SRH Hochschule Heidelberg

Masters Thesis

Skeleton-Based Hand Gesture Comparison

Author:
Amogh Shrinivas Goudar

Supervisor:
Prof. Dr.-Ing. Mehrdad Jalali
Dr. Binh Vu

A thesis submitted in fulfilment of the requirements
for the degree of MSc. Applied Data Science and Analytics

in the

School of Information, Media and Design

September 2025

Declaration of Authorship

I, Amogh Shrinivas Goudar, declare that this thesis titled, 'Skeleton-Based Hand Gesture Comparison' and the work presented in it is my own. I confirm that this work submitted for assessment is expressed in my own words and is my own. Any uses made within it of the works of other authors in any form (e.g., ideas, equations, figures, text, tables, programs) are appropriately acknowledged at any point of their use. A list of the references employed is included.

- This work was done wholly for a research degree at this University.
- It has been clearly stated in this work for any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution.
- Where I have consulted the published work of others, credit is always given to the concerned work.
- Where I have quoted from the work provided by others, the source is always given unless it is entirely my work.
- I have acknowledged all main sources that supported me and helped during this work.

Signed: _____

Date: _____

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisor, Prof. Dr.-Ing. Mehrdad Jalali, for his invaluable guidance, encouragement, and continuous support throughout the course of this thesis. His expertise, insightful feedback, and patience were instrumental in shaping my work and motivating me to push beyond my limits.

I would also like to thank Prof. Dr.-Ing. Binh Vu for his guidance and support as my second supervisor.

I am equally thankful to SRH University Heidelberg, the Applied Data Science and Analytics program, and all the professors for providing me with the resources, academic environment, and opportunities that made this research possible.

I would also like to acknowledge my colleagues and peers for their constructive discussions and support, which greatly enriched my research experience. Special thanks go to my family and friends for their constant encouragement, understanding, and unconditional support throughout this journey.

Finally, I am deeply grateful to everyone who motivated and supported me during this process. Without their encouragement, this work would not have been possible.

Abstract

Hand gestures are among the most natural and intuitive forms of human communication, playing a vital role not only in everyday interaction but also in specialized domains such as sign language, virtual reality, and human–robot collaboration. The ability to automatically recognize and compare hand gestures is therefore a key enabler for next-generation human–computer interfaces. However, hand gesture recognition has challenges such as high dimensionality, temporal complexity, and variations in execution. This thesis addresses the task of 3D skeleton-based hand gesture comparison, aiming to determine whether two gesture sequences represent the same action.

A spatio-temporal transformer architecture is employed to model both the structural relationships of hand joints and the temporal evolution of gestures. Using a metric learning objective with triplet loss, the model learns an embedding space where similar gestures are clustered closely and dissimilar ones are separated. Experiments on the FPHAB dataset demonstrate that the approach effectively learns discriminative embeddings for gesture sequences. Further analysis highlights the influence of embedding dimensionality, margin selection, and temporal context on model accuracy.

These findings show that transformer-based architectures are well-suited for skeleton-based gesture recognition, offering a strong foundation for intuitive gesture-driven interaction in future applications.

Keywords: Triplet Loss, Transformer-Based Architectures, Skeleton Data, Spatio-Temporal, Gesture Classification, Gesture Comparison

Contents

List of Figures

List of Tables

Listings

Abbreviations

HCI	Human–Computer Interaction
HGR	Hand Gesture Recognition
FPHAB	First-Person Hand Action Benchmark
3D	Three-Dimensional
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory Network
GNN	Graph Neural Network
ST-GCN	Spatial–Temporal Graph Convolutional Network
ViT	Vision Transformer
MLP	Multilayer Perceptron
ReLU	Rectified Linear Unit
t-SNE	t-Distributed Stochastic Neighbor Embedding

Chapter 1. Introduction

1.1 Overview

The development of intuitive and natural interaction technologies has been a central theme in human–computer interaction (HCI) for decades. From the early days of computing, where punch cards and command-line interfaces restricted accessibility, to the proliferation of graphical user interfaces and touchscreens, each stage of progress has aimed to reduce the cognitive and physical gap between humans and machines. Yet, despite these advancements, conventional input devices such as keyboards, mice, and touchscreens remain fundamentally limited. They require explicit physical manipulation and often fail to capture the rich expressiveness of human communication.

The human hand, with its fine motor skills and symbolic gestures, represents one of the most powerful tools for interaction. Gestures are deeply ingrained in human culture, serving not only as supplements to spoken language but also as standalone systems of communication (e.g., sign languages). Recognizing and interpreting gestures therefore holds enormous potential for enabling more natural, seamless, and inclusive human–computer interfaces. In particular, dynamic hand gestures - sequences of movements performed over time are especially expressive and capable of conveying complex meanings. These range from simple commands such as "swipe left" to nuanced communicative acts in sign language or symbolic control of robots.

In recent years, the rise of augmented and virtual reality (AR/VR), smart environments, and assistive technologies has intensified interest in gesture-based interaction. In these contexts, traditional devices are not only cumbersome but sometimes infeasible. For example, in a surgical setting, touch-based controls may compromise sterility, whereas hand gestures provide a sterile, contactless interface. Similarly, in immersive VR environments, physical controllers disrupt the illusion of natural presence, while free-hand interaction offers a more compelling alternative. Gesture recognition thus sits at the frontier of next-generation HCI.

1.2 Background Context

Hand gesture recognition is a subfield of computer vision and pattern recognition that seeks to automatically detect, classify, and interpret hand movements. Broadly, gestures can be categorized into:

- **Static gestures:** where the recognition task involves classifying a single hand pose (e.g., thumbs-up, stop sign).

- **Dynamic gestures:** where recognition requires analyzing a sequence of poses over time (e.g., waving, clapping, or opening a jar).

Static gesture recognition has seen significant success with classical machine learning approaches and early deep learning models. However, dynamic gestures are considerably more complex, as they require capturing temporal dynamics in addition to spatial structure. The meaning of a gesture may depend not only on the shape of the hand at each frame but also on the trajectory, speed, and rhythm of its movement.

Different data modalities have been explored for gesture recognition, including RGB video, depth maps, optical flow, and skeleton representations. RGB-based methods provide rich appearance information but are sensitive to lighting and background clutter. Depth-based methods improve robustness but require specialized sensors. Skeleton-based approaches abstract the hand into joint positions, offering compact, interpretable, and noise-resistant features. With the advent of powerful pose estimation algorithms (e.g., OpenPose), skeleton data can now be extracted reliably from standard video, broadening its applicability.

Early research in gesture recognition relied on handcrafted features (e.g., HOG, SIFT, motion vectors) combined with traditional classifiers such as Hidden Markov Models (HMMs) or Support Vector Machines (SVMs). While effective in constrained settings, these methods struggled with generalization in real-world conditions. The rise of deep learning, particularly Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), marked a turning point, enabling end-to-end learning of both spatial and temporal features. More recently, transformer-based architectures have further advanced the state of the art by introducing self-attention mechanisms capable of capturing long-range dependencies across entire gesture sequences.

1.3 Motivation

The motivation for this research arises from both practical needs and technical challenges. Practically, gesture recognition holds promise in domains as varied as AR/VR interaction, healthcare, sign language interpretation, and human–robot collaboration. Technically, the problem remains unsolved due to several key challenges:

1. **High dimensionality:** A hand model often includes over 21 joints, each with 3D coordinates. A gesture sequence spanning hundreds of frames results in a very high-dimensional input space.
2. **Temporal dependencies:** The semantics of a gesture depend on timing, rhythm, and ordering of movements, which must be modeled over long sequences.
3. **Inter-subject variability:** Different individuals perform gestures with variations in style, speed, and scale, introducing ambiguity.

4. **Occlusions and noise:** Hands may be partially occluded or joints inaccurately detected, especially in cluttered real-world environments.

These challenges highlight the need for models that can jointly capture spatial and temporal dependencies, generalize across subjects, and remain robust to noise and occlusion. Transformer-based architectures, with their global self-attention mechanism, provide a promising solution.

1.4 Problem Statement

This thesis focuses on the task of *dynamic 3D hand gesture comparison*, i.e., determining whether two sequences of hand joint movements represent the same underlying gesture. Unlike standard classification, which maps inputs to predefined labels, comparison requires learning an embedding space where similarity is measured directly. The central problem is therefore:

How can a transformer-based spatial–temporal model, trained with a metric learning objective such as triplet loss, be adapted to learn compact and discriminative embeddings for dynamic 3D hand gestures?

1.5 Research Questions

This work is guided by the following research questions:

1. **RQ1:** How can a transformer architecture be adapted to process sequences of 3D hand joint coordinates effectively?
2. **RQ2:** To what extent does triplet-loss-based training enable the model to learn a discriminative embedding space for gesture comparison?
3. **RQ3:** How do design choices such as embedding dimensionality, margin value, sliding window size, and distance metric influence recognition performance?
4. **RQ4:** What evaluation metrics and visualization strategies best capture the quality and separability of gesture embeddings?

1.6 Aims and Objectives

The primary aim of this thesis is implement transformer-based architecture for dynamic hand gesture comparison. To achieve this, the following objectives are defined:

- Using Spatio-Temporal model architecture to process sequences of 3D hand joint coordinates.
- Implement temporal modeling strategies (e.g., sliding windows) to capture dependencies in variable-length sequences.
- Train the model with a triplet loss objective to learn a discriminative embedding space.
- Analyze the effect of hyperparameters such as embedding size, margin, and distance metrics.

- Analyze the results from experiments and visualize learned embeddings with techniques such as t-SNE.

1.7 Contributions

The main contributions of this thesis are as follows: An adaptation of the Spatio-Temporal Transformer architecture for processing sequences of 3D hand joints in gesture recognition. A methodology for training the model with a triplet loss objective to learn a discriminative embedding space for gesture comparison. An empirical evaluation of how temporal modeling strategies and hyperparameters influence performance. A comprehensive evaluation protocol, including both quantitative metrics and qualitative visualization of gesture embeddings.

1.8 Thesis Structure

The remainder of this thesis is organized as follows:

- **Chapter 3: Theoretical Background** introduces the foundational concepts in computer vision, deep learning, and transformers.
- **Chapter 4: Related Work** highlights literature review in this domain.
- **Chapter 4: Methodology and Implementation** outlines our methodology.
- **Chapter 5: Experimental Setup** explains data preprocessing, training configurations, and evaluation metrics.
- **Chapter 6: Results and Analysis** reports results from gesture classification and gesture comparison experiments.
- **Conclusion and Future Work** summarizes contributions, discusses limitations, and suggests future directions.

Chapter 2. Theoretical Background

2.1 Introduction to Image Processing

Image processing serves as the backbone of computer vision, providing the foundational techniques for preparing, enhancing, and transforming raw visual data into forms suitable for analysis. While modern computer vision systems rely heavily on deep learning to learn representations directly from data, many of these models still benefit from preprocessing steps rooted in classical image processing. In gesture recognition tasks, where input data may suffer from noise, variable lighting, or background clutter, these methods play a vital role in ensuring reliable downstream performance.

2.1.1 Image Enhancement

The goal of image enhancement is to improve the perceptual quality of an image or highlight important structures for subsequent processing. One widely used technique is *histogram equalization*, which redistributes the intensity values of an image so that the resulting histogram is more uniformly spread. By stretching the contrast across the full dynamic range, histogram equalization brings out details that would otherwise remain hidden in darker or brighter regions of the image. This is particularly useful in gesture recognition when the hand may appear under different illumination conditions, ensuring that the joint and finger boundaries remain visible.

Another category of enhancement methods involves filtering. Smoothing filters, such as the Gaussian filter, are applied to suppress high-frequency noise that often arises in captured images. The Gaussian kernel weights nearby pixels according to a normal distribution, effectively blurring the image while preserving its overall structure. In contrast, sharpening filters emphasize edges and fine textures by amplifying high-frequency components. Both types of filtering are valuable: smoothing stabilizes noisy skeleton or contour data, while sharpening ensures that critical edges of the hand remain distinguishable.

Normalization is another important step, particularly in the context of machine learning models. By scaling pixel values or joint coordinates into a consistent range (e.g., $[0, 1]$), normalization ensures stable gradient descent during training and avoids numerical instabilities. For dynamic hand gesture sequences, normalizing each frame also reduces the impact of scale differences between users or recording setups.

2.1.2 Edge Detection and Feature Extraction

Detecting edges is one of the earliest and most fundamental tasks in computer vision. Edges correspond to significant local changes in intensity, often indicating object boundaries, surface orientations, or texture transitions. In the context of hand gesture recognition, edges help delineate the outline of the hand and the separation between fingers, providing critical cues for pose estimation.

Several classical operators have been developed for edge detection. Gradient-based operators such as the Sobel and Prewitt filters compute intensity derivatives in horizontal and vertical directions, highlighting regions of abrupt change. These methods are simple yet effective for identifying basic contours. A more sophisticated approach is the *Canny edge detector*, which applies Gaussian smoothing, computes gradients, performs non-maximum suppression to thin out edges, and uses hysteresis thresholding to retain only the most significant boundaries. The result is a clean and well-connected edge map that can guide subsequent segmentation or pose estimation steps.

Although deep learning now dominates feature extraction, classical edge detection remains relevant as a preprocessing tool or as a way to validate the accuracy of skeleton extraction by comparing joint positions with detected hand contours.

2.1.3 Image Segmentation

Segmentation is the process of partitioning an image into meaningful regions, such as isolating the hand from the background. In dynamic gesture recognition, segmentation ensures that only the region of interest (the hand) is analyzed, reducing the influence of distracting background elements.

One of the simplest approaches to segmentation is *thresholding*, where pixels are classified as foreground or background depending on whether their intensity values fall above or below a chosen threshold. Adaptive thresholding extends this idea by computing thresholds locally for different regions of the image, which is more effective under uneven lighting. Region-based methods go further by grouping pixels based on similarity in color or texture, forming coherent regions that may correspond to objects. Clustering algorithms, such as k-means, treat segmentation as a grouping problem in feature space, where pixels are clustered into groups that represent different parts of the image.

For gesture recognition, segmentation is critical when working with RGB data, as it isolates the hand shape. When depth cameras are used, segmentation can be greatly simplified by selecting pixels within a specific depth range corresponding to the hand. Regardless of modality, segmentation ensures that the extracted features originate from the relevant region, improving the accuracy and efficiency of recognition.

2.1.4 Morphological Operations

Morphological operations are a family of techniques that process binary or segmented images based on shape. These methods apply structuring elements to modify the geometry of objects within an image. For example, *erosion* removes small foreground regions or shrinks boundaries, effectively eliminating noise or thin protrusions. Conversely, *dilation* expands foreground regions, filling in small gaps or holes. More complex operations, such as *opening* (erosion followed by dilation) and *closing* (dilation followed by erosion), are used to smooth object boundaries while preserving their overall structure.

In gesture recognition systems, morphological operations can refine the hand silhouette obtained from segmentation, ensuring that finger gaps are preserved while spurious background noise is removed. These clean silhouettes can then be used to extract contours, skeletons, or other structural descriptors.

2.1.5 Relevance to Gesture Recognition

Although the current state-of-the-art relies heavily on deep neural networks, the role of classical image processing techniques cannot be underestimated. They serve as the first stage in many recognition pipelines, ensuring that the input data is of sufficient quality for learning algorithms. In skeleton-based gesture recognition, preprocessing often includes smoothing trajectories to reduce jitter, normalizing joint coordinates to account for subject size variations, and aligning skeletons across frames to a common reference point. These steps are conceptually similar to traditional image processing, but applied in the domain of structured data rather than pixels.

Moreover, image processing techniques remain indispensable when validating or augmenting pose estimation outputs. For instance, edge detection can be used to verify whether predicted joint locations lie along visible hand contours, while segmentation masks can constrain the search space for pose estimators. Thus, even as deep learning advances, classical image processing continues to play a supporting role in building reliable and robust gesture recognition systems.

2.2 Introduction to Computer Vision

Computer Vision (CV) is an interdisciplinary field that focuses on enabling machines to interpret, analyze, and understand visual data from the world. In its essence, computer vision attempts to replicate the remarkable ability of the human visual system, which can seamlessly perceive objects, interpret scenes, and understand motion. The ultimate goal of CV is to extract meaningful information from images or videos and use this information for decision-making, recognition, and interaction.

The field of computer vision has evolved considerably over the past decades. Early research in the 1960s and 1970s concentrated on low-level image processing techniques such as edge

detection, segmentation, and shape analysis. These methods relied heavily on handcrafted features and heuristic algorithms. Although useful in controlled environments, they often failed under real-world conditions where variations in illumination, viewpoint, or occlusion occur.

With the advent of machine learning in the late 1990s and early 2000s, computer vision started shifting from purely handcrafted features (such as SIFT and HOG) to data-driven approaches. These methods allowed for better generalization across different visual tasks by learning discriminative features from labeled datasets. A major milestone came in 2012 with the success of AlexNet (13), a deep Convolutional Neural Network (CNN) that achieved unprecedented accuracy in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). Since then, deep learning has become the dominant paradigm in computer vision, powering applications ranging from object detection and facial recognition to medical image analysis.

