UNIVERSITÄT ZU LÜBECK
INSTITUTE OF MEDICAL INFORMATICS

From the Institute of Medical Informatics
of the University of Lübeck
Director: Prof. Dr. rer. nat. habil. Heinz Handels

Geometric Deep Learning for Hand Skeleton Tracking and Gesture Recognition on 3D Point Clouds

Dissertation
for
Fulfillment of Requirements for the Doctoral Degree
of the University of Lübeck

from the Department of Computer Sciences and Technical Engineering

Submitted by
Niklas Hermes
from Minden

Lübeck, 2024

First referee: Prof. Dr. Mattias Heinrich
Second referee: Prof. Dr.-Ing. Erhardt Barth

Date of oral examination: 30.10.2024

Approved for printing. Lübeck, 07.11.2024

Abstract

The automated machine interpretation of hand poses and the spatial modeling of the hand hold promising potential for a touchless form of interaction between humans and machines. For decades, new methods have been developed and new sensors designed to enable this intuitive form of communication. Due to the groundbreaking success of deep learning techniques, the focus in recent years has primarily shifted towards image processing methods, with particular emphasis on image modalities with a regular grid structure. The hand as an anatomically highly complex organ, proves to be a challenging structure to model due to the numerous dependencies between the muscles and ligament apparatus. In this context, three-dimensional geometric domains offer a natural representation of objects whose processing is increasingly gaining significance through recent advances in the field of geometric deep learning.

This thesis focuses on exploring new possibilities arising from processing irregular data structures such as point clouds and graphs using learning-based methods in the field of hand analysis and interpretation of human gestures. Various techniques and methods are developed and presented in this context, generally originating from the spectrum of geometric deep learning methods and covering a variety of applications in the field of hand analysis. Learning on graphs is investigated for extracting local spatial features as well as to analyze the global context of a given structure, enabling the localization of keypoints. In this context, it is also essential to sensibly design the information flow between two semantically different structures by employing a suitable strategy to connect the nodes within the graph. Another crucial aspect of this thesis is the incorporation of temporal information in the form of a motion analysis into the estimation of hand poses. A new technique to optimize methods for scene flow estimation specifically for analyzing the movement of the hand skeleton is presented. This area is complemented by introducing a novel deep learning approach that analyzes the motion information of a hand to refine the per-frame prediction of a hand pose. Finally, it is investigated to what extent gesture recognition can benefit from the extracted hand information using the presented techniques for hand analysis on point clouds.

The methodological components of this work highlight the potential benefits of analyzing three-dimensional geometric domains through specialized deep learning techniques. In this context, the rich spatial information of these modalities has proven to be advantageous regarding the challenges associated with the hand. As a wide spectrum of tasks has been covered in this work, the integration of the presented

techniques in many areas is conceivable. Especially, the contributions provide significant value in applications for gesture-based human-machine interaction.

Zusammenfassung

Die automatisierte maschinelle Interpretation einer Handpose sowie die räumliche Modellierung der Hand besitzen ein viel versprechendes Potential für eine berührungslose Form der Interaktion zwischen Mensch und Maschine. Seit Jahrzenten werden neue Verfahren entwickelt und neue Sensoren entworfen, um diese intuitive Form der Kommunikation zu ermöglichen. Durch den bahnbrechenden Erfolg von Deep Learning-Verfahren hat sich der Fokus in den letzten Jahren vor allem auf bildverarbeitende Methoden gerichtet und insbesondere Bildmodalitäten mit regelmäßiger Gitterstruktur wurden intensiv erforscht. Die Hand als anatomisch äußerst komplexes Greiforgan manifestiert sich durch die vielen Abhängigkeiten der Muskel- und Bandapparate im Allgemeinen als anspruchsvoll zu modellierende Struktur. In diesem Kontext stellen dreidimensionale geometrische Domänen aufgrund der natürlichen Darstellung von Objekten eine an Potenzial reiche Datenstruktur dar, deren Verarbeitung durch jüngste Fortschritte im Bereich des Geometric Deep Learning zunehmend an Bedeutung gewinnt.

Im Fokus dieser Arbeit steht die Untersuchung neuer Möglichkeiten, die sich durch die Verarbeitung irregulärer Datenstrukturen wie Punktwolken und Graphen anhand lernender Verfahren im Bereich der Handanalyse und Interpretation menschlicher Gestik ergeben. In diesem Zusammenhang werden verschiedene Techniken und Methoden entwickelt und vorgestellt, die im Allgemeinen dem Spektrum der Geometric Deep Learning-Verfahren entspringen und eine Vielzahl von Anwendungen im Feld der Handanalyse abdecken. Das Lernen auf Graphen wird sowohl für die Extraktion lokaler räumlicher Merkmale untersucht, als auch um den globalen Kontext einer vorliegenden Struktur zu analysieren und so die Lokalisation von Keypoints zu ermöglichen. In diesem Kontext gilt es ebenfalls den Informationsfluss zwischen zwei semantisch unterschiedlichen Strukturen, durch eine geeignete Strategie zur Verknüpfung der Knoten innerhalb des Graphen, sinnvoll zu gestalten. Ein weiterer wichtiger Teil dieser Arbeit umfasst die Einbindung zeitlicher Informationen in Form einer Bewegungsanalyse in die Schätzung von Handposen. Hierzu wird eine neue Technik zur Optimierung von Verfahren zur Bestimmung des Szenenflusses für die spezifische Analyse der Bewegung des Handskeletts entwickelt. Dieser Bereich wird durch die Vorstellung eines neuen Deep Learning-Verfahrens vervollständigt, welches die Bewegungsinformationen einer Hand analysiert, um die statische Schätzung einer Handpose zu präzisieren. Abschließend wird untersucht, inwieweit die Erkennung von Gesten von den extrahierten Handinformationen durch die vorgestellten Verfahren zur Handanalyse auf Punktwolken profitieren kann.

Durch die methodischen Anteile dieser Arbeit können die potentiellen Vorteile der Analyse dreimensionaler geometrischer Domänen durch spezielle Deep Learning-Verfahren herausgestellt werden. Dabei konnten besonders die mit der Hand verknüpften Problematiken von den reichen räumlichen Informationen der Modalitäten profitieren. Da ein breites Spektrum von Aufgaben innerhalb dieser Arbeit abgdeckt wurde, ist die Ergänzung vieler Bereiche durch die vorgestellten Techniken denkbar, insbesondere aber liefern die entwickelten Methoden bei Anwendungen für die gestenbasierte Mensch-Maschine-Interaktion einen klaren Mehrwert.

Contents

Chapter 1

Introduction

1.1 Motivation

The hand is one of the most crucial organs for human interaction with the environment. In addition to tactile perception and object manipulation, the hand grants humans the ability to execute gestures, enabling a form of non-verbal communication. From early childhood, gestures play a pivotal role in our lives, especially in expressing and interpreting emotions and feelings [Lhommet et al., 2014]. For decades, active research has been conducted to make this form of communication usable for human-machine interaction. Furthermore, with the rapid technological advancement, its significance continues to rise. In recent years, research has been particularly focused on achieving an intuitive human-machine interaction, especially in the Virtual Reality (VR) and Augmented Reality (AR) fields [Li et al., 2019b; Reifinger et al., 2007], as well as in the automotive sector [Ohn-Bar et al., 2014], continuously evolving and advancing.

