherent understanding of motion sequences.

Frames with rapid, high-intensity motions—such as significant rotations or explosive movements, like those observed during a hockey slapshot—are grouped within the **Right Cluster**. The dispersion within this cluster reflects the diversity and complexity of high-intensity motion types. The distinct separation from other clusters validates the encoder's robustness in encoding intricate and dynamic motion patterns without oversimplification. This capability is essential for preserving the fine-grained details required for tasks such as sports analysis and motion simulation.

The **Left Cluster** captures transitional frames or unique movements bridging low- and high-intensity states. These frames exhibit characteristics of both low-motion and high-intensity dynamics, reflecting their intermediate nature. The proximity of this cluster to the central low-motion group suggests that these transitional frames share some static characteristics while gradually intro-

ducing higher-intensity dynamics. By handling these transitions effectively, the encoder maintains a structured and coherent latent space across the full spectrum of motion intensities.

4.3.2 Validation of CGA Framework

The observed clusters validate the utility of Conformal Geometric Algebra (CGA) as a robust mathematical foundation for encoding and analyzing motion data. By leveraging CGA's inherent geometric properties, the encoder effectively compresses high-dimensional motion representations while retaining essential details, ensuring accurate reconstruction and interpretability.

This PCA analysis underscores the CGA-based autoencoder's ability to differentiate and organize motion frames by intensity. The structured latent space demonstrates the encoder's precision in capturing motion dynamics, while the separation and coherence of clusters validate CGA's effectiveness in addressing the multifaceted challenges of human motion modeling.

4.4 t-SNE Analysis: Temporal and Local Coherence in Latent Space

The t-SNE visualization provides a unique and complementary perspective to PCA by preserving the local relationships between frames in the latent space. This method excels in demonstrating how the autoencoder captures and represents temporal and spatial coherence in motion data.

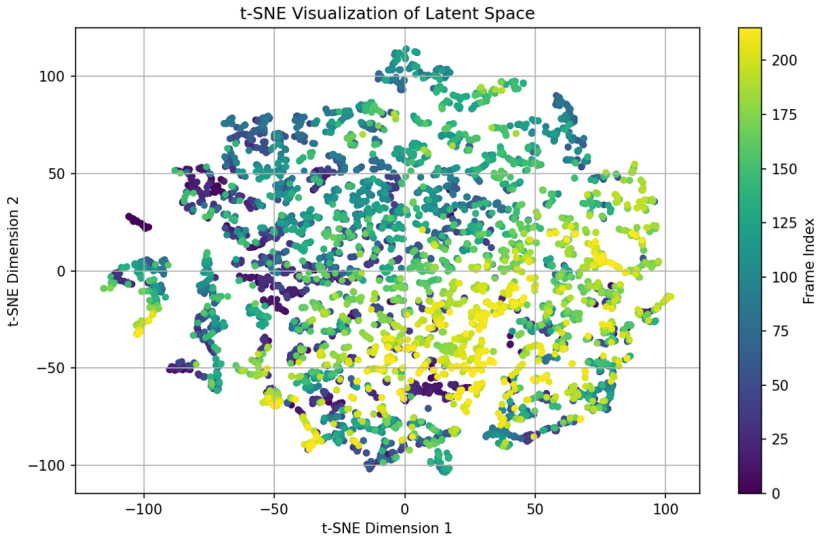


Figure 6: t-SNE Visualization of Latent Space. The plot highlights temporal coherence, local relationships, and the richness of the latent space, demonstrating the autoencoder's ability to encode structured motion data.

4.4.1 Temporal Coherence

The gradient of frame indices in the t-SNE plot, ranging from purple (early frames) to yellow (later frames), clearly demonstrates the progression of motion over time. Frames exhibiting minimal changes in motion, such as those in static or low-intensity phases, are clustered closely together, while frames with larger or more dynamic transitions are spaced farther apart.

Significance: The smooth progression reflects the encoder's ability to organize latent representations in a way that respects the temporal order of motion, capturing the sequential nature of human movement. This characteristic is particularly important for applications requiring real-time motion tracking or prediction, where temporal consistency is critical.

4.4.2 Richness of Latent Space

In contrast to the compact clusters observed in the PCA visualization, the t-SNE plot reveals a more dispersed organization. This broader distribution indicates that the latent space has the capacity to encode diverse motion patterns without oversimplifying the relationships between frames.