Another important dimension in computer vision is the study of motion and dynamics, particularly in video analysis. Recognizing human activities, gestures, and behaviors requires not only spatial understanding of static frames but also temporal modeling of sequences. This shift from static image analysis to spatio-temporal understanding has led to the integration of models such as Recurrent Neural Networks (RNNs) and, more recently, Transformers, which excel at capturing long-term dependencies.

Figure 2.1 provides a high-level overview of the history of computer vision. This transition highlights how the field has matured from handcrafted rules to data-driven representations capable of handling complex real-world challenges.

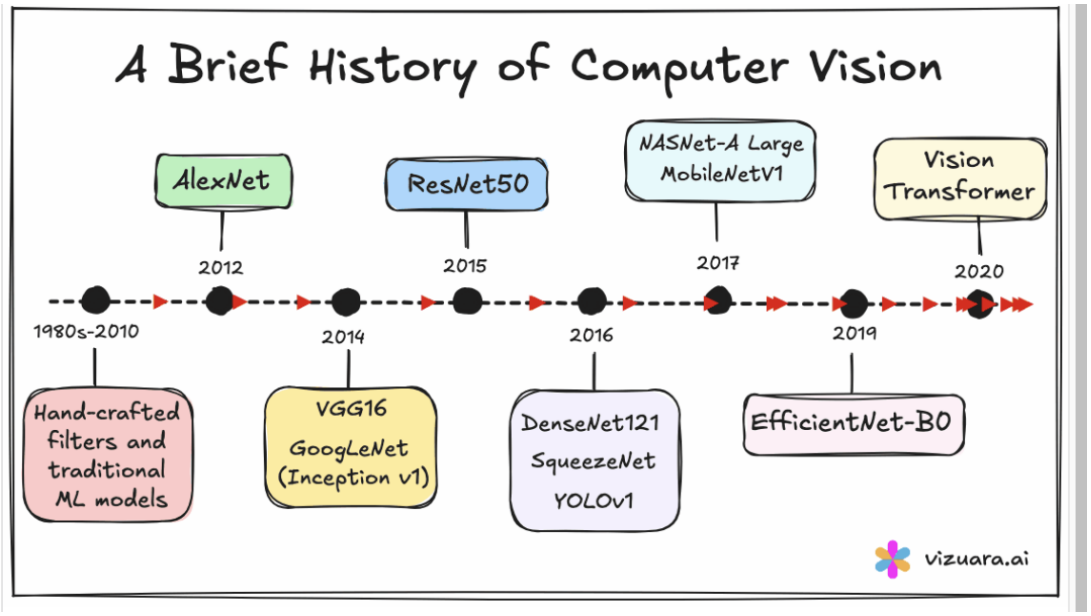

FIGURE 2.1: Evolution of computer vision: from classical handcrafted features (edges, corners) to transformer-based methods for spatio-temporal modeling.

2.2.1 Key Challenges in Computer Vision

Despite the tremendous progress made in recent years, computer vision still faces fundamental challenges. Variations in illumination, occlusion, and changes in viewpoint remain significant obstacles for recognition systems. For example, the same hand gesture may appear drastically different depending on whether it is captured in daylight, indoors, or under shadows. Similarly, partial occlusion of the hand can obscure crucial details, making recognition difficult. These challenges highlight the need for models that are not only accurate but also robust and invariant to real-world variability.

Another critical issue is scalability. While deep learning methods achieve impressive results when trained on large annotated datasets, collecting and labeling such data is expensive and time-consuming. This problem is particularly evident in specialized tasks such as dynamic hand gesture recognition, where high-quality annotated datasets are limited. Transfer learning, data augmentation, and self-supervised learning have been proposed as solutions, but scaling recognition systems to diverse environments remains an open problem.

2.2.2 Applications of Computer Vision

Computer vision plays a central role across a broad spectrum of applications. In healthcare, it supports medical imaging tasks such as tumor detection, X-ray analysis, and gait monitoring for rehabilitation. In autonomous driving, vision systems interpret traffic signs, pedestrians, and obstacles in real time. In the entertainment industry, CV powers augmented and virtual reality systems, allowing users to interact with digital environments in natural ways. Gesture recognition, in particular, is a promising application, as it provides an intuitive interface for interacting with machines without physical contact. From gaming to assistive technologies for individuals with disabilities, gesture-based systems are increasingly becoming part of everyday life.

2.2.3 Gesture Recognition as a Subfield of Computer Vision

Gesture recognition lies at the intersection of computer vision, machine learning, and human-computer interaction. Hand gestures serve as a natural modality for communication and control, enabling intuitive interaction with digital devices. In this context, computer vision systems must detect hands, extract meaningful features, and compare gesture patterns across time and individuals. The ability to compare gestures robustly opens up applications in sign language translation, rehabilitation monitoring, virtual reality, and biometric authentication.

Recent progress in deep learning has significantly advanced gesture recognition. Early approaches relied on handcrafted features such as optical flow or shape descriptors to capture motion. However, these methods often struggled with variations in execution style or environmental conditions. Modern approaches leverage spatio-temporal deep learning architectures—such as

convolutional networks, recurrent models, and transformers—that can capture both the local structure of gestures and their temporal evolution. Skeleton-based representations, in particular, have emerged as a powerful modality, offering robustness to variations in appearance and lighting.

In summary, computer vision has progressed from simple image processing techniques to sophisticated deep learning models capable of interpreting highly complex visual patterns. While challenges such as robustness, scalability, and data efficiency remain, the field continues to advance rapidly, opening new opportunities for human-computer interaction. Gesture recognition, as an application of computer vision, illustrates both the progress made and the challenges that remain, making it an ideal testbed for advancing state-of-the-art methods.

2.3 Image Formation and Camera Models

The first step in understanding computer vision is to study how images are formed. A camera captures a three-dimensional (3D) scene and projects it onto a two-dimensional (2D) image plane. This process, known as the *image formation model*, provides the mathematical relationship between 3D points in the real world and their corresponding 2D image coordinates. Accurate modeling of this projection process is essential for tasks such as camera calibration, 3D reconstruction, and gesture recognition.

2.3.1 The Pinhole Camera Model

The simplest and most widely used image formation model is the *pinhole camera model*. In this model, a 3D point $\mathbf{X} = (X, Y, Z)^T$ in the world is projected onto a 2D point $\mathbf{x} = (x, y)^T$ on the image plane.

The projection is governed by the principle of perspective geometry:

$$x = f \cdot \frac{X}{Z}, \quad y = f \cdot \frac{Y}{Z}$$

where f is the focal length of the camera (i.e., the distance between the pinhole and the image plane). This model assumes that light rays pass through a single optical center before reaching the image plane (24).

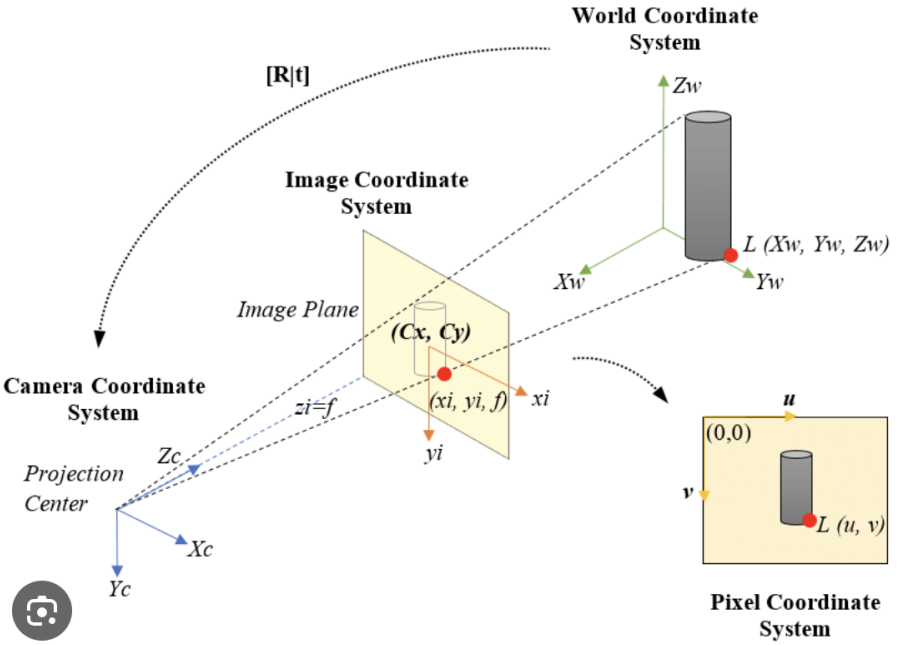

FIGURE 2.2: The pinhole camera model: a 3D point (X, Y, Z) is projected onto a 2D point (x, y) on the image plane using perspective projection.

2.3.2 Intrinsic Parameters

The intrinsic parameters of a camera define how the camera maps 3D rays to pixel coordinates in the image plane. The intrinsic matrix $\mathbf{K}$ is typically written as:

$$\mathbf{K} = \begin{bmatrix} f_x & \gamma & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

where:

- f_x, f_y: focal lengths in terms of pixels along the x and y axes,
- (c_x, c_y): coordinates of the principal point (usually near the image center),
- γ: skew parameter (accounts for non-orthogonal image axes, typically zero).

2.3.3 Extrinsic Parameters

The extrinsic parameters describe the position and orientation of the camera with respect to the world coordinate system. They are represented by a rotation matrix $\mathbf{R}$ and a translation vector $\mathbf{t}$:

$$\mathbf{P} = \mathbf{K}[\mathbf{R} \mid \mathbf{t}]$$

where $\mathbf{P}$ is the complete 3×4 projection matrix mapping 3D world coordinates to 2D image coordinates (10).

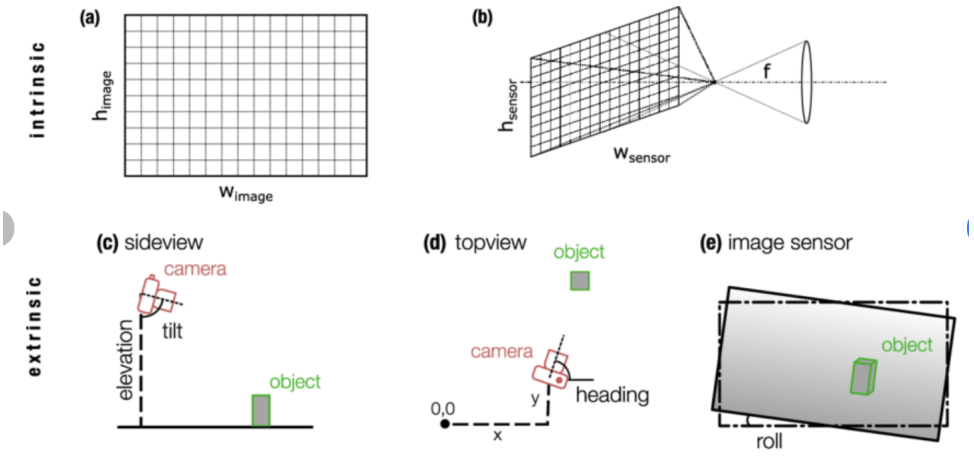

FIGURE 2.3: Illustration of camera intrinsic and extrinsic parameters. (8)

2.3.4 Lens Distortion

In real cameras, the pinhole model is an approximation. Lenses introduce distortions that must be corrected for accurate computer vision applications. Two common types of distortion are:

- **Radial distortion:** Causes straight lines to appear curved, especially near the image edges. It is typically modeled by polynomial terms:

$$x_{distorted} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6), \quad y_{distorted} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

  where $r^2 = x^2 + y^2$ and k_1, k_2, k_3 are distortion coefficients.
- **Tangential distortion:** Caused by imperfect lens alignment, modeled as:

$$x_{distorted} = x + [2p_1 xy + p_2(r^2 + 2x^2)], \quad y_{distorted} = y + [p_1(r^2 + 2y^2) + 2p_2 xy]$$

  where p_1, p_2 are tangential distortion parameters.

2.3.5 Camera Calibration

Camera calibration is the process of estimating the intrinsic and extrinsic parameters as well as lens distortion coefficients. A common method involves capturing multiple images of a known calibration pattern, such as a checkerboard, and solving for the parameters using optimization techniques. Accurate calibration is essential in 3D reconstruction and gesture recognition, where precise spatial measurements are required.

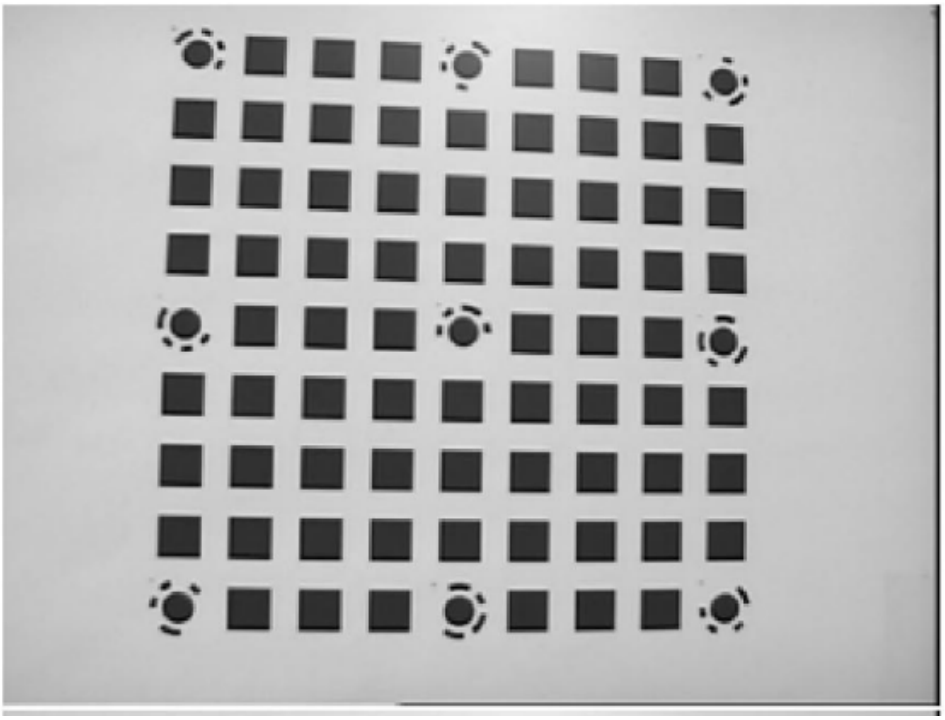

FIGURE 2.4: Camera calibration using a checkerboard pattern. By observing known 3D points and their 2D projections, intrinsic and extrinsic parameters can be estimated. (29)

2.4 Multi-View Geometry

Computer vision often involves analyzing a scene from multiple viewpoints. Understanding the geometric relationship between images captured from different camera poses is essential for tasks such as 3D reconstruction, structure-from-motion, and gesture analysis. Two fundamental concepts in multi-view geometry are *homography* and *epipolar geometry*.

2.4.1 Homography

A homography describes the mapping between two images of the same planar surface or between images taken from different viewpoints under pure camera rotation. Mathematically, the relation between a point $\tilde{\mathbf{x}}$ in one image and its corresponding point $\tilde{\mathbf{x}}'$ in another is expressed as:

$$\tilde{\mathbf{x}}' = \mathbf{H}\tilde{\mathbf{x}}$$

where $\mathbf{H}$ is a 3×3 homography matrix and $\tilde{\mathbf{x}}, \tilde{\mathbf{x}}'$ are homogeneous image coordinates.

Homographies are widely used for tasks such as image stitching, planar rectification, and camera pose estimation. For instance, in panoramic image stitching, multiple overlapping images are aligned by estimating the homography between them (10).

2.4.2 Epipolar Geometry

While homography applies to planar surfaces or rotations, epipolar geometry describes the relationship between two arbitrary views of a 3D scene. Given two cameras observing the same 3D point, the projection of that point in each image is constrained to lie on a corresponding pair of epipolar lines.

The fundamental relationship is expressed by the *epipolar constraint*:

$$\tilde{\mathbf{x}}'^T \mathbf{F} \tilde{\mathbf{x}} = 0$$

where $\mathbf{F}$ is the 3×3 fundamental matrix, $\tilde{\mathbf{x}}$ is the homogeneous coordinate of a point in the first image, and $\tilde{\mathbf{x}}'$ is the corresponding point in the second image.

If the intrinsic parameters of both cameras are known, the relationship is represented using the *essential matrix* $\mathbf{E}$:

$$\tilde{\mathbf{x}}'^T \mathbf{E} \tilde{\mathbf{x}} = 0$$

with $\mathbf{E} = \mathbf{K}^T \mathbf{F} \mathbf{K}$, where $\mathbf{K}$ is the intrinsic calibration matrix.

Epipolar geometry is fundamental in stereo vision, enabling depth estimation by triangulating corresponding points across two views. It is also crucial for gesture recognition tasks when multiple cameras are used to capture motion sequences (17).