Early methods often relied on wearable devices such as specially designed gloves [Wang et al., 2009] or other external sensors attached to the hand. Through these devices, the spatial position of significant points on the hand can be determined, allowing inferences about the user's gestures. Regardless of the precision of these methods, attaching devices to the hand always imposes limitations on the mobility of users. Consequently, especially with the groundbreaking success of deep learning-based approaches, there has been increasing research on image-based solutions for spatiotemporal hand structure recognition. However, all methods applied in hand pose estimation face significant challenges that need to be addressed.

Firstly, the anatomical complexity of the hand must be emphasized as one of the major challenges. The simple act of extending or flexing a finger results from the interplay of numerous joints, ligaments, and muscle structures. This allows for precise control over individual fingertips. The complex ligament structure in the hand creates numerous dependencies between different parts, meaning the movement of one finger consistently affects the movement possibilities of all other fingers. Furthermore, there is significant interindividual variability in the hand. Muscle groups and ligaments differ among individuals and some people even lack certain structures. This, in turn, leads to variations in the possible range of motion of the fingers depending on the individual

anatomy, complicating the creation of a generalized model. Another challenge is the occlusion of important hand parts in everyday movements. In numerous gestures or interactions with objects, parts of the hand remain invisible to an observer or camera, making vital information not directly observable and requiring contextual or temporal inference.

Especially, methods based on two-dimensional data struggle to adequately overcome these challenges. This can be observed, for example, in the generation of art using artificial intelligence. Figure 1.1 shows the results of the Dezgo text-to-image tool [Stable Diffusion AI, 2023] with the input *woman / man holding an apple*. In both images, the background, face, hair, clothing, and even the apple appear realistic and plausible. However, the hand of the depicted woman exhibits artifacts and thus appears unrealistic. In the image on the right, although no artifacts are present, the left hand posture of the person seems implausible, particularly the little finger in front of the apple and the remaining fingers behind it. This example highlights that reducing image data to a two-dimensional projection in combination with the seemingly infinite possibilities of hand poses presents a significant problem. Therefore, it can be beneficial to conduct the analysis of the hand in three-dimensional space.

For several years, systems for capturing 3D data have been continually evolving. Among the most widely used methods are stereo vision (SV) and time-of-flight (ToF) sensors. The former typically consists of two perspective cameras that employ the triangulation principle to determine the distance of individual points. ToF cameras emit near-infrared light that is reflected by objects and returned to the sensor. The distance at each pixel can be calculated based on the determined phase of the incoming light. Both systems generate a depth map, which contains the distance from the sensor to surfaces in the captured scene for each pixel. Using the intrinsic camera information, a 3D coordinate in space can be calculated for each pixel, resulting in a set of 3D points, known as a *point cloud*. As the spatial characteristics of the captured scene are represented, this data structure generally includes dense and sparse regions, making it irregular.

In recent developments in the field of machine learning, there has been notable progress in methods specifically designed to handle irregular structures. Geometric Deep Learning [Bronstein et al., 2021] focuses particularly on learning in higher-dimensional domains, such as manifolds, graphs and point clouds. The underlying concept often involves processing the local neighborhood of an element of the input structure through a permutation-invariant function to extract local information. When this function is applied to each point in the cloud or each node of the graph in their respective neighborhoods, it yields a permutation-equivariant function over the entire structure. This approach allows extracted information to be associated with structures independently of the order of points or nodes. Notable examples of such methods are PointNet++ [Qi et al., 2017] for point clouds and Dynamic Graph CNN [Wang et al.,

(a) Woman holding an apple.

(b) Man holding an apple.

Fig. 1.1: Two images are depicted, generated using the text-to-image tool developed by Dezgo [Stable Diffusion AI, 2023] with the assistance of artificial intelligence. The respective captions display the text input that produced the image. Both graphics show a person holding an apple, with the artificial intelligence generating plausible representations of the background, face, hair, clothing, and even the apple. The only unrealistic aspects are the hands in each image. In the left image, there are noticeable artifacts, such as a missing part of a finger and another finger not connected to the hand. In the right image, the anatomy of the left hand of the person is unrealistic with the little finger positioned in front of the apple while the remaining fingers are placed behind it.

2018] for graphs as underlying domain. A more comprehensive introduction to this area of learning is provided in Sec. 2.3 of this thesis.

The field of geometric learning is less actively explored than the processing of two-dimensional information, leading to a significantly lower density of knowledge. The necessary consideration of hand-related challenges further narrows down existing insights, with many areas remaining underexplored in this context. This thesis aims to investigate the extent to which the spatiotemporal analysis of the hand can benefit from utilizing three-dimensional irregular structures and the associated information. Additionally, new methods will be designed and evaluated to overcome the described challenges associated with hand-related research.

1.2 Objectives

The primary objective of this work is to overcome the aforementioned challenges associated with hand-related problems. This is to be achieved by leveraging information embedded in 3D data within novel Geometric Deep Learning approaches, facilitating a

thorough analysis of the hand and generating valuable information for touchless human-machine interaction systems. This fundamental objective can be further specified and divided into the following three sub-areas: 1.) Hand pose estimation using 3D point cloud data. 2.) Extraction of temporal motion information from 3D point cloud sequences and integrating it into hand pose estimation. 3.) Gesture recognition based on sequences of predicted hand poses. The following provides a detailed summary of all three sub-objectives.

Hand pose estimation using single 3D point clouds. The first objective of this thesis is to investigate the potential of geometric 3D data in predicting hand poses. For this purpose, solutions based on geometric deep learning to determine the 3D positions of keypoints on the hand using a 3D point cloud will be developed and comprehensively analyzed. Since conventional devices for generating spatial data, such as stereo vision and time-of-flight sensors, generally produce depth data in a 2D grid, a first necessary step is to directly translate this type of data into a 3D point cloud using the camera intrinsic parameters. This ensures to be as independent as possible from camera-specific information, focusing primarily on the geometric characteristics of the generated data. Furthermore, only static single-frame information will be considered at this stage, knowing that the challenge of overcoming occlusion issues in parts of the hand is difficult to overcome this way. However, this ensures that this initial objective specifically targets the complexity of hand anatomy.

Suitable geometric deep learning-based solutions examine a given point cloud for its local features. However, to simulate the numerous dependencies in hand anatomy, it is additionally necessary to contextualize the local information within a global framework. As another aim, considering the overall application, an intuitive human-machine communication interface is desired. Consequently, this necessitates a fast and direct analysis of an incoming hand, emphasizing solutions with limited computational complexity to consistently ensure real-time capability of a potential application.

Integration of temporal information. As mentioned in the previous section, occlusions of hand parts constitute a challenging problem, especially when using a single sensor and the associated dependency on the camera view. Therefore, the second objective of this work involves the analysis of temporal information in sequences of 3D point clouds, with a focus on two main areas. Firstly, it is essential to extract important information associated to a sequence of point clouds, as well as incorporating them into the hand analysis. Specifically, the goal is to comprehensively investigate motion information represented as scene flow and particularly evaluate its application in hand analysis.

The second area involves the integration of extracted motion information into hand pose estimation to compensate for the lack of information due to occluded parts of the hand. This includes optimizing scene flow methods specifically for hand movement and integrating this information into a network for improved hand pose estimation by analyzing a sequence of point clouds.

In summary, the first two objectives aim to explore the potential of 3D point clouds in generating and appropriately utilizing spatial and temporal information for hand pose estimation.