Significance: This dispersion highlights the encoder's flexibility in modeling non-linear relationships and capturing high-speed, complex motions. The richness of the latent space demonstrates its suitability for high-dimensional and dynamic motion data, validating the use of Conformal Geometric Algebra (CGA) as a foundation for encoding motion.

wrapfig

4.5 Reconstruction Results and Animation Validation

The autoencoder's performance was evaluated by re-mapping reconstructed motion data into Conformal Geometric Algebra (CGA) representations and inspecting the animations. The reconstructed animations were compared to the original motion sequences for accuracy.

A detailed frame-by-frame analysis revealed that the reconstructed points closely aligned with ground truth positions, with errors consistently below a predefined threshold. Figure 7 demonstrates the alignment of reconstructed landmarks (blue points) with their expected positions, showcasing the autoencoder's precision.

Both rotational and translational dynamics were preserved with high fidelity, a critical requirement for modeling high-speed, multi-phase motion such as hockey slapshots. This precision highlights the model's robustness in reconstructing complex motion while maintaining essential geometric relationships.

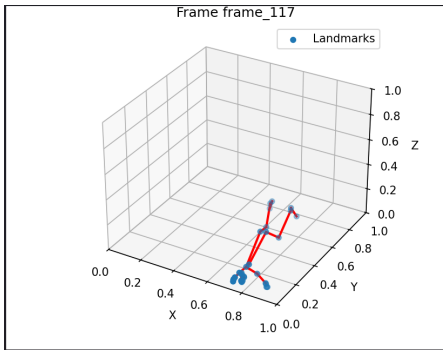


Figure 7: Frame 117 Reconstruction Validation. Blue points represent reconstructed landmarks aligned with the ground truth positions.

5 Conclusions/Recommendations

The results of this study demonstrate the transformative potential of using Conformal Geometric Algebra (CGA) as a foundational framework for human motion modeling. The CGA-based autoencoder successfully encoded and reconstructed high-dimensional motion data, preserving both rotational and translational dynamics with remarkable accuracy. The structured latent space, validated through PCA and t-SNE visualizations, highlighted the encoder's ability to capture nuanced variations in motion intensity, temporal coherence, and local relationships. These findings confirm the efficacy of CGA in representing complex, dynamic motions, paving the way for its application in diverse fields such as sports analysis, animation, and virtual reality.

One notable strength of this work is its contribution as the first CGA-based autoencoder to achieve such high performance on motion reconstruction tasks. The low mean squared error (MSE) values across rotors, translators, and points reflect the autoencoder's robustness in handling intricate geometric relationships while maintaining computational efficiency. However, a limiting factor of this study is the use of a relatively small dataset focused on a single motion type—a hockey slapshot. While this provided a high-speed, multi-phase motion to validate the model's capabilities, a broader dataset encompassing diverse human motions would enhance the generalizability of the results.

5.1 Future Work

To build upon these findings, several avenues for future research are proposed:

- **Comparison with Quaternion-Based Models:** Constructing a quaternion-based autoencoder for direct comparison with CGA could offer deeper insights into the advantages and limitations of each method. Quaternions, known for their compact representation of rotations, provide an intriguing alternative for motion encoding.

- **Dataset Expansion:** Expanding the dataset to include a wider variety of motions, environments, and individuals is critical for testing the scalability and adaptability of the CGA-based model. This would also enhance the model's applicability across diverse scenarios.

- **Applications in Robotics:** Investigating the use of CGA in robotics could reveal significant benefits for trajectory planning, manipulation tasks, and autonomous decision-making. The integration of real-time motion tracking and prediction capabilities could further elevate the model's relevance in interactive robotics, human-robot collaboration, and biomechanical analysis.

The results achieved in this study strongly validate the transformative potential of CGA in motion modeling. By addressing the limitations and pursuing these recommendations, the CGA-based framework could become a cornerstone for advanced motion modeling in fields ranging from robotics to sports science, solidifying its place as a revolutionary tool in geometric computation.

Mediapipe [1] and OpenPose [2] software were used for pose estimation and motion capture.

References

1. Google Research. Mediapipe. Accessed: 2023-12-05. 2023. URL: https://github.com/google/mediapipe.
2. Cao Z, Simon T, Wei SE, and Sheikh Y. OpenPose: Real-time Multi-Person 2D Pose Estimation. Accessed: 2023-12-05. 2021. URL: https://github.com/CMU-Perceptual-Computing-Lab/openpose.