2.4.3 Relevance to Gesture Analysis

In gesture recognition, homography can be applied for normalizing hand regions under viewpoint changes, while epipolar geometry facilitates robust depth estimation in multi-camera setups. Together, these concepts provide the geometric foundation for higher-level tasks such as 3D skeleton reconstruction and motion analysis.

2.5 Structure-from-Motion and Depth Estimation

While homography and epipolar geometry describe relationships between two views, structure-from-motion (SfM) extends these ideas to multiple views, allowing the recovery of both the 3D structure of a scene and the relative motion of the camera. This is particularly relevant in gesture recognition, where hand or body movements can be reconstructed in three dimensions from video sequences.

2.5.1 Structure-from-Motion (SfM)

Structure-from-Motion refers to the process of estimating 3D scene geometry and camera motion simultaneously from a set of 2D image correspondences (25). The typical pipeline involves:

1. Detecting and matching features (e.g., SIFT, ORB) across multiple images.

2. Estimating the fundamental or essential matrix between image pairs.
3. Recovering camera poses and triangulating 3D points.
4. Refining the reconstruction using *bundle adjustment*, which minimizes the reprojection error across all views.

The mathematical formulation of bundle adjustment can be expressed as:

$$\min_{\mathbf{R},\mathbf{t},\mathbf{X}} \sum_{i,j} \|\mathbf{x}_{ij} - \pi(\mathbf{R}_i, \mathbf{t}_i, \mathbf{X}_j)\|^2$$

where $\mathbf{x}_{ij}$ is the observed projection of 3D point $\mathbf{X}_j$ in camera i, and π is the projection function parameterized by the camera pose $(\mathbf{R}_i, \mathbf{t}_i)$.

2.5.2 Depth Estimation

Depth estimation refers to the process of inferring the distance of objects from the camera. Traditional stereo vision uses two calibrated cameras to triangulate depth, while modern methods employ learning-based monocular depth estimation.

The depth Z of a point can be computed in stereo vision as:

$$Z = \frac{f \cdot B}{d}$$

where f is the focal length, B is the baseline (distance between the two cameras), and d is the disparity between corresponding points in the two images.

Recent advances use deep learning to estimate depth from a single image by training neural networks on large datasets with ground-truth depth maps. These methods are particularly useful in gesture recognition scenarios where specialized depth cameras (e.g., Microsoft Kinect) or monocular cameras are employed (6).

2.5.3 Relevance to Gesture Analysis

For gesture recognition, depth information provides an additional modality that significantly improves robustness against illumination changes and cluttered backgrounds. Structure-from-motion can be used to reconstruct 3D motion trajectories of the hands, while depth maps facilitate accurate segmentation and skeleton extraction. These geometric cues complement deep learning models in capturing the dynamics of human gestures.

2.6 Tracking and Motion Analysis

Understanding motion is a cornerstone of gesture recognition. While static image analysis provides spatial cues such as shape and texture, gestures are inherently dynamic and require temporal modeling of motion. Over the years, several approaches have been developed to track

objects (such as hands) and analyze their motion across sequences of frames. These methods can be broadly categorized into classical model-based tracking and modern learning-based motion estimation.

2.6.1 Optical Flow

Optical flow is one of the oldest and most widely used techniques for estimating motion in videos. It describes the apparent displacement of brightness patterns between consecutive frames, providing dense or sparse motion fields. Early formulations such as the Lucas–Kanade method and the Horn–Schunck algorithm remain foundational in computer vision.

The Lucas–Kanade method assumes that the intensity of a pixel remains constant between two frames and approximates motion by solving a system of equations over a small neighborhood. This method is well-suited for tracking small, slow motions but fails when large displacements or occlusions occur. On the other hand, the Horn–Schunck method imposes a global smoothness constraint, resulting in dense optical flow fields. Although more robust to noise, it is computationally expensive and sensitive to illumination changes. Later improvements, such as Farneback's polynomial expansion approach, have enabled real-time dense optical flow, making it practical for gesture analysis.

In the context of gesture recognition, optical flow captures the fine-grained motion of hands and fingers. Representations such as Motion History Images (MHI) and Motion Energy Images (MEI) are derived from optical flow to encode temporal dynamics into compact forms. These representations have been successfully used in early gesture recognition systems, providing a bridge between raw video input and feature-based classification.

2.6.2 Kalman Filtering for Hand Tracking

When continuous tracking of hand trajectories is required, filtering techniques such as the Kalman Filter are highly effective. The Kalman Filter is a recursive estimator that predicts the current state of a system based on its previous state and a new noisy observation. It models the dynamics of motion using a state-space representation:

$$\mathbf{x}_t = \mathbf{A}\mathbf{x}_{t-1} + \mathbf{w}_{t-1}, \quad \mathbf{z}_t = \mathbf{H}\mathbf{x}_t + \mathbf{v}_t$$

Here, $\mathbf{x}_t$ represents the hidden state (e.g., the position and velocity of a hand), $\mathbf{z}_t$ the observed measurement, $\mathbf{A}$ the state transition matrix, and $\mathbf{H}$ the observation matrix. The terms $\mathbf{w}_t$ and $\mathbf{v}_t$ denote process and measurement noise, respectively, assumed to be Gaussian.

Kalman Filtering is particularly suitable for linear, Gaussian systems, where it provides an optimal solution in the least-squares sense. In gesture recognition, it is often used to stabilize hand trajectories obtained from video or depth cameras, filtering out noise from pose estimation

algorithms. For example, jitter in the detected position of a fingertip can be smoothed, ensuring that the downstream recognition model receives consistent inputs.

2.6.3 Particle Filters and Nonlinear Tracking

For more complex scenarios where hand motion exhibits nonlinear dynamics or when occlusions occur, the Kalman Filter becomes insufficient. Particle Filters, also known as Sequential Monte Carlo methods, extend the idea of recursive estimation by representing the posterior distribution of the state with a set of weighted samples, or particles.

Each particle hypothesizes a possible state of the system, and weights are updated based on how well the predicted observation matches the actual measurement. Over time, particles converge around the true state of the system. Particle Filters are highly flexible and can handle multimodal distributions, making them robust to ambiguities in hand tracking, such as when two hands cross paths or when partial occlusion occurs.

In gesture recognition, Particle Filters have been used to track not only the global position of the hand but also its articulated structure, such as joint angles. This makes them particularly valuable in skeleton-based approaches, where precise localization of joints is critical for accurate recognition.

2.6.4 Trajectory Analysis

Once hand motion is tracked reliably, trajectory analysis becomes an important tool for recognizing gestures. A trajectory is defined as the path traced by a point (e.g., wrist or fingertip) over time in two- or three-dimensional space. Trajectories can be represented parametrically, such as splines, or encoded as time series.

Several techniques have been developed to compare and classify motion trajectories. Dynamic Time Warping (DTW) is a classical method that aligns two sequences by stretching or compressing them in time to minimize a distance measure. This is particularly useful for gestures, where the same action may be performed at different speeds. Another common approach involves representing trajectories in feature spaces such as curvature, velocity, and acceleration, and then applying machine learning classifiers.

In skeleton-based gesture recognition, trajectory analysis often focuses on high-level motion descriptors, such as the relative displacement of joints. For example, the trajectory of the wrist relative to the elbow can differentiate between a pointing gesture and a waving gesture. By combining trajectory features with spatio-temporal models, recognition systems achieve greater robustness and interpretability.

2.6.5 Relevance to Gesture Recognition

Tracking and motion analysis form the theoretical backbone of dynamic gesture recognition. Optical flow provides pixel-level motion cues, Kalman Filters ensure stable tracking, Particle Filters handle nonlinearities and occlusions, and trajectory analysis bridges raw motion with semantic gesture categories. Although many of these classical methods have been superseded by deep learning approaches, they continue to provide valuable insights and remain integral in hybrid systems. Moreover, the ideas of temporal smoothness, uncertainty modeling, and trajectory alignment directly influence modern architectures such as Recurrent Neural Networks and Transformers.

2.7 Mathematical Foundations of Computer Vision

Computer vision is an applied discipline, but its theoretical underpinnings lie firmly in mathematics. Every algorithm, from simple edge detectors to deep learning–based architectures, ultimately relies on mathematical abstractions for representing data, analyzing relationships, and optimizing solutions. In particular, three areas of mathematics are indispensable for modern vision research: **linear algebra**, which provides the language for representing and manipulating images and geometric transformations; **probability theory and statistics**, which allow us to reason under uncertainty and variability; and **optimization**, which defines the process of learning and fine-tuning models from data. This section introduces these mathematical principles in detail, emphasizing their relevance for gesture recognition tasks.

2.7.1 Linear Algebra in Computer Vision

Linear algebra plays a fundamental role in almost every computer vision application. Images, videos, and skeletons are naturally represented in matrix or tensor form, and transformations between coordinate systems are elegantly expressed using matrix operations. Without linear algebra, it would be impossible to formalize the transition from the three-dimensional real world to the two-dimensional image plane.

2.7.1.1 Vectors and Matrices

An image can be viewed as a structured arrangement of numbers. A grayscale image of size $H \times W$ is modeled as a matrix $\mathbf{I} \in \mathbb{R}^{H \times W}$, where each entry corresponds to a pixel intensity. In the case of color images, each pixel contains three values (for the red, green, and blue channels), leading to a tensor $\mathbf{I} \in \mathbb{R}^{H \times W \times 3}$. For videos, time adds a fourth dimension, yielding $\mathbf{V} \in \mathbb{R}^{H \times W \times 3 \times T}$, where T is the number of frames.

The power of linear algebra lies in the ability to apply transformations concisely. Consider a 2D point $\mathbf{x} = (x, y, 1)^T$ expressed in homogeneous coordinates. Scaling, rotation, or translation of the point can all be written as matrix multiplications. For example, a rotation by angle θ is

given by:

$$\mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{x}' = \mathbf{R}\mathbf{x}.$$

This concise representation enables robust handling of geometric problems such as camera calibration, image alignment, and 3D reconstruction.

2.7.1.2 Eigenvalues and Eigenvectors

Eigenvalues and eigenvectors capture the fundamental modes of variation in data. In computer vision, they are most prominently used in Principal Component Analysis (PCA). PCA projects high-dimensional data into a lower-dimensional subspace while retaining as much variance as possible. For gesture recognition, PCA can compress high-dimensional skeleton sequences into compact descriptors that preserve meaningful motion patterns. This dimensionality reduction both accelerates computation and helps filter out noise.

2.7.1.3 Singular Value Decomposition (SVD)

The Singular Value Decomposition is one of the most powerful tools in numerical linear algebra. It decomposes a matrix $\mathbf{A}$ into:

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T,$$

where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices and Σ contains the singular values. SVD has numerous applications in computer vision: it is central to solving systems of linear equations, estimating the fundamental matrix in stereo vision, and reducing noise in visual data. In gesture recognition, SVD can be used to factorize motion trajectories into dominant components, allowing us to capture the most significant temporal dynamics while ignoring small perturbations.

2.7.2 Probability and Statistics in Computer Vision

Real-world data is noisy and uncertain. A camera sensor might misread pixel intensities, an occlusion might hide part of the hand, or two people might perform the same gesture in slightly different ways. Probability and statistics provide the mathematical framework to model this uncertainty, making them indispensable for gesture recognition systems.

2.7.2.1 Bayesian Inference

Bayesian methods enable reasoning about uncertainty by combining prior knowledge with observed evidence. Bayes' theorem is expressed as:

$$P(H|D) = \frac{P(D|H)P(H)}{P(D)},$$

where H is a hypothesis and D is the observed data. In gesture recognition, the hypothesis H could represent a particular gesture class, while the data D corresponds to observed joint positions. If we know that certain gestures are more likely to occur (prior knowledge), Bayesian inference allows us to update our belief when new sensor readings are observed. This framework naturally incorporates uncertainty and is particularly powerful when dealing with incomplete or ambiguous data.

2.7.2.2 Gaussian Distributions

The Gaussian distribution is perhaps the most widely used probabilistic model in computer vision. Many natural processes, such as measurement noise from a camera sensor, can be approximated by Gaussian statistics. The multivariate Gaussian distribution extends this idea to multiple dimensions, which is particularly relevant for modeling skeleton data. Each hand pose, represented as a vector of joint coordinates, can be modeled as a sample from a multivariate Gaussian distribution, where the covariance matrix encodes the correlations between different joints.

2.7.2.3 Markov Models and Temporal Dynamics

Gestures unfold over time, and understanding temporal dependencies is crucial. Classical approaches often relied on Hidden Markov Models (HMMs), where each hidden state corresponds to a stage of the gesture, and transitions between states model the temporal evolution. Although HMMs have largely been supplanted by deep learning architectures such as RNNs and transformers, they laid the conceptual foundation for sequential modeling. The Markovian assumption — that the current state depends only on the previous state — remains an important simplification in many vision algorithms.

2.7.3 Optimization in Computer Vision

Optimization is the engine that drives modern machine learning. Almost all vision algorithms, from fitting a line to a set of points to training a deep neural network with millions of parameters, can be expressed as optimization problems.

2.7.3.1 Gradient Descent and Its Variants

The most widely used optimization technique is gradient descent. Given a loss function $\mathcal{L}(\theta)$, the parameters θ are iteratively updated as:

$$\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}(\theta),$$

where η is the learning rate. Variants such as stochastic gradient descent (SGD), RMSProp, and Adam improve convergence by adjusting the step size or incorporating momentum. In

gesture recognition, these algorithms are used to minimize loss functions such as cross-entropy (for classification) or triplet loss (for embedding learning).

2.7.3.2 Convexity and Non-Convex Problems

Convex optimization guarantees global minima, making it easier to solve problems such as linear regression. However, deep neural networks introduce highly non-convex loss surfaces. Despite the theoretical difficulty, gradient-based methods often find good local minima in practice, which generalize well to unseen data. Understanding convexity provides intuition for why certain optimization strategies work better than others.

2.7.3.3 Regularization and Generalization

A persistent challenge in gesture recognition is the limited size of datasets, which increases the risk of overfitting. Regularization techniques such as L_2 penalties, dropout layers, and data augmentation help ensure that models generalize to unseen gestures. For example, random scaling, rotation, or temporal jittering of skeleton sequences during training can significantly improve robustness to variations in real-world scenarios.

2.7.4 Information Theory in Computer Vision

While linear algebra, probability, and optimization form the backbone of computer vision, another powerful mathematical tool that often goes unnoticed is **information theory**. Information theory provides a framework for quantifying uncertainty, similarity, and divergence between probability distributions, all of which are central to learning and recognition tasks. In gesture recognition, these principles are directly applied in designing loss functions, evaluating models, and understanding how much information is carried by skeleton sequences.

2.7.4.1 Entropy and Uncertainty

Entropy, introduced by Claude Shannon, is a measure of uncertainty in a random variable. For a discrete random variable X with distribution $p(x)$, the entropy is defined as:

$$H(X) = -\sum_{x} p(x) \log p(x).$$

In computer vision, entropy captures the unpredictability of features or predictions. For example, if a gesture recognition model outputs a nearly uniform probability distribution across classes, the entropy is high, reflecting uncertainty. In contrast, a confident prediction corresponds to low entropy. Thus, entropy can serve as an indicator of model confidence.

2.7.4.2 Cross-Entropy and Classification Loss

One of the most widely used loss functions in deep learning is *cross-entropy*, which measures the discrepancy between the true distribution $p(x)$ and the predicted distribution $q(x)$. It is

defined as:

$$H(p, q) = -\sum_{x} p(x) \log q(x).$$

In gesture classification, $p(x)$ corresponds to the one-hot encoded ground truth (the correct gesture class), while $q(x)$ is the predicted probability distribution from the model. Minimizing cross-entropy encourages the model to assign higher probability to the correct class. This connection highlights how information-theoretic principles directly translate into learning objectives.

2.7.4.3 Kullback–Leibler Divergence

The Kullback–Leibler (KL) divergence quantifies how one probability distribution diverges from another:

$$D_{\mathrm{KL}}(p\|q) = \sum_{x} p(x) \log \frac{p(x)}{q(x)}.$$

KL divergence is particularly relevant in generative models, regularization, and embedding learning. For gesture recognition, KL divergence can be used to enforce that embeddings of similar gestures (same class) remain close to each other in distributional terms, while dissimilar gestures remain apart. In this sense, KL divergence provides a probabilistic interpretation of similarity that complements metric-based approaches such as Euclidean or cosine distance.

2.7.4.4 Relevance to Gesture Recognition

Entropy, cross-entropy, and divergence measures all play critical roles in transformer-based gesture recognition systems. Entropy provides insight into model confidence, cross-entropy drives the optimization of classification tasks, and KL divergence helps align learned distributions with desired behavior. Together, these concepts extend the mathematical toolkit of computer vision, allowing researchers to design loss functions and evaluation criteria that are both principled and effective. In the context of skeleton-based gesture recognition, information theory underlines how much information is encoded in joint trajectories and how effectively a model extracts it.