Gesture Recognition. After exploring new methodological improvements for hand pose estimation in the previous objectives, the third objective of this thesis is to focus on the application in human-machine interaction through gesture recognition. Without loss of generality, a gesture depends on both a pose and a movement, each of which can be either static or dynamic. Therefore, it is logical to analyze and compare the potential of static as well as dynamic pose estimation as starting points for gesture recognition. Both options can be utilized to predict sequences of hand poses from their respective point clouds, which, in turn, can be used for gesture classification.

In this context, the real-world application is an integral part of the objective, aiming to minimize resource consumption to ensure the real-time capability of the entire application.

1.3 Structure and Contributions

Within this thesis, various methods for integrating three-dimensional and temporal information into hand analysis will be developed and evaluated. To ensure a coherent flow, the structure of this work is outlined below. Essentially, this thesis can be divided into three distinct parts. Firstly, Chapter 2 introduces the theoretical foundations underlying this work. Subsequently, the main component of the work comprises Chapters 3 to 6, where newly developed methods and their evaluation are presented. The structure of these chapters is uniform, starting with a brief summary and thematic contextualization within the work. This is followed by the sections: 1.) Thematic introduction and outline of relevant, related works, 2.) Comprehensive presentation of methodologies, 3.) Presentation of experiments and evaluation, 4.) Discussion and conclusion.

Figure 1.2 schematically depicts the thematic relations of the methodological chapters. While the chapters *Hand Pose Estimation* (Chapter 3) and *Scene Flow Estimation* (Chapter 4) do not initially show a direct connection, the knowledge gathered there is incorporated in the subsequent Chapter 5 on *Hand Tracking*. The final methodological Chapter 6, *Gesture Recognition*, builds upon all three previous chapters. Furthermore, the graphic illustrates the input data used in each chapter, as well as the predicted outcome. The scientific contributions of these chapters are highlighted in more detail below:

- In Chapter 3, a novel approach is introduced for predicting the spatial position of important points on the hand from a 3D point cloud, referred to as hand pose estimation. The incoming hand point cloud is complemented with a support point set to expand the convex hull and appropriately focus the search space for the keypoint

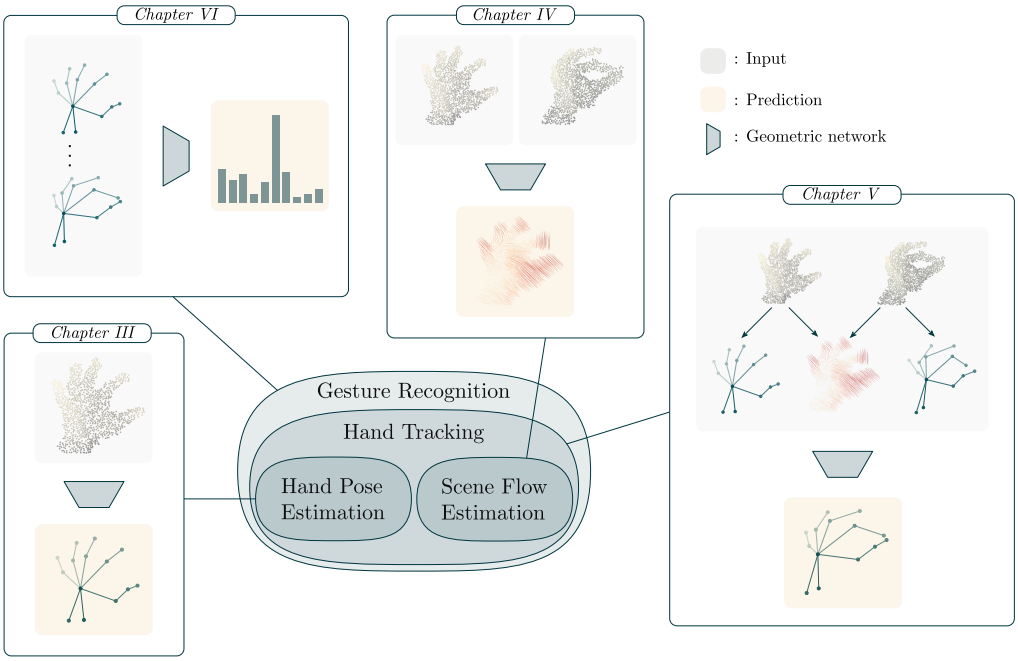


Fig. 1.2: Schematic overview of the content of the methodological chapters of this work, with subsequent chapters building upon the results of the initial ones. Each chapter comprises various geometric networks that, given a specific input, generate a prediction to address a particular task.

positions. Subsequently, the individual keypoints are computed as a weighted sum of the combined point cloud, enabling the proposed hierarchical graph CNN to learn one-dimensional weights only. This approach achieves a reduction in computational complexity while simultaneously enhancing accuracy compared to other state-of-the-art methods. The presented methods and parts of the experiments have been published in:

[Hermes et al., 2022] Hermes, N., Hansen, L., Bigalke, A., and Heinrich, M. P. "Support Point Sets for Improving Contactless Interaction in Geometric Learning for Hand Pose Estimation". In: *Bildverarbeitung für die Medizin 2022*. Springer Fachmedien Wiesbaden, 2022, pp. 89–94

- Chapter 4 focuses on the motion analysis of objects within a scene. A widely used method to determine the motion of objects, given a sequence of 3D point clouds, is the estimation of scene flow. Broadly speaking, this involves computing a displacement vector for each point of the clouds between two time points. However, existing evalu-

ations of these methods often rely on datasets where objects undergo only local rigid transformations. Since typical hand movements involve deformations and changes in shape, a new benchmark with specifically prepared datasets is introduced. This benchmark particularly investigates the three sub-tasks of estimating translation, rotation, and deformation of objects. Evaluating existing methods on these new datasets has contributed to a broader understanding in the field of hand movement. The described contributions have been published in:

[Hermes et al., 2023] Hermes, N., Bigalke, A., and Heinrich, M. P. "Point cloud-based scene flow estimation on realistically deformable objects: A benchmark of deep learning-based methods". *Journal of Visual Communication and Image Representation* 95, 2023, p. 103893

- The approach introduced in Chapter 3 estimates a hand pose based on a single given hand point cloud. Chapter 5 focuses on analyzing information from a sequence of point clouds to achieve an improved estimation of a hand pose. To achieve this, the insights gained from the motion analysis in Chapter 4 are used to optimize various scene flow methods for estimating hand skeleton movements. The skeleton flow is embedded into a module to leverage the temporal information of hand poses. Additionally, it is complemented by another module that examines individual hand poses for anatomical plausibility and corrects them if necessary. The interplay of these modules allows for the enhancement of any existing hand pose estimations within a sequence. Besides the improvement of accuracy, the evaluation also demonstrates that the presented network exhibits generalizability, making it largely independent of the choice of the method used for per-frame-based hand pose estimations. The methodology has been accepted for publication and is currently in press:

  [Hermes et al., in press] Hermes, N., Bigalke, A., and Heinrich, M. P. "Incorporating Temporal Information into 3D Hand Pose Estimation using Scene Flow". In: *VISAPP - 19th International Conference on Computer Vision Theory and Applications*. in press

- After introducing methods for utilizing either individual 3D point clouds for hand pose estimation or a sequence of point clouds for more accurate estimation of keypoint positions, the final methodological Chapter 6 of this work aims to investigate whether the identified keypoints are suitable for gesture recognition. To this end, experiments are conducted for an existing lightweight network for gesture recognition based on sequences of hand poses, that was trained and evaluated on public datasets. The network is trained using the predicted hand poses and, for better comparability, also using pseudo ground truth keypoint positions. The results of the experiments show that our estimated hand poses are even more effective for gesture recognition than the pseudo ground truth keypoints provided by the datasets. Furthermore, it

is shown that the recognition of more subtle gestures benefits from the additional integration of temporal information and the associated improvements in keypoint prediction.

As the final part of this work, Chapter 7 provides a brief summary of the methods presented and discusses the most important insights gathered in relation to the objectives of this work. The chapter concludes this work with a discussion of limitations encountered in the methods and introduces potential avenues for future development.

Chapter 2

Background

The aim of this chapter is to establish the theoretical foundations that form the basis for the algorithms developed within this thesis. To achieve this, Section 2.1 initially introduces various geometric domains on which functions and formalisms can be defined. Subsequently, the tasks addressed within this work are examined in more detail, datasets used are presented and suitable metrics for evaluation are introduced. As the primary focus of this work is the analysis of three-dimensional data, this section in particular covers tasks related to three-dimensional geometric domains. In the last section of this chapter, the fundamental concepts of deep learning on geometric domains are explained, along with a more detailed focus on some widely used approaches.

2.1 Geometric Domains

Let us once again consider the overarching goal of analyzing the geometric properties of a hand to draw semantic conclusions based on these properties. In an abstract sense, the hand under investigation exists within a system, such as a predefined space in which the hand can move freely. For humans, there is no need to further specify such a system, as the focus is not on the geometric structures of the hand but rather on additional information, such as language or facial expressions, used for interpreting gestures [Clough et al., 2020]. However, machine interpretation requires mapping this unspecified system into a known domain in which well-defined mathematical laws or rules are also established and must be adhered to. Subsequently, geometric domains are explained that can be utilized to represent hands in a machine-processable manner.

2.1.1 Graphs

A graph is one of the most fundamental structures for modeling various objects, including their relationships and connections. For example, it can be used to represent the relationships in a family tree or interactions between users in a social network. In this abstract form, a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ can be seen as a collection of objects $\mathcal{V}$, also known as *nodes*, and their connections $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, referred to as *edges*. Depending on the specific application, it often makes sense to impose additional requirements on

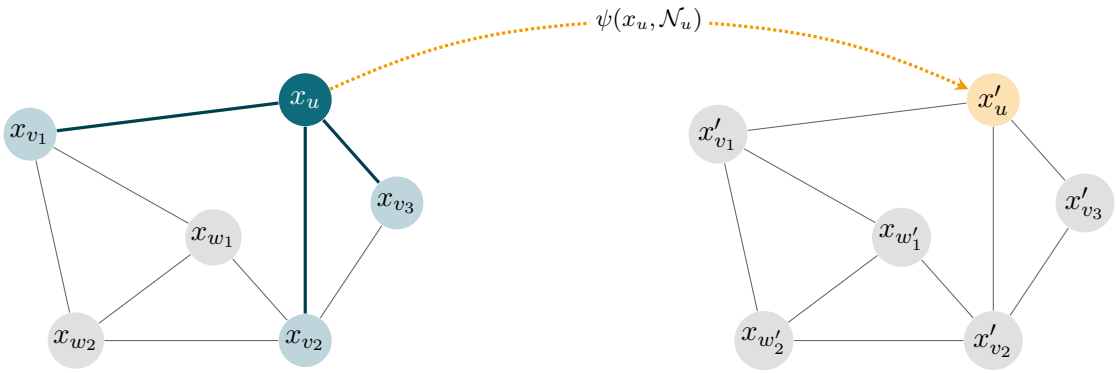


Fig. 2.1: Example visualization for constructing a permutation-equivariant function on a graph. The illustration shows the application of a permutation-invariant function ψ for the node u. The function utilizes both the features x_u of the node and the features of the local neighborhood $\mathcal{N}_u = \{v_1, v_2, v_3\}$ of u to generate a new feature vector x'_u. If this process is repeated for all other nodes in parallel, a new graph with permutation-equivariant features is obtained.

a graph, thus specializing it further. For example, the edges of a graph can indicate not only connectivity but also direction. This means that the edge (u, v) represents a connection from node u to node v, without implying that v also has a connection to u. This type of graph is also called a *directed* graph. If the edges have no direction, the graph is referred to as *undirected*.

In general, no specific order of the nodes of a graph can be assumed. An essential requirement for functions defined and applied to graphs is therefore permutation invariance. This means that the result of a given function ψ on two graphs should be identical if they only differ in the order of nodes. This allows a graph to be translated into information regardless of its order. However, often the interest lies in local features of the graph, not the entire information. In this case, permutation-equivariant functions are required to adequately permute the resulting output of the function when the graph is permuted. One way to create permutation-equivariant functions on graphs is to apply a permutation-invariant function ψ on the local neighborhood of each node, as exemplified in Fig. 2.1. Here, the features x_v of the local neighborhood $\mathcal{N}_u$ of node u are transformed into a new feature x'_u through the permutation-invariant function ψ. Performing this step for all other nodes results in a permutation-equivariant function over the entire graph.

2.1.2 Grids

In the field of computer vision, one of the most widely used geometric domain is a grid. Typically, grids are planar graphs with their vertices defined in a 2D plane. The edges

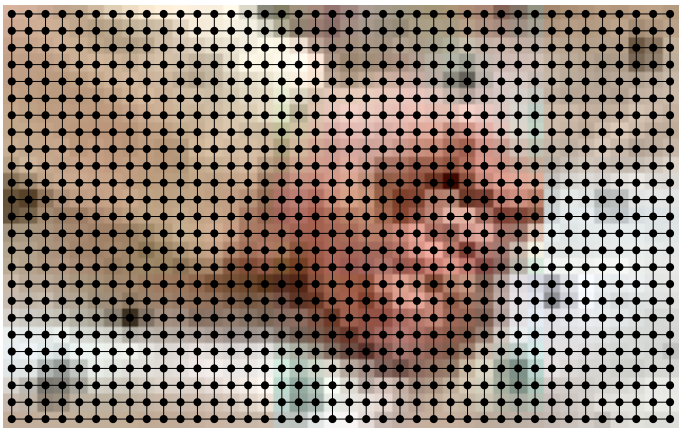


Fig. 2.2: Example visualization of an image embedded into a grid. All vertices are connected to their vertical and horizontal neighbors. The color information is shown in the background of the vertices for better illustration.

in a grid are defined to directly connect neighboring vertices. Due to their regular structure, grids have ordered nodes, which effectively addresses the issue of permutation invariance. The regular structure of grids simplifies the definition of many algorithms and operations, such as convolution. Therefore, grids are particularly suitable for representing and processing 2D images, as they already possess a planar structure.

A grid can be directly acquired using cameras. Since the sensor typically stores image data in a matrix format, the image data can be directly converted into a grid, as exemplified in Fig. 2.2. The pixel positions directly define the position and order of the vertices, which contain color data as features. Since the neighborhood relationships are also provided by the sensor, the connections between vertices can be defined with appropriate considerations for boundary conditions.

2.1.3 Manifolds

The most accurate mathematical description of a human hand or a human body is that of a manifold. In its simplest sense, a manifold describes a topological space that, when examined locally, is close to Euclidean space. An illustrative example of a manifold is the earth's surface. Globally, it resembles a sphere or ellipsoid and is not a Euclidean space. However, when examined locally, meaning zooming in on the earth's surface, we can approximate it with a plane, making it similar to $\mathbb{R}^2$. Similarly, the surface of the

human body can be regarded as a manifold because, at every point, a neighborhood can be found that resembles a 2-dimensional Euclidean space.