2.8 Deep Learning in Computer Vision

The advent of deep learning has fundamentally reshaped the field of computer vision. Unlike classical computer vision techniques, which relied heavily on manually engineered features such as edges, corners, or descriptors like SIFT and HOG, deep learning approaches are data-driven and capable of automatically learning meaningful representations directly from raw image data. This shift from handcrafted features to learned features represents a paradigm change, one that has driven breakthroughs across nearly all areas of computer vision, including object detection, scene understanding, facial recognition, and gesture analysis.

The central idea behind deep learning is to build hierarchical models where higher layers capture increasingly abstract patterns. For instance, in the context of visual data, the first layers of a deep network may capture low-level features such as oriented edges or color blobs, while deeper layers learn more complex patterns such as textures, object parts, or even complete semantic categories. This ability to learn representations in a hierarchical manner makes deep learning especially powerful for computer vision tasks, where raw pixels alone do not provide sufficient information for robust reasoning.

2.8.1 Neural Networks: Building Blocks of Deep Learning

Artificial neural networks are loosely inspired by biological neurons. Each neuron takes a weighted sum of its inputs, adds a bias, and applies a nonlinear activation function. A single neuron is limited in expressive power, but when neurons are organized into layers and stacked deeply, the resulting network can approximate highly complex functions.

Formally, the output of a neuron for input vector $\mathbf{x}$ is defined as:

$$y = \sigma\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

where w_i are the learnable weights, b is the bias term, and σ is the activation function, which introduces non-linearity into the network. Without nonlinearities, the network would collapse into a linear transformation regardless of depth, severely limiting its representational capacity.

Training such networks involves minimizing a loss function — for instance, cross-entropy loss in classification tasks — using optimization algorithms such as stochastic gradient descent (SGD). The process of computing gradients and updating weights is performed by *backpropagation*, which is at the heart of deep learning. Through multiple iterations and exposure to large datasets, networks learn feature hierarchies that generalize well to unseen data.

2.8.2 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) represent the most important class of models in modern computer vision. The key idea behind CNNs is to exploit the spatial locality and structure of images. Unlike fully connected layers where every neuron connects to every input, convolutional layers use local receptive fields, meaning that each neuron processes a small patch of the input. This design dramatically reduces the number of parameters and captures the spatial hierarchies present in images.

The convolution operation is mathematically expressed as:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i-m, j-n) K(m,n)$$

where I is the input image, K is the convolution kernel (or filter), and $S(i, j)$ is the result of applying the kernel at location (i, j). Intuitively, different filters specialize in detecting specific patterns — for example, one filter may detect horizontal edges, another vertical edges, and others more complex textures.

Pooling layers, such as max pooling or average pooling, further reduce the spatial dimensions, providing translational invariance and reducing computation. Stacked convolutional and pooling layers result in feature maps that grow progressively more abstract. Finally, fully connected layers near the output combine these features for decision-making.

The impact of CNNs on computer vision cannot be overstated. LeNet-5, developed in the 1990s, pioneered CNNs for handwritten digit recognition. The real breakthrough came with AlexNet, which won the ImageNet competition in 2012 by a wide margin and demonstrated the effectiveness of deep CNNs trained on GPUs. Subsequent architectures such as VGGNet, GoogLeNet, and ResNet introduced deeper networks, improved parameter efficiency, and innovations such as residual connections, pushing the boundaries of recognition accuracy.

In the context of gesture recognition, CNNs serve as powerful spatial feature extractors. For instance, when applied to video frames of hand gestures, CNNs can detect contours, textures, and joint positions, which form the building blocks for temporal analysis in subsequent stages.

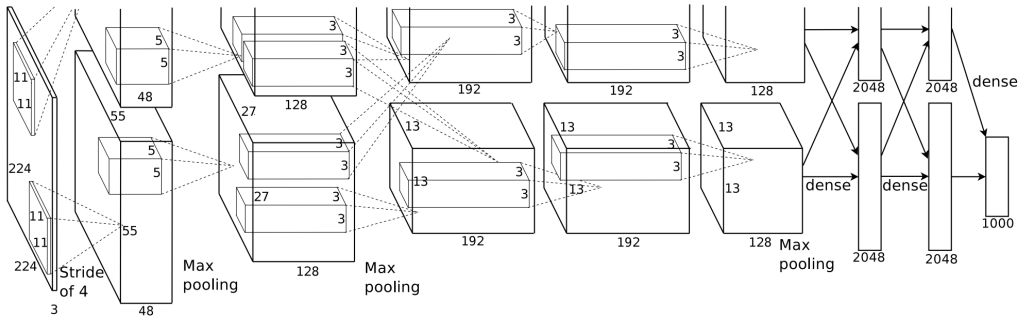

FIGURE 2.5: Illustration of a Convolutional Neural Network (CNN) architecture, showing convolutional filters, pooling, and fully connected layers (13).

2.8.3 Recurrent Neural Networks (RNNs)

While CNNs are excellent at extracting spatial features from static images, gesture recognition requires an understanding of how these features evolve over time. This is where Recurrent Neural Networks (RNNs) play an important role. RNNs are designed to handle sequential data by maintaining an internal memory (hidden state) that evolves with each new input in the sequence.

For an input sequence $\{x_t\}$, the hidden state at time t is updated as:

$$h_t = \sigma(W_h h_{t-1} + W_x x_t + b)$$

This recurrent formulation allows information from previous timesteps to influence the current prediction. However, vanilla RNNs suffer from vanishing and exploding gradients, making it difficult for them to capture long-term dependencies.

To overcome these limitations, advanced architectures such as Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs) were introduced. LSTMs incorporate memory cells and gating mechanisms (input, forget, and output gates) that regulate the flow of information. This allows them to preserve information over long sequences, making them particularly well-suited for modeling dynamic gestures where the order and timing of movements are critical.

For example, in recognizing a waving gesture, a CNN may identify the spatial configuration of the hand, but it is the LSTM that captures the repetitive left-right motion pattern over time. This ability to model temporal dependencies makes RNNs an essential component of video-based gesture recognition systems.

2.8.4 Graph Neural Networks (GNNs)

An emerging line of research in computer vision involves the use of Graph Neural Networks (GNNs). Unlike CNNs and RNNs, which operate on grid-structured or sequential data, GNNs are designed to process graph-structured data. This makes them particularly useful for skeleton-based gesture recognition, where the human body is naturally represented as a graph: joints correspond to nodes, and bones correspond to edges.

In a GNN, each node updates its representation by aggregating information from its neighbors. At layer k, the representation of a node v is updated as:

$$h_v^{(k)} = \sigma\left(\sum_{u \in \mathcal{N}(v)} W^{(k)} h_u^{(k-1)} + b^{(k)}\right)$$

where $\mathcal{N}(v)$ denotes the set of neighbors of v. This mechanism enables the network to capture both local interactions (e.g., movement of a single finger) and global dependencies (e.g., coordination of the entire arm).

A particularly influential architecture in this domain is the Spatial-Temporal Graph Convolutional Network (ST-GCN). ST-GCN extends the graph representation over time, treating the gesture sequence as a spatio-temporal graph. This allows the network to jointly model spatial body structure and temporal dynamics, making it highly effective for action and gesture recognition.

For gesture analysis, GNNs provide a structured and interpretable framework. Unlike CNNs that treat the input as unstructured pixels, GNNs exploit the inherent kinematic structure of the human body, leading to improved accuracy and robustness, especially in skeleton-based datasets.

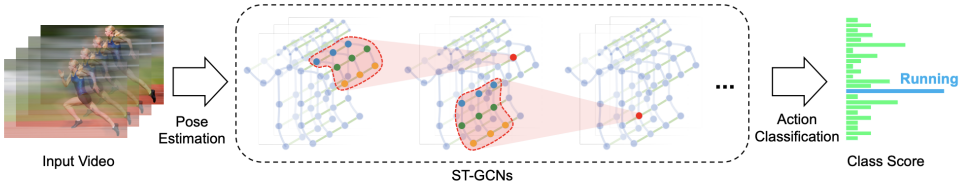

FIGURE 2.6: Skeleton-based gesture recognition using Spatial-Temporal Graph Convolutional Networks (ST-GCN). Joints are represented as nodes and bones as edges in a spatio-temporal graph (27).

2.8.5 Summary

Deep learning has transformed computer vision by providing powerful tools to learn hierarchical, temporal, and structured representations of data. CNNs extract spatial features, RNNs capture sequential dynamics, and GNNs model the structured dependencies of the human skeleton. Together, these architectures have established the foundation for state-of-the-art gesture recognition systems. However, they still face limitations in capturing very long-range dependencies and global contextual relationships, which motivates the recent adoption of transformer architectures in computer vision.

2.9 Vision Transformers

Convolutional Neural Networks (CNNs) have long dominated computer vision tasks due to their ability to capture local spatial patterns in images. However, CNNs are inherently limited by their local receptive fields and the difficulty of modeling long-range dependencies across the entire image. In contrast, transformers, originally proposed for natural language processing, are based on the *self-attention* mechanism, which directly models relationships between all elements in a sequence, regardless of their distance. This makes transformers particularly attractive for vision, where global context often plays a crucial role.

2.9.1 From NLP to Vision

The transformer architecture was introduced in the seminal work *"Attention is All You Need"* (26), where it achieved state-of-the-art results in machine translation. The key insight was to replace recurrence with self-attention, allowing parallel computation and better capture of long-range dependencies. Inspired by this success, researchers sought to adapt transformers to

images. The challenge, however, was that images are 2D grids of pixels, not 1D token sequences like words in a sentence.

The Vision Transformer (ViT) addressed this challenge by dividing an image into fixed-size patches, flattening them, and treating them as tokens in a sequence. This simple but powerful idea enabled the direct application of transformers to image recognition.

2.9.2 Patch Embeddings

The first step in ViT is to split an input image of size $H \times W \times C$ (height, width, channels) into non-overlapping patches of size $P \times P$. This results in $\frac{HW}{P^2}$ patches. Each patch is flattened into a vector and linearly projected into a D-dimensional embedding space:

$$z_0^i = \mathbf{E} \cdot \text{patch}_i + \mathbf{E}_{pos}^i$$

where $\mathbf{E}$ is the learnable projection matrix, $\mathbf{E}_{pos}^i$ is the positional encoding for patch i, and z_0^i is the input embedding to the transformer. A special `[CLS]` token is also prepended, whose final embedding is used for classification.

2.9.3 Self-Attention in ViT

Once the image is represented as a sequence of patch embeddings, the transformer encoder processes it using self-attention. For a set of embeddings $Z = \{z^1, z^2, \ldots, z^N\}$, queries ($Q$), keys ($K$), and values ($V$) are computed as linear projections:

$$Q = ZW_Q, \quad K = ZW_K, \quad V = ZW_V$$

The attention output is then:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

This operation allows each patch to attend to all other patches, capturing global relationships that CNNs would require many layers to approximate.

ViT employs *multi-head attention*, where the above process is repeated in parallel with different learned projections, enabling the model to learn diverse types of relationships.

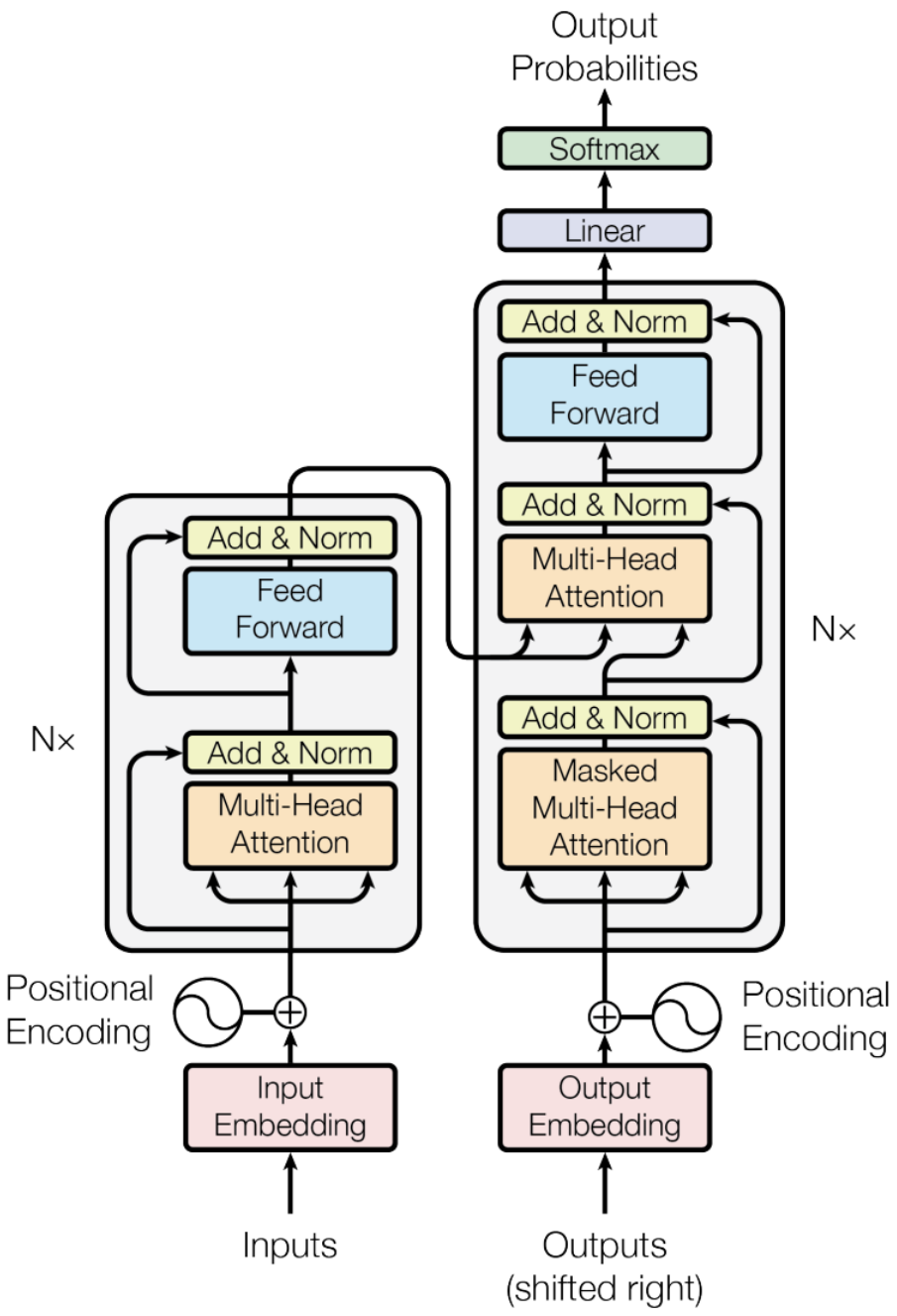

FIGURE 2.7: Vision Transformer (ViT) (26)

2.9.4 Transformer Encoder

The ViT encoder consists of stacked layers of multi-head self-attention and feed-forward networks, with residual connections and normalization. For layer l, the update can be written

as:

$$Z'_l = \text{MSA}(\text{LN}(Z_{l-1})) + Z_{l-1}$$

$$Z_l = \text{MLP}(\text{LN}(Z'_l)) + Z'_l$$

where MSA denotes multi-head self-attention, MLP is a feed-forward network, and LN is layer normalization.

After several layers, the `[CLS]` token embedding contains a global summary of the image, which is fed into a classification head.

2.9.5 Advantages of Vision Transformers

ViTs offer several advantages over CNNs:

- **Global context:** Self-attention enables direct modeling of relationships between any pair of patches, regardless of distance.
- **Scalability:** ViTs scale effectively with model size and dataset size, often outperforming CNNs when trained on large datasets.
- **Flexibility:** The same architecture can be applied across different modalities (e.g., images, videos, skeletons) by appropriately defining tokens.

However, ViTs also have challenges: they require large amounts of training data, are computationally expensive compared to CNNs, and lack inherent inductive biases such as translation equivariance. These drawbacks are often mitigated by hybrid approaches (CNN + transformer) or pretraining on massive datasets.

2.9.6 Relevance to Gesture Recognition

In gesture recognition, ViTs provide a natural way to capture both local details and global context. For example, a waving gesture involves both the local motion of the hand and its relationship to the arm and body. By attending across all patches, ViTs can integrate these cues more effectively than CNNs or RNNs alone. Furthermore, ViTs can be extended to spatio-temporal domains, enabling direct modeling of gestures in video sequences. This makes them a compelling theoretical foundation for the methods developed in this thesis.

2.10 Data Modalities in Computer Vision

Gesture recognition can be approached through various data modalities, each providing a different perspective on human motion. The choice of modality significantly impacts the robustness, computational cost, and applicability of recognition systems. In this section, we review the most common modalities used in computer vision for gesture analysis, with a particular emphasis on skeleton-based representations.

2.10.1 RGB Images and Videos

RGB images are the most basic and widely available modality, captured by standard cameras. They provide rich visual information including color, texture, and appearance cues. Gesture recognition systems using RGB data typically rely on convolutional neural networks (CNNs) to extract spatial features and recurrent models or transformers to capture temporal dynamics across frames. While RGB data is easy to obtain, it is sensitive to variations in illumination, viewpoint, and background clutter, which can degrade recognition accuracy.