In practice, the manifold itself is rarely available. Instead, discretization must be employed. A commonly used technique in computer graphics is the generation of a triangular mesh. To do this, n points on the manifold with their corresponding positions $x_1, ..., x_n$ are sampled, creating a so called *point cloud*. Subsequently, triangles are formed from these points, such that the assembly of all triangles approximates the manifold. This process also describes the generation of a graph, where the nodes are derived from the point cloud and the edges correspond to the sides of the triangles. While creating the graph connections for a triangular mesh is by no means trivial, a simpler option is to utilize the k-nearest neighbor algorithm (k-NN) to define the connections of the graph for each node.

The point cloud of an object can be obtained using specific sensors such as time-of-flight. A drawback, however, is that this typically results in only a subset of the surface of the object. Without further optimization, this generally provides the point cloud for only a portion of the manifold.

2.2 Tasks on Geometric Domains

The presented geometric domains offer the potential to consider and solve a variety of different tasks. Within this work, the focus will be on pose estimation, scene flow determination and gesture recognition. Depending on the chosen geometric domain, the definition of these tasks also varies. Therefore, in the following, we will focus on manifolds and their discretization possibilities, particularly 3D point clouds.

2.2.1 Pose Estimation

The goal of 3D pose estimation is to recognize the transformation of an object, which essentially consists of translation, rotation, scale and deformation. This transformation is often estimated based on the 3D positions of specific structures that best describe the spatial characteristics of the object under investigation. These keypoints can represent anatomical structures, such as joint positions or fingertip locations, as well as prominent geometric points like the center of the hand. A particular challenge in this context is that the keypoints may not necessarily be directly part of the considered geometric input domain. For example, when considering a manifold describing a finger, joints and the fingertip are typically defined as keypoints. By definition, the fingertip's position is also part of the finger's surface and thus part of the observed manifold. However, the positions of the joints are not part of the surface and are located outside the manifold. This problem also naturally exists when dealing with discretizations of the manifold.

One way to determine the positions of keypoints is to attach sensors or markers to the object under investigation. For example, the motion capture system from

Xsens [Movella Inc, 2023] is based on magnetic sensors, gyroscopes, and accelerometers attached to the body to determine the position and orientation of individual segments. After a body-specific calibration, the entire motion of a human body or hand can be approximated. These marker-based systems are well-suited for generating data or creating 3D models but are not practical for gesture control in real-world applications due to the increased effort required to attach sensors and perform calibration.

With the success of deep learning-based methods in image processing, image-based, markerless approaches for pose estimation have been actively developed. In these approaches, image data such as RGB or 3D data is used, features are detected and analyzed, and then the positions of keypoints are determined [Moon et al., 2018; Zimmermann et al., 2017]. Below, 3D pose estimation on point clouds is explained as one possible discretization of a manifold in more detail.

We aim to utilize the information from an incoming 3D point cloud $\mathcal{P}$ to determine the 3D positions of a predefined set of keypoints $\mathcal{K} \in \mathbb{R}^{N_K \times 3}$ with the number of keypoints $N_K \in \mathbb{N}$. A common approach is to estimate an offset vector field V from which the position of a keypoint $k \in \mathcal{K}$ can be inferred. This means that for each point $p \in \mathcal{P}$ in a point cloud, the vector $v = k - p$ pointing directly to the keypoint's position is sought. Assuming that each point contributes equally to the solution, the mean of all positions can be calculated to determine the keypoint's position. However, it has been observed in practice that as the Euclidean distance from a point increases, the error in determining the vector also increases. To account for this, another approach is to decompose the offset vectors into a heatmap $\mathcal{H}$ and a uniform vector field $\mathcal{U}$ [Ge et al., 2018b; Wan et al., 2018]. An illustration of this decomposition of information is shown in Fig. 2.3. In the heatmap (Fig. 2.3b), the Euclidean distance to the keypoint is encoded at each point. A suitable approach is to set the ground truth heatmap values as

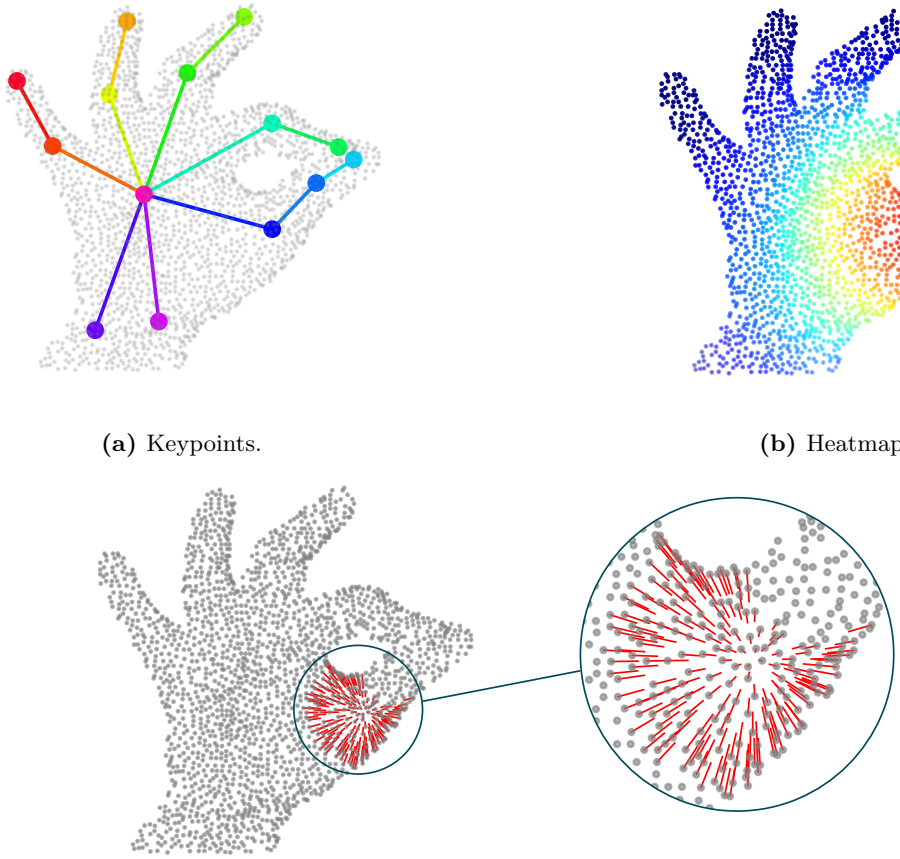
$$\mathcal{H}(p,k) = \begin{cases} 1 - \frac{\|k-p\|_2}{r}, & \text{for } \|k-p\|_2 \leq r \\ 0, & \text{else} \end{cases} \tag{2.1}$$

for an appropriate maximum radius $r \in \mathbb{R}$. The uniform vector field (Fig. 2.3c) encodes the directions to the ground truth keypoint with

$$\mathcal{U}(p,k) = \begin{cases} \frac{k-p}{\|k-p\|_2}, & \text{for } \|k-p\|_2 \leq r \\ 0, & \text{else} \end{cases} \tag{2.2}$$

The heatmap and the unit vector field can be used to infer the vector field V using

$$V(p,k) = r(1 - \mathcal{H}(p_i,k))\mathcal{U}(p_i,k). \tag{2.3}$$



(a) Keypoints.

(b) Heatmap.

(c) Unit vector field.

Fig. 2.3: Visualization of inferring keypoint positions. The first image (a) illustrates an exemplary definition of hand keypoints. The second and third image demonstrate inferring a single keypoint's position using a heatmap (b) to encode the distance information and a unit vector field (c) that contains the directions from the points of the cloud towards the keypoint.

A weighted mean ultimately leads to the 3D keypoint position $\hat{k}$, where nearby points having a greater influence on the position determination than points farther away:

$$\hat{k} = \frac{\sum_{i=1}^{N_P} \mathcal{H}(p_i, k)\,(V(p_i, k) + p_i)}{\sum_{i=1}^{N_P} \mathcal{H}(p_i, k)}. \tag{2.4}$$

During training, the ground truth positions of the keypoints can be used to evaluate the accuracy of the heatmaps and vector fields generated by the network. In the testing

phase, where no ground truth keypoints are available, the predicted heatmaps and vector fields are used to infer the keypoint positions.

Datasets. Within this work, the primary focus is on hand pose estimation using 3D point clouds. Consequently, this task is primarily centered on two specific datasets that serve as valuable resources for training and evaluating hand pose estimation algorithms.

The NYU [Tompson et al., 2014] hand pose dataset consists of 72 757 frames in the training set and 8252 frames for evaluation. Each frame includes color data, depth maps captured by Kinect and annotated ground truth 3D positions of 36 keypoints. Notably, this dataset provides both real depth data and synthetically rendered depth data for hand poses. It is important to mention that the training data is recorded from a single subject, while the test set includes data from two different subjects.

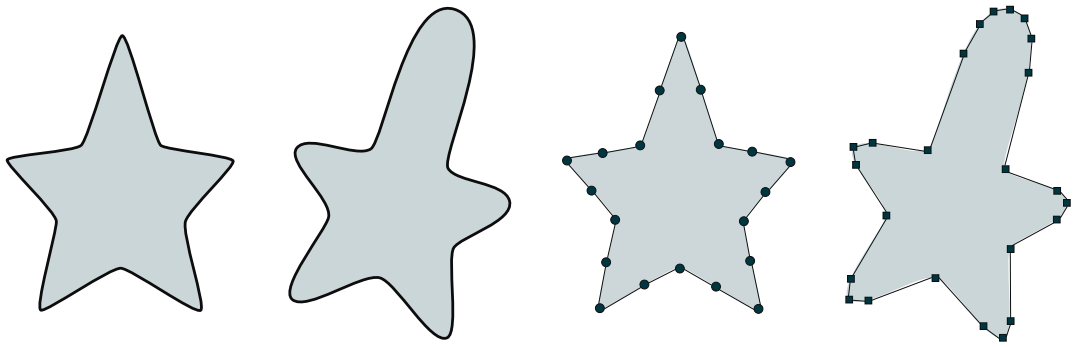
The MSRA [Sun et al., 2015] hand gesture dataset provides depth images along with ground truth 3D positions of 21 hand keypoints. Data is captured from nine different subjects, involving 17 different gestures each containing 500 frames. Unlike the NYU dataset, the MSRA dataset does not provide the entire depth image but a cropped image only containing the hand. Both datasets include camera intrinsic parameters, enabling the straightforward conversion of depth maps into 3D point clouds.

Metrics. To obtain a rough measure of the quality of a hand pose estimation method, the mean *endpoint error* (EPE) is often used. This is the average of the Euclidean distances between the estimated keypoints and the ground truth keypoints. For a more precise assessment, the so called *accuracy* is used. Accuracy determines the proportion of successfully estimated hand poses, where successfully in this context means that the EPE of all keypoints must be below a certain threshold. By varying the threshold, an accuracy curve for a method can be obtained, which provides a more detailed understanding of its performance.

2.2.2 Scene Flow

3D scene flow is closely related to 2D optical flow and deals with the estimation of three-dimensional motion in a scene. In a narrower sense, the goal is to predict the motion of objects in an image sequence relative to an observer, in order to draw conclusions about the dynamics of the scene. Assuming image data of a scene taken at two consecutive time points, t and $t + \Delta t$, is available. The objective is to estimate the motion of the objects during the time Δt. Without loss of generality, objects in this time can undergo translation, rotation, and deformation. For simplicity we further assume that the scene contains only one object to be analyzed.

An illustration of the following explanations can be seen in Fig 2.4. First, let us consider the more abstract case of two given manifolds, $\mathcal{M}_s$ (Fig. 2.4a) and $\mathcal{M}_t$ (Fig. 2.4b), where $\mathcal{M}_s$ describes an object at time t, and $\mathcal{M}_t$ describes the same object at time $t+\Delta t$. The goal in this abstract case is to find a motion transformation κ_M that maps the source manifold $\mathcal{M}_s$ to the target manifold $\mathcal{M}_t$, such that $\kappa_M(\mathcal{M}_s) = \mathcal{M}_t$.



(a) $\mathcal{M}_s$ **(b)** $\mathcal{M}_t$ **(c)** $\mathcal{G}_s$ **(d)** $\mathcal{G}_t$

Fig. 2.4: Exemplary representation of two manifolds captured at two consecutive time points for which motion within the time difference needs to be determined (left). On the right are the respective graphs as potential discretizations of the manifolds.

To establish a practical connection, we move from manifolds to graphs, which are one possible way to discretize these objects. Therefore, the graphs $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s)$ (Fig. 2.4c) and $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t)$ (Fig. 2.4d) show potential discrete representations of the manifolds $\mathcal{M}_s$ and $\mathcal{M}_t$. However, discretization is often accompanied with a loss of information. For instance, certain local structures might not be represented at a specific point in time due to occlusions or different sampling of the same structure. Therefore it is a better goal to to find a transformation that best approximates $\mathcal{G}_s$ to $\mathcal{M}_t$ instead of finding a transformation that maps $\mathcal{G}_s$ to $\mathcal{G}_t$. It is meaningful to search for a transformation that satisfies the following properties:

1. The transformed graph $\kappa(\mathcal{G}_s)$ should exhibit the smallest possible distance to $\mathcal{G}_t$, with respect to an appropriate distance metric D.