2.10.2 Depth Data

Depth cameras, such as Microsoft Kinect or Intel RealSense, provide per-pixel distance information in addition to RGB values. Depth maps make it easier to segment the human body from the background and provide three-dimensional information about the scene. Depth data is particularly valuable in indoor environments where controlled lighting and short-range interactions are common. However, depth sensors have limitations such as sensitivity to sunlight, limited range, and increased hardware costs.

2.10.3 Optical Flow

Optical flow represents the apparent motion of pixels between consecutive frames. It captures motion dynamics independent of appearance and has been widely used for action and gesture recognition. Optical flow provides dense motion information but is computationally expensive to compute, and its accuracy degrades under fast motion or large displacements.

2.10.4 Skeleton-Based Representations

Skeleton-based representations abstract the human body into a set of keypoints corresponding to joints (e.g., elbows, wrists, knees). These joints are connected by edges to form a skeleton graph that encodes the kinematic structure of the body. Compared to raw RGB or depth data, skeletons offer a compact and high-level description of motion that is less affected by background noise, clothing, or lighting variations.

Skeleton data can be obtained using specialized depth cameras (e.g., Kinect), motion capture systems, or pose estimation algorithms such as OpenPose and MediaPipe. These systems use deep learning models to detect 2D or 3D joint positions from RGB or RGB-D inputs.

Formally, a skeleton at time t can be represented as a set of J joints:

$$S_t = \{(x_1^t, y_1^t, z_1^t), \ldots, (x_J^t, y_J^t, z_J^t)\}$$

where (x_j^t, y_j^t, z_j^t) are the coordinates of joint j. Over time, a gesture sequence forms a trajectory in this high-dimensional joint space.

2.10.5 Advantages of Skeleton Data

Skeleton-based representations offer several advantages for gesture recognition:

- **Compactness:** A skeleton requires far fewer data points than raw RGB frames, reducing computational overhead.
- **Robustness:** Skeletons are less sensitive to variations in clothing, lighting, and background.
- **Interpretability:** The representation directly encodes body joints and movements, making results easier to interpret.
- **Cross-domain generalization:** Skeletons abstract away appearance details, enabling better generalization across subjects and environments.

2.10.6 Challenges

Despite their benefits, skeleton-based approaches face challenges:

- **Estimation errors:** Pose estimation algorithms may fail under occlusion or fast motion, leading to noisy joint coordinates.
- **Loss of appearance cues:** Unlike RGB data, skeletons do not capture hand shape, facial expressions, or object interactions.
- **Sensor limitations:** Depth-based skeleton extraction (e.g., Kinect) requires additional hardware and may not function outdoors.

2.11 Summary

This chapter has provided the theoretical background necessary to understand the methods developed in this thesis. We began by reviewing the foundations of computer vision, including the principles of image formation, the pinhole camera model, and multi-view geometry. These concepts established how images are captured and how three-dimensional information can be reconstructed from two-dimensional projections. Such geometric reasoning continues to play a role in tasks such as calibration, depth estimation, and motion analysis.

We then transitioned to the era of deep learning, which has reshaped computer vision by enabling models to automatically learn features directly from raw data. Convolutional Neural Networks (CNNs) were introduced as powerful tools for spatial feature extraction, while Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks addressed the temporal dynamics inherent in video sequences. More recently, Graph Neural Networks (GNNs) have been applied to structured data such as skeletons, enabling spatio-temporal reasoning over joint positions and body movements.

Building upon this foundation, we examined the Vision Transformer (ViT), which represents a paradigm shift in visual representation learning. By treating images as sequences of patches

and applying self-attention, ViTs can capture both local details and long-range dependencies within an image. This capability makes them particularly relevant for gesture recognition, where understanding the coordination of body parts requires reasoning over both spatial and temporal dimensions. Although transformers are computationally demanding and require large datasets, their global context modeling and scalability provide a compelling advantage over traditional CNN- and RNN-based approaches.

We also reviewed the role of different data modalities in gesture recognition, including RGB, depth, optical flow, and skeleton-based representations. Among these, skeleton data was highlighted as particularly well-suited to the goals of this thesis due to its compactness, robustness to environmental factors, and interpretability. At the same time, challenges such as pose estimation errors and the loss of appearance cues were acknowledged. The combination of skeleton representations with advanced transformer-based models offers a promising direction for accurate and generalizable gesture recognition.

In conclusion, the evolution of computer vision from geometry-based models to deep learning and finally to transformers reflects the field's continuous push toward more expressive, robust, and scalable solutions. Each stage contributes unique strengths: geometric methods provide interpretability, deep learning enables feature learning, and transformers deliver global reasoning capabilities. The integration of these paradigms forms the theoretical foundation for the methods explored in the following chapters, where we apply transformer-based architectures to skeleton-based gesture recognition.

Chapter 3. Related Work

This chapter provides a comprehensive overview of the existing methods and literature related to the skeleton-based recognition and gesture analysis tasks. In this chapter, we first examine traditional approaches, followed by an exploration of various deep learning architectures. Finally, the chapter discusses available datasets and benchmarks, concluding with a summary of the current state of research.

3.1 Traditional Approaches

Skeleton-based hand gesture recognition initially focused on designing handcrafted features that capture the spatial and temporal patterns of skeletal joint movements. These methods rely on geometric and kinematic characteristics extracted from sequences of 3D joint coordinates. Such features include relative joint positions, inter-joint distances, joint angles, and velocities, which encode the dynamic shape and motion of the hand. De Smedt et al. (3) proposed a novel dynamic hand gesture recognition method using 3D skeleton data obtained from the Intel RealSense depth camera. They extracted shape-based features from connected joints and encoded them via Fisher Vectors processed with a temporal pyramid scheme, followed by classification with a linear Support Vector Machine (SVM).

However, traditional methods often suffer from limitations in generalizability across datasets due to their dependence on manually designed features tailored to specific gesture sets or environmental conditions. Moreover, these features fall short in capturing the complex spatio-temporal dependencies inherent in dynamic hand gestures with high intra-class variability (3), (21).

3.2 Deep Learning Based Approaches

The advent of deep learning has significantly advanced the capabilities of hand gesture recognition systems by enabling the automatic learning of complex features from raw skeleton data. Deep learning models can overcome many limitations of traditional handcrafted approaches. These models are typically categorized according to the primary network architecture used: Recurrent Neural Networks (RNN), Convolutional Neural Networks (CNN), Graph Convolutional Networks (GCN), and Transformer-based architectures.

3.2.1 CNN Based Approaches

Convolutional Neural Networks (CNNs) have been widely adopted in skeleton-based hand gesture recognition by transforming sequences of 3D joint coordinates into image-like representations suitable for convolutional processing. The core idea is to exploit CNNs' strength in capturing local spatial hierarchies while designing representations that encode both spatial joint relations and temporal dynamics.

A pioneering approach in this direction was Skeleton Optical Spectra (11), where skeleton sequences were mapped into texture-like images called Skeleton Optical Spectra. CNNs were then applied to extract discriminative spatio-temporal features for recognition. Although designed for full-body actions, this representation strategy inspired adaptations for hand gesture recognition. SkeleMotion (12) approach encodes skeleton sequences by calculating the orientation and magnitude of joint displacements, and represents them as RGB images for CNN processing. This method provided a more compact representation while preserving critical temporal information, achieving competitive results on large-scale gesture datasets.

To further enhance spatio-temporal modeling, Global Context-Aware CNN (28) that hierarchically aggregates local motion patterns into higher-level semantic features was proposed. This framework demonstrated the ability to learn co-occurrence patterns among joints, addressing one limitation of earlier CNN methods that primarily focused on local joint neighborhoods. In addition, hierarchical and multi-stream CNNs have been developed to leverage different aspects of gesture dynamics. For instance, a multi-stream CNN combining depth, optical flow, and skeleton modalities for dynamic hand gesture recognition in real-time was proposed (20).

Despite these successes, CNN-based approaches face challenges such as potential loss of fine-grained structural details during skeleton-to-image conversion and limited ability to model non-local joint correlations. Nonetheless, their efficiency and strong performance on several gesture benchmarks have made CNNs a key stepping stone toward more advanced architectures.

3.2.2 RNN Based Methods

Recurrent Neural Networks (RNNs) have been widely applied in skeleton-based recognition tasks due to their natural ability to model temporal dependencies across sequential data.

Early applications of RNNs in gesture recognition adopted standard architectures to learn temporal transitions between frames. However, traditional RNNs often suffered from vanishing or exploding gradients when modeling long sequences. To address this, gated architectures such as Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRU) became popular.

In the domain of hand gestures, a real-time deep GRU network for dynamic hand gesture recognition from 3D skeletal data (18) was proposed. This approach introduced an attention

mechanism to highlight the most informative joints and frames, achieving state-of-the-art results on benchmark datasets while maintaining efficiency for real-time applications. Hybrid RNN architectures have also been explored to improve spatial modeling. For example, a spatio-temporal LSTM network that explicitly models joint co-occurrences by combining spatial attention with temporal recurrence was proposed (15).

Other variants, such as attention-based recurrent relational networks, extend RNNs by integrating graph-like reasoning into recurrent modules, allowing them to better capture the interplay between different joints across time (23). These models bridge the gap between purely sequential learning and structural modeling.

3.2.3 GCN Based Approaches

Graph Convolutional Networks (GCNs) have become a prominent choice for skeleton-based gesture recognition due to their ability to model the human hand as a graph structure, where joints are represented as nodes and bones as edges. Unlike CNNs or RNNs that treat skeleton sequences as grids or vectors, GCNs exploit the natural topology of skeleton data, allowing more effective representation of spatial dependencies and temporal evolution.

One of the earliest and most influential works in this direction is the Spatial-Temporal Graph Convolutional Network (ST-GCN) (27), which explicitly encoded both spatial and temporal connections of skeleton joints and demonstrated state-of-the-art performance in skeleton-based action tasks, inspiring adaptations for hand gesture recognition. Building on this, 2s-AGCN model (14) introduced graph streams capturing actional (motion-related) and structural (physical connectivity) relationships. By dynamically updating graph connections during training, this method improved robustness to diverse execution styles of gestures.

To address multi-scale motion patterns, MS-G3D (16) approach utilized multi-scale graph convolutions and temporal kernels to capture fine- and coarse-grained dependencies across joints and frames, making it highly relevant for recognizing both subtle finger movements and large-scale hand trajectories. Another relevant advancement was Shift-Based Graph Convolutional Network (2), where a lightweight shift-based operation replaced standard convolutions to reduce computation while maintaining competitive accuracy. Such efficiency-oriented designs are particularly useful for real-time gesture interfaces.

3.2.4 Transformer Based Approaches

Transformer architectures (26) have recently gained popularity in skeleton-based gesture recognition due to their powerful self-attention mechanisms that capture long-range dependencies in sequence data. Unlike CNNs and RNNs, Transformers model global context by dynamically weighting relationships between all joints across temporal frames, which is beneficial for understanding complex gesture dynamics.

An important foundation for embedding-based comparison approaches was established by Schroff et al. in FaceNet (22), where a triplet-loss formulation was introduced to learn discriminative embeddings for face recognition and clustering. This idea has been widely adopted beyond facial analysis, including gesture comparison, as it enforces that samples of the same class (anchor–positive) remain close in embedding space while samples of different classes (anchor–negative) are pushed apart.

In the context of gesture recognition, Transformer-based methods leverage this embedding principle while modeling spatio-temporal dynamics of skeleton sequences through self-attention. For example, 3D-Jointsformer (31) is a hybrid model that combines a 3D-CNN with Transformer self-attention for skeleton-based hand gesture recognition. This model first extracts local spatio-temporal embeddings from skeletal joint sequences using a 3D-CNN and then uses a Transformer to model long-range temporal dependencies; it achieved high accuracy on Briareo and Multimodal Hand Gesture datasets while running in real time.

Recently, a lightweight Spatio-Temporal Skeleton Attention Transformer (ST-SAT) (9) tailored for long-distance gesture recognition in UAV control was published. This model emphasizes efficiency by selectively attending to salient skeletal joints across time, enabling robust recognition even in resource-constrained aerial platforms.

Despite their promising results, Transformer models often require larger datasets and computational resources. Nevertheless, their ability to learn global semantic dependencies makes them a compelling direction for skeleton-based gesture classification and comparison tasks.

3.3 Datasets and Benchmarks

The availability of well-annotated datasets plays a pivotal role in advancing research on skeleton-based hand gesture recognition. High-quality datasets provide standardized benchmarks, allowing fair comparison across methods and promoting reproducibility. Hand gesture recognition datasets typically vary in terms of scale, sensor modality (RGB, depth, skeleton), number of gesture classes, and recording conditions. However, challenges such as viewpoint variation, intra-class variability, finger occlusion, and noise in skeleton extraction remain significant factors that influence model performance.

These are the most widely used skeleton-based hand gesture datasets: the Dynamic Hand Gesture (DHG-14/28) dataset (4), the SHREC dataset from the 3D Shape Retrieval Contest (5), the Briareo dataset (19), and the First-Person Hand Action Benchmark (FPHAB) dataset (7).

Despite the availability of several benchmark datasets, skeleton-based gesture recognition continues to face challenges such as limited subject diversity, small-scale training sets compared to vision datasets, and restricted environmental variability. Recent works have emphasized the

need for larger, multimodal datasets and improved evaluation protocols to foster generalization to real-world Human–Computer Interaction applications. Our work contributes to this space by leveraging the challenging FPHAB dataset to evaluate both gesture classification and gesture comparison tasks.

3.4 Summary

The literature on skeleton-based hand gesture recognition demonstrates a clear evolution from handcrafted features to deep learning architectures. Traditional approaches provided important foundations but lacked generalizability across datasets due to their reliance on manually engineered descriptors. CNN-based methods improved spatial modeling by transforming skeleton sequences into image-like representations, but they often discarded fine-grained joint dependencies during this conversion. RNN-based approaches introduced the ability to model temporal dynamics directly, yet they struggled with effectively capturing spatial correlations between joints. GCNs addressed this limitation by exploiting the natural graph structure of skeletal data, achieving strong performance but often at the expense of computational efficiency. More recently, Transformer-based architectures have shown promise in capturing global spatio-temporal dependencies, though they require large-scale datasets and significant training resources.

While most existing works have focused on skeleton-based action recognition/classification, very few have explored the problem of gesture *comparison*, which is crucial for applications like gesture-based authentication, retrieval, and similarity search. Furthermore, many methods emphasize either spatial or temporal modeling, but not both in a unified and efficient manner. Another gap in the literature is the limited exploration of embedding-based formulations (e.g., triplet loss) that can directly optimize similarity in the feature space, despite their success in other domains such as face recognition (22).

Chapter 4. Methodology and Implementation

This chapter outlines the methodology followed in this work, including the design of the model architecture, dataset selection, experimental setup, and evaluation metrics adopted to assess the performance of both classification and comparison tasks.

4.1 Model Architecture

The architecture we adopted in this work is based on PoseFormer (30), a Spatial-Temporal Transformer-based archietcture developed for 3D human pose estimation. PoseFormer was designed to take as input 2D skeleton sequences extracted from videos and predict corresponding 3D joint positions. Its strength lies in the way it combines spatial encoding of joint relationships with temporal modeling of motion dynamics, making it a natural candidate for gesture recognition tasks.

While the original PoseFormer outputs a sequence of 3D poses, our goal was to adapt the same principles for two different tasks: gesture classification and gesture comparison. To achieve this, we retained the spatial and temporal transformer modules but redefined the input representation and redesigned the output layers to suit our tasks. In particular, the input layer was redefined to process sliding windows of skeleton sequences specific to hand gestures, and the output layer was redesigned to produce either class probabilities (for classification) or embeddings (for comparison). The following subsections describe each component of the model architecture in detail.

4.1.1 Spatial Transformer Module

The first stage of the architecture is the spatial transformer module, which is responsible for modeling the relationships among the joints of the hand in a single frame. In the First-Person Hand Action Benchmark (FPHAB) dataset we used, each frame of the skeleton sequence consists of 21 joints. Each joint is represented by its 3D coordinates (x, y, z), resulting in a structured skeleton representation per frame.

In the PoseFormer architecture, 2D joint coordinates were projected into a higher-dimensional embedding space through a learnable linear mapping. We applied the same idea but with 3D joint inputs instead of 2D, as our dataset provides full (x, y, z) skeleton data. Once embedded, position encodings are added to preserve the order of joints, and the sequence of joint embeddings is processed using multi-head self-attention layers.

Functionally, this module allows the network to learn how joints interact within a frame. For instance, the relative positioning of the thumb and index finger is critical in distinguishing

a squeeze gesture from an flip gesture. Similarly, the alignment of the palm with the wrist provides important cues. By leveraging self-attention, the model does not rely on predefined joint connectivity but instead dynamically learns which joints influence each other, leading to a more flexible representation of intra-frame structure.

4.1.2 Temporal Transformer Module

After obtaining a spatial embedding for each frame, the sequence of frame-level embeddings is passed to the temporal transformer module. This stage captures dependencies across time, which is essential since gestures are defined not only by static hand poses but by their motion dynamics.