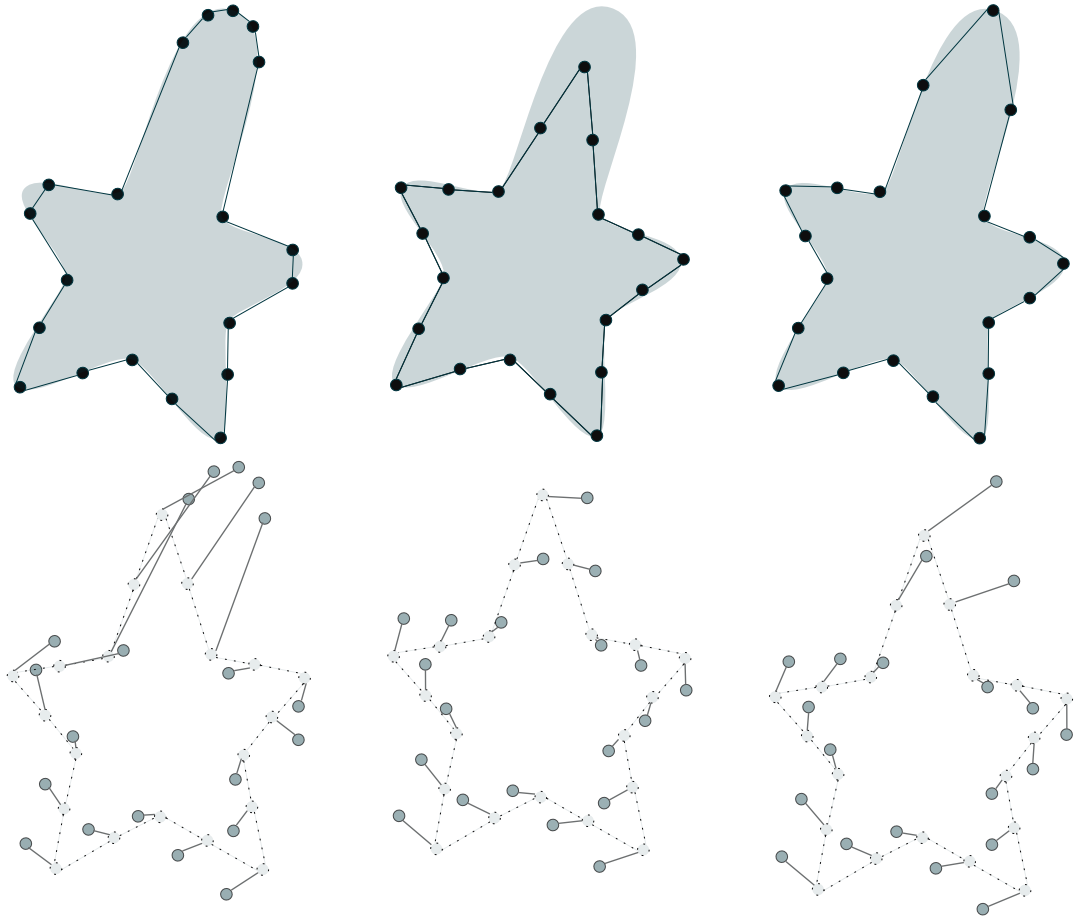
2. Neighborhood distances within $\mathcal{G}_s$ should be preserved as much as possible.

In the discrete case, the transformation κ can be further specified. An appropriate approach is the application of a flow vector field $\mathcal{F} \in \mathbb{R}^{|\mathcal{V}_s| \times 3}$ to all vertices of the graph. This allows achieving both rotation and translation, as well as deformation of the object to be mapped. With the required properties in mind, this leads to:

$$\mathcal{F}(\mathcal{G}_s, \mathcal{G}_t) = \underset{F \in \mathbb{R}^{|\mathcal{V}_s| \times 3}}{\arg\min}\ D(\mathcal{G}_s + F, \mathcal{G}_t) + \lambda R_{neighbor}(\mathcal{G}_s, F) \tag{2.5}$$

with the neighborhood-regularisation

$$R_{neighbor}(\mathcal{G}_s, F) = \sum_{v \in \mathcal{V}_s} \frac{1}{|\mathcal{N}_v|} \sum_{w \in \mathcal{N}_v} |\|w - v\|_2 - \|w + F(\mathcal{I}(w)) - (v + F(\mathcal{I}(v)))\|_2|. \tag{2.6}$$



(a) Limited neighborhood regularization.

(b) Limited target graph similarity.

(c) Both properties demanded.

Fig. 2.5: Exemplary representation of the transformed graphs $\mathcal{G}_s + \mathcal{F}$ for potential scene flow estimations, depending on the conditions imposed on the transformation. On the left is a transformed graph with an estimated flow that does not fully meet the second condition and in the middle, the first condition is not fully enforced. If both properties are satisfied, the graph depicted on the right could be a potential solution. The top row shows the transformed graphs while the bottom row visualizes the flow fields $\mathcal{F}$.

In this context, $D : (\mathcal{G}, \mathcal{G}) \to \mathbb{R}$ represents a suitable distance measure for the similarity of two graphs and the function $\mathcal{I}(\cdot)$ maps from a vertex to its corresponding index in the set of vertices.

An illustration of the importance of the mentioned properties for the flow field $\mathcal{F}$ is shown in Fig 2.5 for the exemplary graphs $\mathcal{G}_s$ and $\mathcal{G}_t$. Figure 2.5a illustrates a suboptimal solution when neighborhood regularization is not included. While the graph (top) appears plausible, a closer look at the motion of the points (bottom)

reveals that semantic correspondences change in part. For instance, the two points in the bend between the small and large spikes are considered part of the large spike in the transformed graph. Conversely, an overly strong emphasis on neighborhood regularization results in an implausible graph (Fig. 2.5b). If both properties are demanded for the transformation, then the graph depicted in Fig. 2.5c could be an appropriate solution.

Datasets. Due to the challenging nature of acquiring knowledge about point correspondences in a point cloud, generating real datasets for scene flow estimation is extremely complicated. Therefore, within this part of the work, synthetic datasets from the domain of human body pose estimation are used.

The Dynamic FAUST [Bogo et al., 2017] dataset provides 3D scans of ten humans, consisting of five male and five female subjects. Each of them performed 14 various sequences, recorded at a frame rate of 60 fps. In addition to the 3D scans, a sparsely populated cloud of registered points is provided, meaning the correspondences of these points are known across time. These correspondences are generated by best fitting an SMPL body model [Loper et al., 2015] to the 3D data.

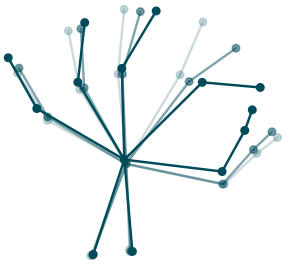
As a second dataset, the SURREAL dataset is utilized in this work for scene flow estimation. In this dataset, an SMPL body model is initially used to generate moving sequences of the model through human motion capture data. Furthermore, the model is used to render synthetic depth data for the sequences. Since the parameters of the SMPL model are known for each frame, a sparsely populated cloud of points with known correspondences can be generated here as well. Overall, this extensive dataset provides 55 001 clips within 1964 sequences containing 115 different subjects for a training subset and 12 528 clips within 703 sequences containing 30 different subjects, whereby every clip contains mostly 100 frames.

Metrics. Assuming that there is a ground truth flow vector to be estimated for each point, the following metrics can be used to evaluate scene flow estimation methods. Similar to the field of hand pose estimation, the endpoint error (EPE) between estimated and ground truth flow vectors is used to obtain a rough measure of the quality of the estimations. For a more detailed analysis, this area often includes the three metrics *ACC Strict*, *ACC Relax* und *Outliers* [Jin et al., 2022; Tishchenko et al., 2020; Wei et al., 2021]. Here, ACC Strict is defined as the proportion of points with an EPE $< 5\,\text{mm}$ or a relative error $< 5\%$, while the less stringent ACC Relax relaxes the criteria as the fraction of points with EPE $< 10\,\text{mm}$ or a relative error $< 10\%$. The Outliers metric provides a characteristic value for the amount of points that significantly deviate from the ground truth and is defined as the fraction of points with an EPE $> 30\,\text{mm}$ or a relative error $> 10\%$.