In the PoseFormer architecture, temporal modeling was used to aggregate information from multiple frames to estimate poses at a central frame. In our use case, the temporal module instead encodes the full motion trajectory of a gesture. Each frame embedding is augmented with a temporal positional encoding to retain ordering information, and multi-head self-attention is applied across frames.

This enables the model to simultaneously capture local motion patterns (e.g., small finger movements between adjacent frames) and long-range dependencies (e.g., the difference between an upward and downward wave of the hand). The temporal transformer effectively integrates spatially rich frame embeddings into a coherent temporal representation, ensuring that the final output reflects both static joint configurations and dynamic motion context.

4.1.3 Input and Output Layers

While the spatial and temporal transformer modules provide the backbone of the architecture, the way in which data enters the model and how outputs are produced differs depending on the task. In this work, we designed two task-specific adaptations: one for gesture classification and one for gesture comparison.

4.1.3.1 Gesture Classification Model

For the classification task, we adapted PoseFormer to take sequences of skeleton data and output probabilities over gesture classes. Each input sample corresponds to a sliding window of T consecutive frames, with every frame containing 21 joints represented by their (x, y, z) coordinates. This results in an input tensor of size $b \times T \times 21 \times 3$, where b denotes the batch size.

For instance, when the sliding window size is set to 101, the model observes 50 frames before and 50 frames after the frame of interest, ensuring that both past and future motion context are included. This provides the network with richer temporal dependencies, which are crucial for distinguishing gestures that may look similar when viewed over only a few frames.

The output layer is adapted to match the classification task by producing one score for each of the 45 gesture categories in the dataset. Thus, the output tensor has the shape $b \times 45$. A softmax activation is applied to transform these raw scores into class probabilities, ensuring that the model assigns each input sequence to exactly one class. The network is trained using the cross-entropy loss function, which penalizes the difference between predicted probabilities and ground-truth class labels.

4.1.3.2 Gesture Comparison Model

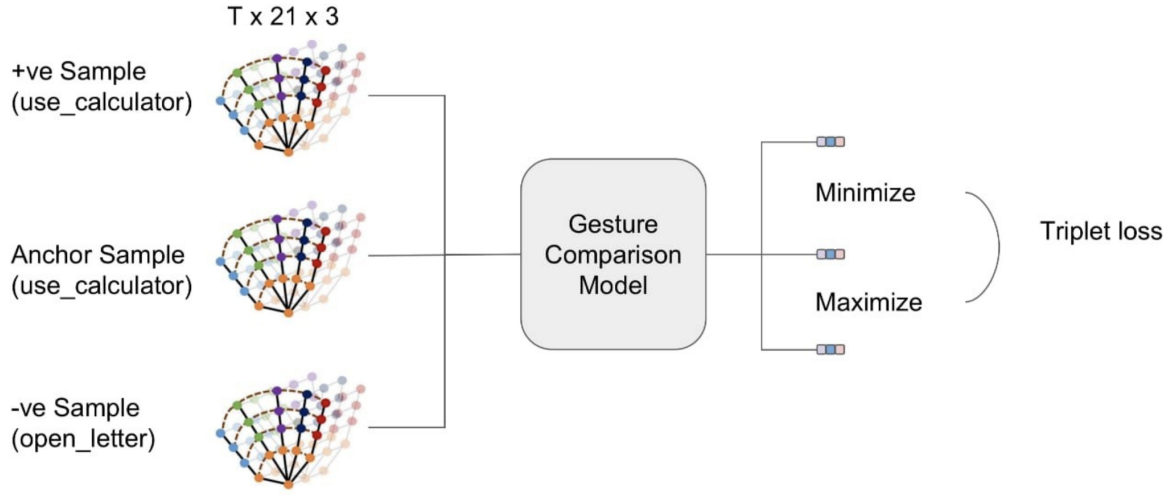

FIGURE 4.1: Approach for training gesture comparison model.

After validating the feasibility of the architecture for classification, we extended it to gesture comparison, where the goal is to measure similarity between two gesture instances. Instead of producing class scores, the model learns to generate embeddings that capture the semantic characteristics of each gesture.

In this setting, the input is structured as triplets: an anchor, a positive sample (belonging to the same gesture class), and a negative sample (from a different class). Each of these is represented as a tensor of size $T \times 21 \times 3$, and together the triplet is fed into the network with shape $b \times 3 \times T \times 21 \times 3$, where b denotes the batch size and the dimension "3" corresponds to the three elements of the triplet.

At the output, instead of class probabilities, the model generates embeddings of fixed size d for each sequence. This results in an output tensor of size $b \times 3 \times d$. These embeddings are trained using the triplet loss function (Figure 4.2), which ensures that the distance between anchor and positive embeddings is minimized, while the distance between anchor and negative embeddings is maximized by at least a margin value. Formally, the triplet loss encourages

the model to construct an embedding space in which similar gestures are close together and dissimilar gestures are farther apart.

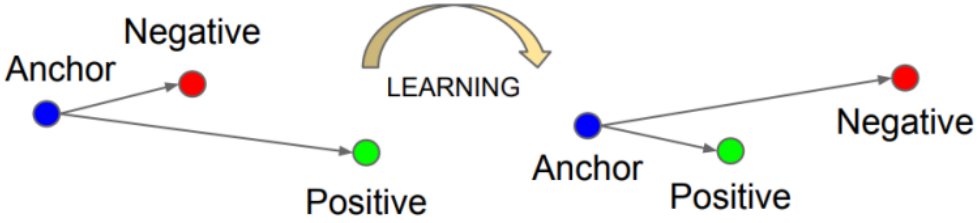

FIGURE 4.2: Triplet loss function (22).

This design allows the architecture to handle not only explicit classification but also more flexible tasks such as gesture verification and retrieval, where the goal is to compare actions directly rather than predict discrete labels.

4.2 Dataset

For this work, we used the First-Person Hand Action Benchmark (FPHAB) dataset (7), which has become a widely used benchmark for evaluating skeleton-based hand gesture recognition models. The dataset was specifically designed to capture natural hand-object interactions from a first-person perspective, making it highly suitable for gesture recognition tasks where subtle finger movements and temporal dynamics play an important role.

The FPHAB dataset contains a total of 1,175 video sequences spanning 45 daily hand actions, such as wave, pinch, flip pages, open wallet, or squeeze sponge. These actions were recorded in three different environments: kitchen, office, and social to provide contextual variety. Six subjects performed the actions multiple times, introducing natural intra-class variation in hand size, execution style, and motion speed. In total, the dataset provides 105,459 annotated frames, making it large and diverse enough for both training and evaluation.

A distinctive feature of FPHAB is its multimodal nature: each video sequence includes synchronized RGB, depth, and 3D skeleton annotations. For our experiments, we relied exclusively on the skeleton data, which offers a compact and privacy-preserving representation of gestures while avoiding the visual redundancy of RGB streams. The skeleton annotations were obtained using a motion capture system, where six magnetic sensors combined with inverse kinematics produced the 3D coordinates of 21 joints per frame. Each joint is represented by its (x, y, z) position, enabling a precise description of hand pose and movement.

The taxonomy of hand-object interactions in the dataset is illustrated in Figure 4.3. This taxonomy highlights the richness of the dataset, ranging from simple free-hand gestures to

complex object manipulations. By covering this broad spectrum of gestures, the FPHAB dataset serves as an ideal testbed for skeleton-based gesture analysis.

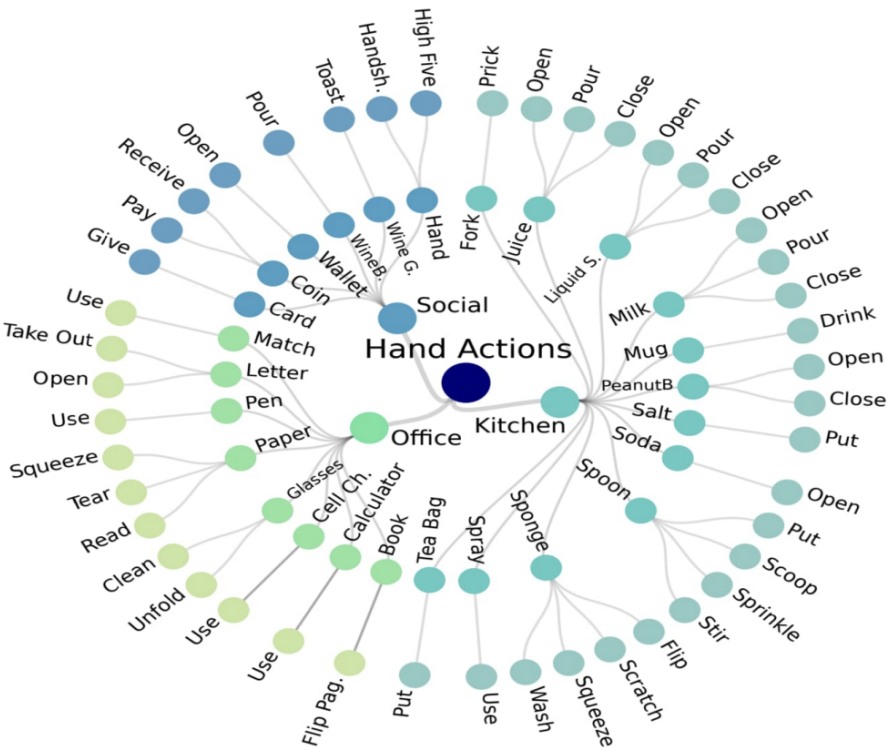

FIGURE 4.3: Taxonomy of actions in the First-Person Hand Action Benchmark dataset (7).

Each sequence is stored in the dataset as a directory containing a `skeleton.txt` file, which records the frame-by-frame joint coordinates. Each line in the file corresponds to a single frame, starting with the frame index followed by 63 floating-point values (21 joints $\times$ 3 coordinates). An excerpt of such a file is shown in Figure 4.4. This format provides a structured, frame-level representation of gestures that can be directly processed by our models.

skeleton.txt

```
0000 48.086000 56.429400 352.093200 43.159500 50.276400 338.435200 54.447500 -21.562800
368.728900 58.865100 -11.065300 387.551400 61.585500 3.028200 400.000200 64.388800 13.037800
412.321000 18.534000 12.126000 372.524200 2.569800 -12.606300 394.623600 -2.370000 -30.931000
415.210200 25.162500 -28.971000 402.525900 3.982900 -34.328700 426.968700 -12.082700 -33.710900
438.864700 37.354900 -33.362800 425.513600 12.643700 -16.857900 429.630000 7.597600 2.121600
415.834400 50.486100 -18.097200 436.966700 25.262500 -10.659400 446.581500 5.115800 3.696000
442.971100 55.011400 1.488100 441.195500 49.295000 -5.552400 458.797100 29.011400 -7.050800
467.183400
0001 46.467100 56.452000 350.930300 41.377200 50.406900 337.284000 51.938800 -21.669000
367.275800 56.607400 -11.290300 386.103600 59.584900 2.726600 398.580100 62.592500 12.659200
410.915000 16.509600 11.908800 370.747900 0.400800 -13.029700 392.425300 -4.652400 -31.338500
412.998500 22.296700 -29.382600 400.521100 0.913000 -34.947100 424.504000 -15.131600 -34.448600
436.433800 34.863100 -33.718900 423.854700 10.188800 -17.208000 428.164100 5.201400 1.912600
414.543300 47.659000 -18.800100 435.054400 22.368300 -11.783200 444.808400 2.286700 2.656500
441.171600 52.038000 0.284800 439.437100 45.489100 -7.393100 457.134300 25.043200 -8.706200
465.149400
0002 44.595800 56.224500 349.229800 39.219100 50.435500 335.582400 48.866200 -22.172000
364.583900 53.968000 -12.104700 383.468300 57.361800 1.706200 396.067500 60.710000 11.434900
408.476900 13.974800 12.213700 368.836600 -2.320400 -12.458600 390.302200 -7.933300 -31.192700
410.340300 19.131000 -29.855600 397.754200 -2.320200 -35.398600 421.683500 -18.103500 -35.264300
433.966300 31.626400 -34.596900 420.830800 7.154500 -17.850000 425.377100 2.203900 1.583100
412.192000 45.555900 -19.657500 432.676400 20.416500 -12.238400 442.523800 0.432100 2.390100
439.113700 49.568500 -1.052600 436.912200 42.601900 -8.860900 454.692400 22.107600 -10.006800
462.608800
0003 42.852700 56.584500 347.243200 37.523700 51.073000 333.462900 46.476600 -22.096500
361.248900 51.527000 -12.397000 380.338600 54.936000 1.167300 393.198900 58.272000 10.653800
405.797600 11.586500 13.479800 367.148700 -5.216600 -10.874500 388.971700 -11.541600 -30.094800
408.325700 16.437200 -29.618100 394.642600 -5.380200 -35.080900 418.896200 -20.850800 -35.282700
431.569900 29.129900 -34.890100 417.667300 4.933400 -18.016200 423.126000 0.104800 1.720300
410.352800 42.947400 -20.359000 429.652900 18.038700 -12.822700 439.985200 -1.755600 2.144900
```

FIGURE 4.4: Example of a skeleton file in the dataset. Each line corresponds to a frame index followed by (x, y, z) coordinates of 21 hand joints.

4.3 Experimental setup

Before training and evaluating our models, the raw skeleton annotations from the dataset had to be preprocessed into a form suitable for our models. The FPHAB dataset provides skeleton sequences in plain text files, where each line corresponds to a single frame. The line begins with a frame index, followed by 63 floating-point values that represent the (x, y, z) coordinates of 21 hand joints.

Each frame of the dataset contains the 3D coordinates of 21 annotated joints, including the wrist, palm center, and finger bones (proximal, intermediate, and distal joints). This provides a structured but compact description of the hand pose. By stacking consecutive frames, we obtain a spatio-temporal skeleton sequence.

For example, a single frame entry in the dataset looks as follows:

```
0000 48.086000 56.429400 352.093200 43.159500 50.276400 338.435200 ...
```

Here, `0000` is the frame index, and (48.086000 56.429400 352.093200) represent the (x, y, z) coordinates of the 1st joint and (43.159500 50.276400 338.435200) represent the (x, y, z) coordinates of the 2nd joint.

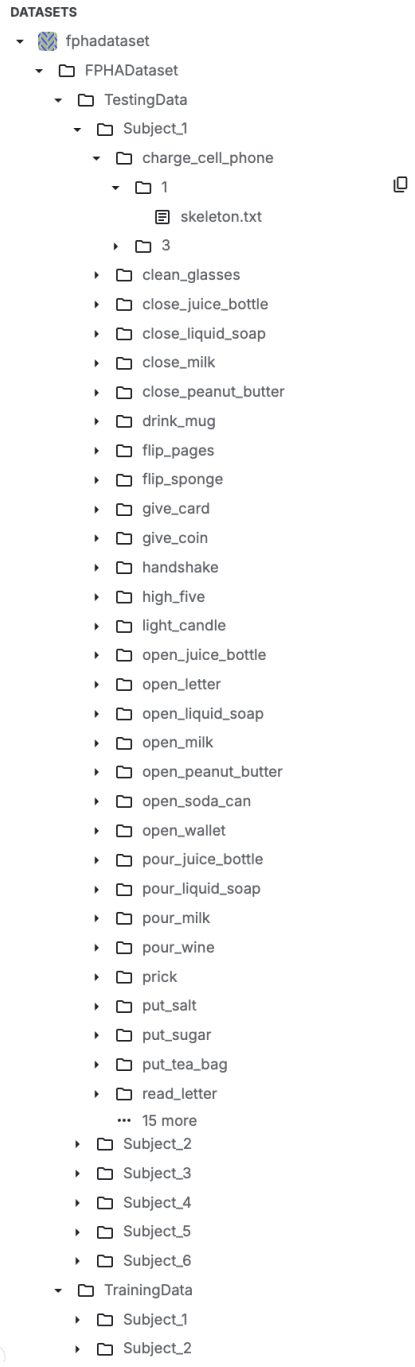

FIGURE 4.5: Directory structure of the dataset.

An illustrative code excerpt for reading and processing skeleton files is shown below:

LISTING 4.1: Training loop for CNN model

```
for epoch in range(10):
    optimizer.zero_grad()
    outputs = model(inputs)
    loss = criterion(outputs, targets)
    loss.backward()
```

```
    optimizer.step()
```

The authors also defined an official split with 600 training and 575 testing instances, which we followed to ensure comparability with prior work. The dataset is organized into directories: TrainingData and TestingData (Figure 4.5). A `skeleton.txt` file is illustrated in Figure 4.4. For our preprocessing, we wrote Python scripts that parsed the skeleton files, segmented them into sliding windows, and stored them in separate directories for training and testing.

After preprocessing, the prepared skeleton sequences were used to train and evaluate both classification and comparison models. The next section describes the evaluation metrics employed to assess their performance.

We used PyTorch deep learning framework for implementation and executed on a NVIDIA Tesla P100 GPU available through Kaggle's computing platform. We trained the models with the Adam optimizer, using a learning rate of 0.01. The training process was carried out with a batch size of 32 and for a total of 10 epochs, which we found sufficient for both classification and comparison models to converge without overfitting.

4.4 Evaluation Metrics

To assess the effectiveness of our models, we employed both quantitative and qualitative evaluation metric. Since our work covers two distinct tasks—gesture classification and gesture comparison, the evaluation criteria were tailored accordingly.

4.4.1 Quantitative Evaluation using Accuracy

For both classification and comparison tasks, accuracy was used as the quantitative metric. Accuracy measures the ratio of correctly predicted instances to the total number of instances, and is formally defined as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

where:

- **TP (True Positive)**: The number of gesture instances correctly identified as belonging to their true class.
- **TN (True Negative)**: The number of gesture instances correctly rejected as not belonging to a class.
- **FP (False Positive)**: The number of gesture instances incorrectly assigned to a class they do not belong to.

- **FN (False Negative)**: The number of gesture instances incorrectly rejected from their true class.

For the classification task, accuracy was directly computed from the softmax predictions by comparing the predicted gesture class with the ground-truth label.

For the comparison task, accuracy was determined by first computing the distances between embeddings of anchor-positive and anchor-negative pairs. A threshold value was then applied to distinguish between "similar" and "dissimilar" gestures. If the anchor-positive distance was below the threshold and the anchor-negative distance was above it, the prediction was considered correct. The evaluation rules are summarized in Table 4.1.

TABLE 4.1: Illustration of how true positives, false positives, true negatives, and false negatives were determined using threshold for gesture comparison.

Evaluation Outcome	**Distance Comparison Rule**
True Positive	Anchor-Positive Distance $\leq$ Threshold
True Negative	Anchor-Negative Distance $>$ Threshold
False Negative	Anchor-Positive Distance $>$ Threshold
False Positive	Anchor-Negative Distance $\leq$ Threshold

One of the challenges in gesture comparison is defining the decision threshold for similarity. To address this, we generated histograms of anchor-positive and anchor-negative distances for different embedding dimensions and distance metrics. The threshold was selected at the point of minimal overlap between the two distributions. This ensured that both false positives and false negatives were minimized simultaneously, leading to a balanced and reliable evaluation metric.

4.4.2 Qualitative Evaluation with t-SNE

In addition to quantitative metrics, we conducted a qualitative evaluation using t-distributed Stochastic Neighbor Embedding (t-SNE). This method reduces high-dimensional embeddings into two dimensions while preserving neighborhood relationships, making it possible to visually inspect the structure of the learned embedding space.

t-SNE allowed us to confirm whether the embeddings learned by the comparison model formed distinct clusters for each of the 45 gesture classes in the dataset. Well-separated clusters indicate strong intra-class compactness and inter-class separability, whereas overlapping clusters reveal ambiguities between similar gestures. The t-SNE plots therefore serve as an important qualitative complement to numerical accuracy values.

Chapter 5. Results and Analysis

This chapter presents the experimental results of the proposed skeleton-based gesture analysis models. We evaluate both the classification model and the gesture comparison model, followed by an examination of the learned embeddings through t-SNE visualization. Special attention is given to how different parameters such as sliding window size, embedding dimension, margin value, and distance metric influence model performance. For the comparison task, histograms are analyzed to determine the decision threshold, which is essential for computing accuracy from true positive, true negative, false positive, and false negative counts.

5.1 Experiments on Gesture Classification

The first set of experiments was performed using the gesture classification model. Here, the primary variable under study was the size of the temporal sliding window. The sliding window determines how many consecutive frames are fed into the model, thus controlling the temporal context available for recognizing each gesture.

TABLE 5.1: Gesture Classification model accuracy across different sliding window sizes

Sliding Window Size	**Accuracy (%)**
21	62.67
41	65.82
61	67.90
81	67.98
101	68.49

As shown in Table 5.1, performance improves steadily as the sliding window grows from 21 to 61 frames, reflecting the importance of temporal information in gesture classification. With a window size of 101 frames, the model achieved its highest accuracy of 68.49%. However, the increase beyond 61 frames yields only marginal improvements, suggesting that beyond a certain point, additional frames do not provide significant new information and may even increase computational cost unnecessarily. This indicates the existence of an optimal temporal context that balances information richness with efficiency.

5.2 Experiments on Gesture Comparison

The second set of experiments investigated the comparison model, which was trained with triplet loss. The purpose was to evaluate how well the model can differentiate between two gestures by

embedding them in a feature space where similar gestures are close and dissimilar ones are far apart. Evaluation of this model required not only analyzing accuracy values but also studying the distribution of distances to determine suitable thresholds for decision-making.

5.2.1 Effect of Margin Values

The margin in the triplet loss function enforces a minimum distance between anchor-positive and anchor-negative pairs. Table 5.2 summarizes the impact of different margin values combined with varying embedding dimensions and distance metrics.

TABLE 5.2: Gesture Comparison model accuracy for different margin values and embedding dimensions (D- Dimension) and distance metrics (E- Euclidean, C-Cosine)

Margin	**64D (E)**	**64D (C)**	**128D (E)**	**128D (C)**
0.2	77.44	76.90	76.60	75.53
0.6	72.43	74.59	74.59	73.45
1.0	76.56	74.90	76.84	76.14

The results indicate that a smaller margin of 0.2 provides the best performance. This setting offers enough flexibility for the model to tolerate natural intra-class variations while still separating different classes. Larger margins (0.6 or 1.0) enforce stricter separation, which reduces the model's ability to handle small differences within the same gesture class, ultimately leading to lower accuracy.

5.2.2 Effect of Embedding Dimensions

The embedding dimension controls the representational capacity of the model. To examine this, we tested embedding sizes of 64, 128, 256, and 512, both using Euclidean distance metric and Cosine distance metric. Table 5.3 reports the accuracy values, while Figures 5.1, 5.2, 5.3, and 5.4 show histogram distributions used for threshold calculation.

TABLE 5.3: Gesture Comparison model accuracy across different embedding dimensions

Embedding Dimension	**Euclidean (%)**	**Cosine (%)**
64	77.44	76.90
128	76.60	75.53
256	74.96	73.21
512	73.43	67.86

To determine thresholds, histograms of anchor-positive and anchor-negative distances were generated. The threshold was selected at the point of minimum overlap between the two distributions, ensuring a balance between false positives and false negatives. For 64D embeddings, the overlap between distributions is relatively small, allowing for a clear threshold and higher accuracy. In contrast, 512D embeddings show significant overlap, which increases misclassifications and reduces performance.

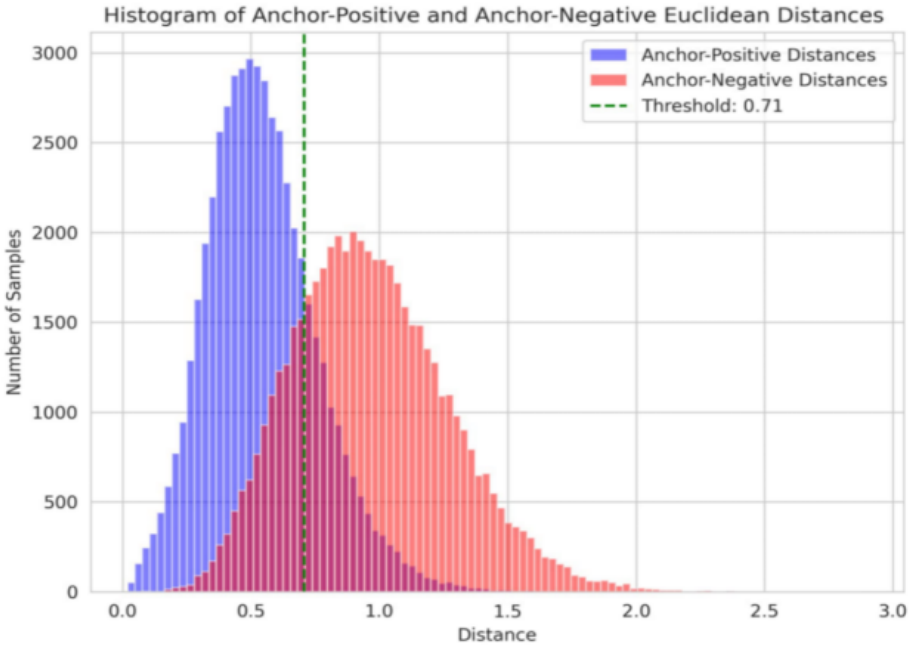

FIGURE 5.1: Histogram of anchor-positive and anchor-negative distances for embeddings of 64D using Euclidean metric.

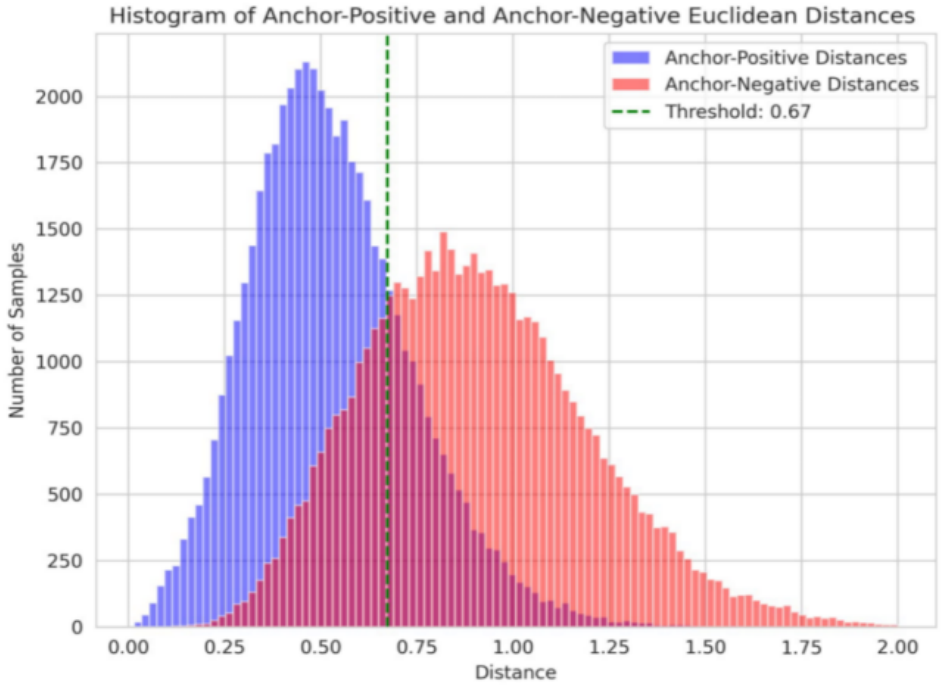

FIGURE 5.2: Histogram of anchor-positive and anchor-negative distances for embeddings of 512D using Euclidean metric.

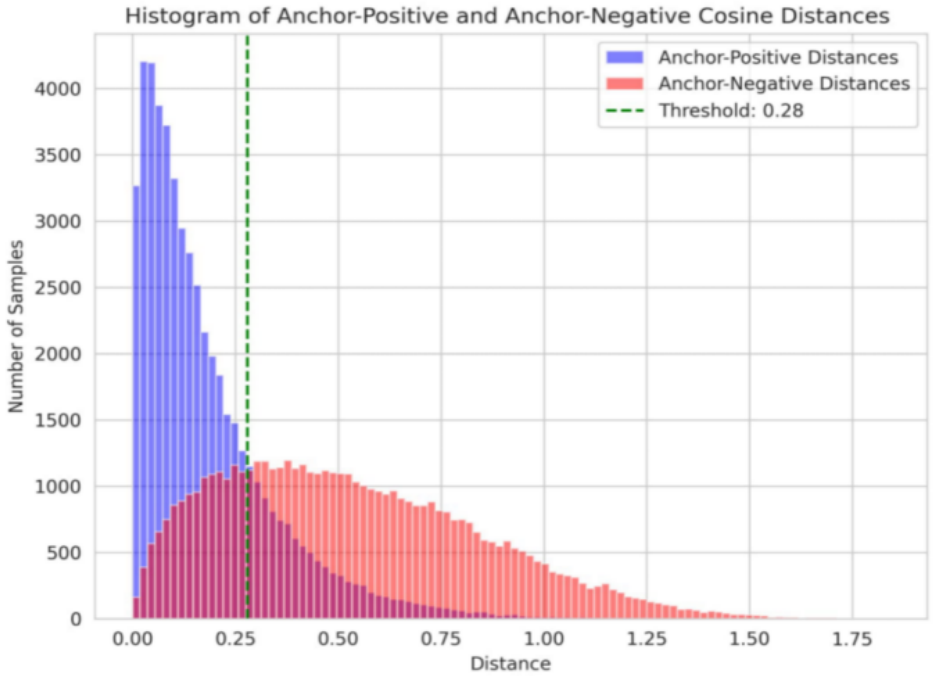

FIGURE 5.3: Histogram of anchor-positive and anchor-negative distances for embeddings of 64D using Cosine metric.

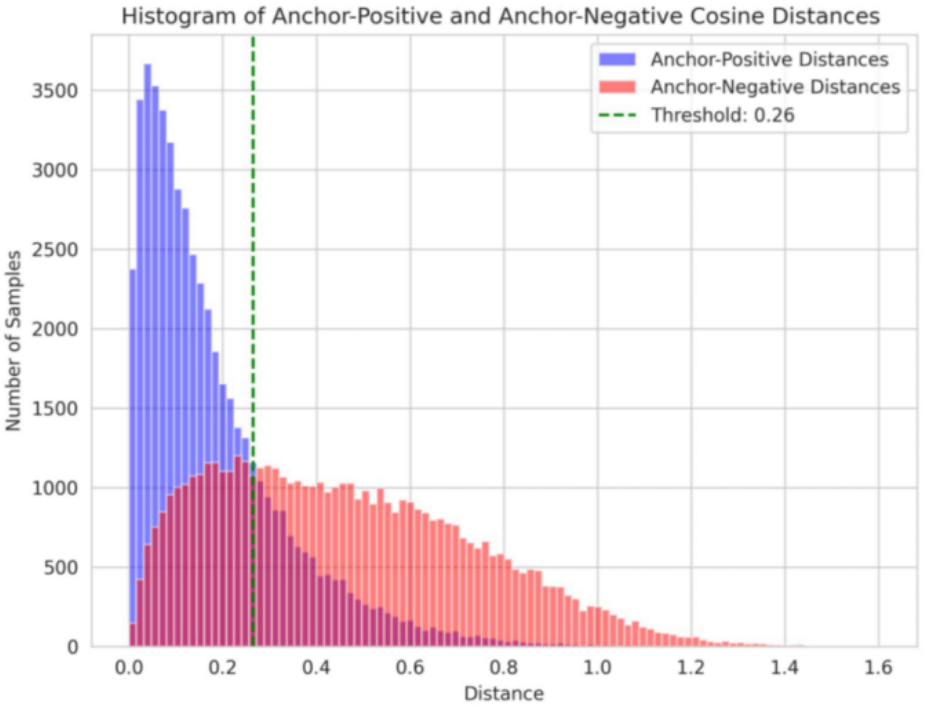

FIGURE 5.4: Histogram of anchor-positive and anchor-negative distances for embeddings of 512D using Cosine metric.

Overall, the results demonstrate that smaller embedding dimensions (64D) not only simplify the representation but also generalize better, whereas larger dimensions tend to overfit and create ambiguities in the embedding space.

5.2.3 Effect of Sliding Window Size

In this experiment, the sliding window size was varied while keeping the margin fixed at 0.2 and the embedding dimension at 64, and Euclidean as distance metric. The results are given in Table 5.4, with histograms shown in Figures 5.5 and 5.6.

From Table 5.4, it is evident that larger sliding windows consistently improve the accuracy of the comparison model. The model achieves only 72.27% accuracy at 21 frames, indicating that the temporal context is insufficient to distinguish between gestures that may look similar in short segments. Increasing the window to 61 frames results in a large jump in accuracy (77.44%), highlighting the benefit of capturing longer motion dependencies. At 101 frames, the model reaches its peak performance of 78.02%, although the improvement compared to 81 frames is marginal. This plateau suggests that there exists a saturation point where adding more frames contributes little to the decision boundary.

TABLE 5.4: Comparison accuracy across different sliding window sizes

Sliding Window Size	**Accuracy (%)**
21	72.27
41	74.07
61	77.44
81	77.87
101	78.02

To evaluate this behavior in more detail, histograms of anchor-positive and anchor-negative distances were generated for both small and large window sizes. Figures 5.5 and 5.6 shows the distributions for window sizes of 21 and 101.

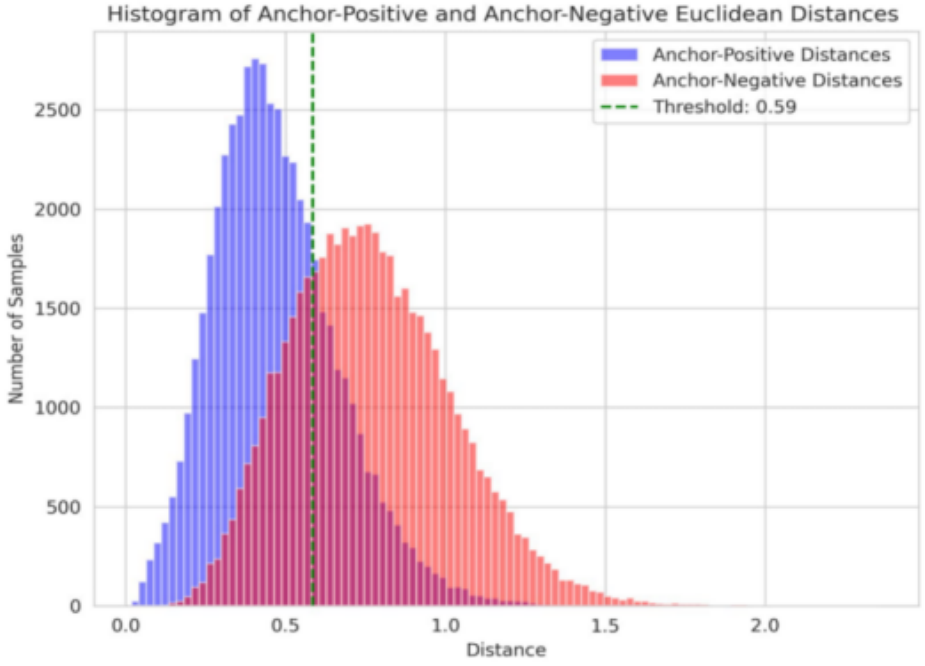

FIGURE 5.5: Histogram of anchor-positive and anchor-negative distances for sliding window of 21 frames using Euclidean distance.

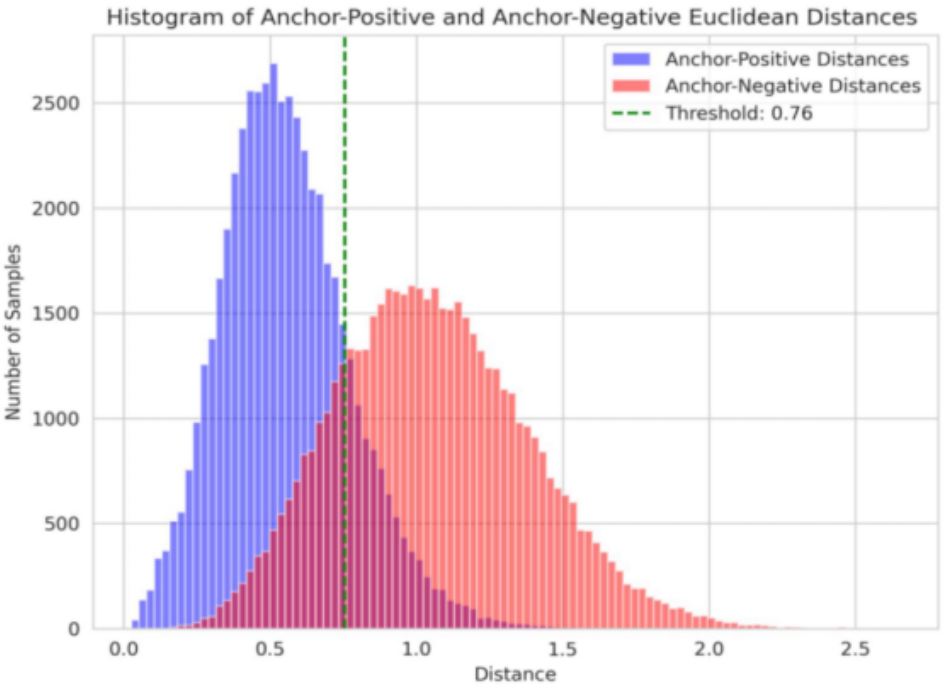

FIGURE 5.6: Histogram of anchor-positive and anchor-negative distances for sliding window of 101 frames using Euclidean distance.

For the smaller window, there is significant overlap between the two histograms, meaning that the threshold cannot cleanly separate the classes, leading to frequent misclassifications. For the larger window size, the overlap region shrinks considerably, making it easier to find a threshold that minimizes both false positives and false negatives. This directly explains the higher accuracy at larger window sizes.

The results also have practical implications. Shorter windows are computationally cheaper and may be preferred in real-time systems with limited processing resources. However, the significant drop in accuracy makes them unsuitable for tasks requiring high reliability, such as medical rehabilitation or biometric authentication. Larger windows, while more accurate, increase the computational cost and latency. Therefore, selecting an appropriate window size should be guided by the target application:

- For real-time gesture recognition where speed is critical, medium-sized windows (around 41–61 frames) offer a good compromise between efficiency and accuracy.
- For offline analysis or applications where accuracy is paramount, larger windows (81–101 frames) are preferable, since they capture nearly complete gesture dynamics.

The experiments confirm that temporal context plays a decisive role in skeleton-based gesture analysis tasks. Increasing the sliding window improves discriminative ability by reducing

histogram overlap and producing cleaner thresholds. However, the marginal gains beyond 81 frames highlight the importance of balancing accuracy against computational cost.

5.3 t-SNE Visualization

To complement the quantitative evaluation, we conducted a qualitative analysis of the learned embedding space using t-distributed Stochastic Neighbor Embedding (t-SNE). This technique projects high-dimensional embeddings into a two-dimensional plane while preserving local neighborhood relationships, making it well-suited for visualizing the structure of the feature space.

The First-Person Hand Action dataset used in our experiments contains 45 different gesture classes. To visualize the embeddings, we assigned a unique color to each of the 45 classes and plotted the anchor embeddings from the test set after applying t-SNE dimensionality reduction. Figure 5.7 shows the plot.

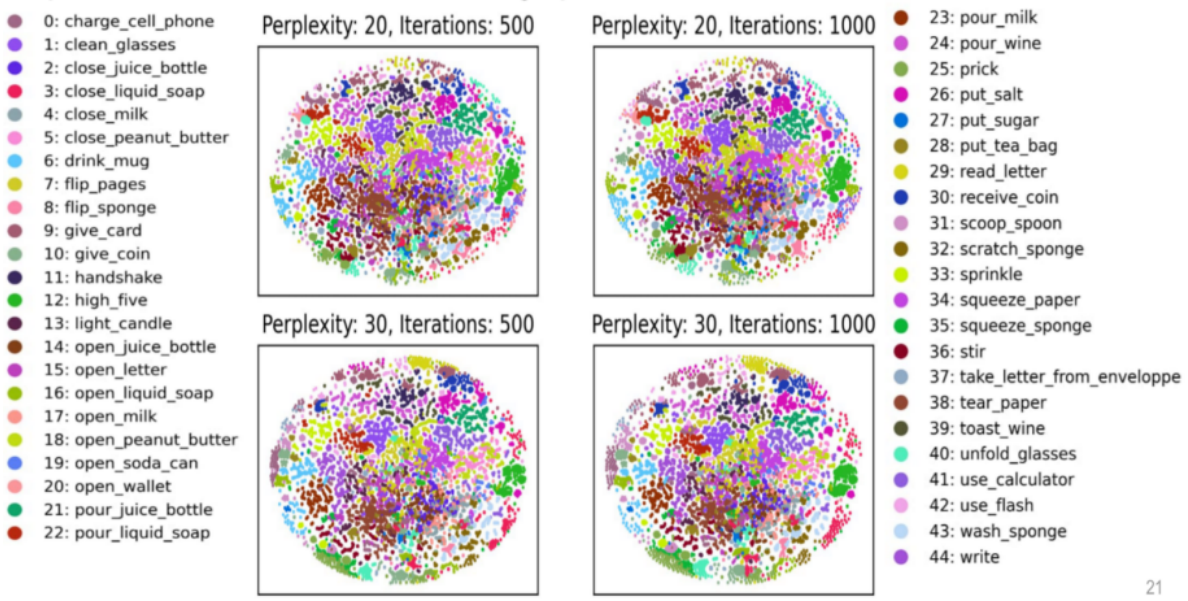

FIGURE 5.7: t-SNE visualization of anchor embeddings of gesture comparison model from the test data.

The visualization reveals several important observations:

- **Cluster Formation:** Samples from the same gesture class are grouped together into well-defined clusters. This indicates that the model successfully captures the discriminative features of different gestures and maps them into separable regions of the embedding space.
- **Separation Between Classes:** Different gesture clusters are clearly separated from each other, demonstrating that the learned embeddings not only capture intra-class similarity but also maximize inter-class separability. This separation is crucial for reliable gesture comparison.

- **Class Diversity:** Since there are 45 distinct classes, some clusters are tightly packed while others are more spread out. Compact clusters correspond to gestures that are highly consistent across different actors and scenarios, whereas broader clusters often belong to gestures with larger natural variability (e.g., different styles of grasping).
- **Overlap Between Certain Gestures:** A few classes that are visually or kinematically similar show partial overlap. For example, gestures involving similar hand-object interactions may produce embeddings that lie close to each other. This overlap explains occasional misclassifications observed in the quantitative results.

The use of 45 different colors to represent the 45 gesture classes makes it visually clear how the model distinguishes between the actions. Gestures such as flip_pages or squeeze_sponge form their own distinct clusters, while gestures with subtle motion differences (e.g., put_salt vs. put_sugar) tend to cluster close together but remain separable.

In summary, the t-SNE visualization provides strong qualitative evidence that the embedding space learned by the transformer-based model is meaningful and structured. The model organizes gestures into coherent clusters that reflect the 45 action categories of the dataset, validating the effectiveness of the learned embeddings beyond numerical accuracy values.

Chapter 6. Conclusion

In this thesis, we investigated skeleton-based hand gesture analysis for both classification and comparison tasks. Our work went beyond reporting numerical accuracy and focused on understanding how design choices such as temporal context, embedding dimensionality, margin values, and distance metrics influence the performance of the models.

Through our experiments, we showed that gesture analysis does not solely depend on building more complex architectures, but also on carefully tuning parameters and evaluation strategies. We found that compact embeddings, when combined with sufficient temporal context, create a feature space that is both discriminative and generalizable. This highlights the importance of striking a balance between simplicity and expressiveness rather than always pursuing larger or deeper models.

We further showed that skeleton-based representations can serve as a practical and privacy-preserving alternative to raw image or video-based approaches. By relying on skeletal motion data, we can capture the essence of human gestures without exposing sensitive visual details, making this approach highly suitable for applications in sensitive or resource-constrained environments.

Finally, we demonstrated that treating gesture analysis as a comparison problem, in addition to classification, provides a richer understanding of human motion. While our experiments were limited to a single dataset, the ability to measure similarity between gestures points toward potential applications in areas such as adaptive user interfaces, rehabilitation monitoring, and biometric authentication.

Chapter 7. Future Work

Although our work has shown promising results, there are several ways in which we can extend and improve this research. A natural step forward is to test the model on additional datasets that capture a wider variety of gestures, actors, and recording conditions. This would allow us to evaluate the generalization ability of our approach and confirm its reliability in diverse scenarios.

Another important direction is to optimize the framework for real-time deployment. While larger sliding windows improve accuracy, they also introduce latency. Reducing this overhead, or designing adaptive mechanisms that adjust the window size depending on gesture complexity, could make the system suitable for interactive applications such as sign language recognition or human-computer interaction.

We also see opportunities in multimodal integration. Combining skeleton-based features with RGB or depth data may improve robustness in cases where skeleton extraction is noisy. Furthermore, alternative similarity learning strategies, such as contrastive or quadruplet loss, could enhance the discriminative quality of the learned embeddings.

Finally, we can adapt our framework to domain-specific applications such as medical rehabilitation, assistive technologies, or biometric authentication, where reliable gesture comparison can provide real-world benefits.

Appendix I: Source Code

Code is available at the following GitHub repository:

https://github.com/AMOGHSG1/Master_Thesis

Appendix II: Tools Used for Thesis Writing

For formatting this thesis, I used LaTeX on the Overleaf platform (Ove). To support the writing and refinement process, I also made use of AI tools such as ChatGPT-5, Perplexity, and Google Gemini, in addition to traditional literature sources and academic references.

Bibliography

[Ove] Overleaf, online latex editor. https://www.overleaf.com/.

[2] Cheng, K., Zhang, Y., He, X., Chen, W., Cheng, J., and Lu, H. (2020). Skeleton-based action recognition with shift graph convolutional network. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 183–192.

[3] De Smedt, Q., Wannous, H., and Vandeborre, J.-P. (2016a). Skeleton-based dynamic hand gesture recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 1–9.

[4] De Smedt, Q., Wannous, H., and Vandeborre, J.-P. (2016b). Skeleton-based dynamic hand gesture recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

[5] De Smedt, Q., Wannous, H., Vandeborre, J.-P., Guerry, J., Le Saux, B., and Filliat, D. (2017). Shrec'17 track: 3d hand gesture recognition using a depth and skeletal dataset. In *3DOR-10th Eurographics workshop on 3D object retrieval*, pages 1–6.

[6] Eigen, D., Puhrsch, C., and Fergus, R. (2014). Depth map prediction from a single image using a multi-scale deep network. *Advances in neural information processing systems*, 27.

[7] Garcia-Hernando, G., Yuan, S., Baek, S., and Kim, T.-K. (2018). First-person hand action benchmark with rgb-d videos and 3d hand pose annotations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 409–419.

[8] Gerum, R. C., Richter, S., Winterl, A., Mark, C., Fabry, B., Le Bohec, C., and Zitterbart, D. P. (2019). Cameratransform: A python package for perspective corrections and image mapping. *SoftwareX*, 10:100333.

[9] Guo, X., Hu, Y., Wang, Y., Zhu, Q., and Mo, Y. (2025). A lightweight spatio-temporal skeleton attention transformer for long-distance gesture recognition in uav control. *IEEE Transactions on Industrial Electronics*.

[10] Hartley, R. and Zisserman, A. (2003). *Multiple view geometry in computer vision*. Cambridge university press.

[11] Hou, Y., Li, Z., Wang, P., and Li, W. (2016). Skeleton optical spectra-based action recognition using convolutional neural networks. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(3):807–811.

[12] Ke, Q., Bennamoun, M., An, S., Sohel, F., and Boussaid, F. (2017). A new representation of skeleton sequences for 3d action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3288–3297.

[13] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

[14] Li, M., Chen, S., Chen, X., Zhang, Y., Wang, Y., and Tian, Q. (2019). Actional-structural graph convolutional networks for skeleton-based action recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3595–3603.

[15] Liu, J., Shahroudy, A., Xu, D., and Wang, G. (2016). Spatio-temporal lstm with trust gates for 3d human action recognition. In *European conference on computer vision*, pages 816–833. Springer.

[16] Liu, Z., Zhang, H., Chen, Z., Wang, Z., and Ouyang, W. (2020). Disentangling and unifying graph convolutions for skeleton-based action recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 143–152.

[17] Longuet-Higgins, H. C. (1981). A computer algorithm for reconstructing a scene from two projections. *Nature*, 293(5828):133–135.

[18] Maghoumi, M. and LaViola Jr, J. J. (2019). Deepgru: Deep gesture recognition utility. In *International symposium on visual computing*, pages 16–31. Springer.

[19] Manganaro, F., Pini, S., Borghi, G., Vezzani, R., and Cucchiara, R. (2019). Hand gestures for the human-car interaction: The briareo dataset. In *International Conference on Image Analysis and Processing*, pages 560–571. Springer.

[20] Molchanov, P., Yang, X., Gupta, S., Kim, K., Tyree, S., and Kautz, J. (2016). Online detection and classification of dynamic hand gestures with recurrent 3d convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4207–4215.

[21] Oudah, M., Al-Naji, A., and Chahl, J. (2020). Hand gesture recognition based on computer vision: a review of techniques. *journal of Imaging*, 6(8):73.

[22] Schroff, F., Kalenichenko, D., and Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823.

[23] Si, C., Chen, W., Wang, W., Wang, L., and Tan, T. (2019). An attention enhanced graph convolutional lstm network for skeleton-based action recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1227–1236.

[24] Szeliski, R. (2022). *Computer vision: algorithms and applications*. Springer Nature.

[25] Triggs, B., McLauchlan, P. F., Hartley, R. I., and Fitzgibbon, A. W. (1999). Bundle adjustment—a modern synthesis. In *International workshop on vision algorithms*, pages 298–372. Springer.

[26] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

[27] Yan, S., Xiong, Y., and Lin, D. (2018). Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

[28] Zhang, X., Xu, C., and Tao, D. (2020). Context aware graph convolution for skeleton-based action recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14333–14342.

[29] Zhang, Z. (2002). A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22(11):1330–1334.

[30] Zheng, C., Zhu, S., Mendieta, M., Yang, T., Chen, C., and Ding, Z. (2021). 3d human pose estimation with spatial and temporal transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 11656–11665.

[31] Zhong, E., Del-Blanco, C. R., Berjón, D., Jaureguizar, F., and García, N. (2023). Real-time monocular skeleton-based hand gesture recognition using 3d-jointsformer. *Sensors*, 23(16):7066.