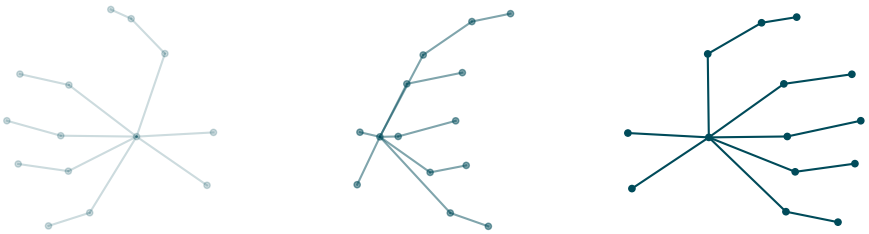
2.2.3 Gesture Recognition

Hand gestures play a significant role in daily communication and can facilitate intuitive interaction between humans and machines. A hand gesture typically consists of a specific pose and a hand movement. Both aspects can exist in static or dynamic forms, as exemplified in Fig. 2.6 for excerpts from a sequence. In the Pinch gesture, the thumb and index finger are brought together and then separated. The pose in this gesture is thus dynamic, while the movement of the entire hand usually remains marginal, making it static. In contrast, the Swipe gesture involves moving the open hand from left to right or vice versa. Here, the pose remains constant and is static, while the entire hand moves, resulting in a dynamic movement. The recognition of gestures is particularly challenging due to variations in pose and movement for a specific gesture across different subjects.

There are generally two fundamental approaches related to the recognition of gestures on 3D point clouds. One approach involves recognizing gestures based on sequences of point clouds [Bigalke et al., 2021], allowing the incorporation of the entire information



(a) Pinch



(b) Swipe.

Fig. 2.6: Exemplary representation of hand poses at three different time points within a sequence where a gesture is performed. The above example illustrates a Pinch gesture, while below a Swipe gesture is shown.

content of this data structure. However, this method requires processing a large number of points and thus usually comes along with higher computational effort. With advancements in geometric deep learning leading to various methods for determining hand poses, another approach involves analyzing hand poses to identify the gesture in a sequence [Shi et al., 2020; Yang et al., 2020]. Since typically fewer than 30 points are contained in each pose, only a few points need to be analyzed, enabling methods with extremely low processing times.

Datasets. The evaluation of gesture recognition methods on point clouds is particularly well-suited using the two public datasets DHG [De Smedt et al., 2016] and SHREC [De Smedt et al., 2017], as they provide not only labeled gestures but also depth data and 22 annotated keypoint positions per frame. Both datasets feature a total of 14 different gestures performed by subjects in two different ways: using fingers and utilizing the entire hand. For the SHREC dataset, 28 subjects were involved in recording, performing gestures between 1 and 10 times. This dataset comprises 2800 sequences for evaluation purposes. On the other hand, the DHG dataset contains sequences from 20 different subjects, each performing the gestures five times, resulting in a total of 2800 sequences available for analysis.

2.3 Geometric Deep Learning

In a more abstract sense, the fundamental concept of supervised machine learning is to draw general conclusions from given observations. Let us assume that we have a set of N observations $\mathcal{O} = \{(x_i, y_i)\}_{i=1}^{N}$. In this case, the incoming data x_i and the associated labels y_i may originate from different domains $\mathcal{X}$ and $\mathcal{Y}$. Furthermore, we assume that an unknown mapping function f generates the labels y with $f(x_i) = y_i$. The goal of learning algorithms is to approximate this unknown function by a parameterizable function $\tilde{f}_\Theta$.

In neural networks, as a possible realization of the parameterizable function, the parameters correspond to the learnable weights of the network. This implies that the network's weights have to be optimized in such a way that the distance between the functions is as small as possible, which corresponds to minimizing the function

$$\mathcal{L}(\Theta) = \sum_{i=1}^{N} d(\tilde{f}_\Theta(x_i), y_i), \tag{2.7}$$

where $d : (\mathcal{Y}, \mathcal{Y}) \to \mathbb{R}$ is a suitable distance measure between the predicted and the observed labels.

The foundation of neural networks is the artificial neuron, which is conceptually inspired by the biological neuron and processes various inputs and combines them into an output. At its core is a linear function $\psi(x) = w^T x + b$ to process the incoming signal. In general, each layer can also have multiple output neurons. Since the linearity

of the function f to be approximated cannot be assumed, it is sensible to introduce a non-linear component. In the case of the artificial neuron, this is the so-called activation function σ, such as a sigmoid or ReLU function, resulting in the definition:

$$\psi(x) = \sigma(w^T x + b). \tag{2.8}$$

In a multi-layer network, the parameterizable function is represented as a composition of many different functions $\tilde{f}_\Theta(x) = (\tilde{f}_{N_{f_\Theta}} \circ \cdots \circ \tilde{f}_{2_\Theta} \circ \tilde{f}_{1_\Theta})(x)$, forming a feed forward network. Each individual function constitutes a layer of the entire network, with the last layer generating the desired output and thus being called the output layer. The middle layers neither receive the observations directly nor output the desired outcome. Instead, they merely transmit a signal. This results in the optimization of the entire network as a whole, rather than each layer individually. For this reason, the middle layers are also referred to as hidden layers.

When the presented artificial neurons are created in multiple layers according to this architecture and connected with each other, meaning that a neuron in one layer is linked to all neurons in the following layer, it is referred to as a fully-connected network or a multi-layer perceptron (MLP). Due to the numerous connections, a complete network involves a high number of parameters, which can complicate the optimization process especially for high-dimensional inputs. Subsequently, insights into other network architectures that leverage the structure of the geometric domain are provided.

2.3.1 Convolutional Neural Networks

Within the field of computer vision, the success of deep learning architectures can largely be attributed to Convolutional Neural Networks (CNNs) [He et al., 2016]. The fundamental principle involves feature extraction through the convolution of incoming data with learnable parameters with the motivation to establish an architecture that simplifies the machine learning process. This is mainly achieved by two important factors. Firstly, incoming data is convolved with kernels of smaller size than the input data. This achieves sparse connectivity, meaning individual neurons are not connected to all neurons in the subsequent layer but selectively within a local patch. Secondly, the weights within convolutional layers are shared. This means that a kernel does not use different weights per position but the same weights are applied to every position. These two techniques lead to a significant reduction in the number of parameters to be learned [Goodfellow et al., 2016].

This is fundamentally enabled by the essential characteristics of the underlying domain for CNNs, which are grids. The identical connectivity of all nodes, except boundaries, to their respective neighbors within this geometric domain ensures that a kernel with a given structure can be applied at every position of the grid. Additionally, the given order of nodes eliminates the need to consider permutation invariance in the

design of a function ψ. Thus, assuming suitable boundary conditions, a 2D CNN layer for an image I on the grid $\mathcal{G}_I$ at position (i, j) can be defined as

$$h_{i,j} = P(\sigma(\psi(x))) = P(\sigma((I * K_\Theta)_{i,j})) \tag{2.9}$$

with the kernel $K_\Theta \in \mathbb{R}^{(2S_K+1)\times(2S_K+1)}$ and the discrete convolution

$$(I * K_\Theta)_{i,j} = \sum_{m,\ n=-S_K}^{S_K} I_{i+m,j+n} K_{\Theta_{m,n}}. \tag{2.10}$$

As before, $\sigma(\cdot)$ represents an activation function and the function $P(\cdot)$ denotes a pooling strategy. The goal of pooling is to transform the incoming grid $\mathcal{G}_I$ into a coarser grid $\hat{\mathcal{G}}_I$. This enables a larger receptive field while keeping the number of parameters constant, thus achieving better invariance to local deformations and aggregation of contextual information.

Over time, new insights have continually expanded the basic concept of CNNs. Thus, Dropout [Srivastava et al., 2014] serves as a principle to prevent overfitting during training by randomly deactivating selected neurons within an iteration. Additionally, batch normalization [Ioffe et al., 2015] can be applied to accelerate and stabilize the training process. In the presented ResNet by He et al. [2016], the CNN layer from Eq. 2.9 is extended with an identity path, where the incoming signal is simply forwarded:

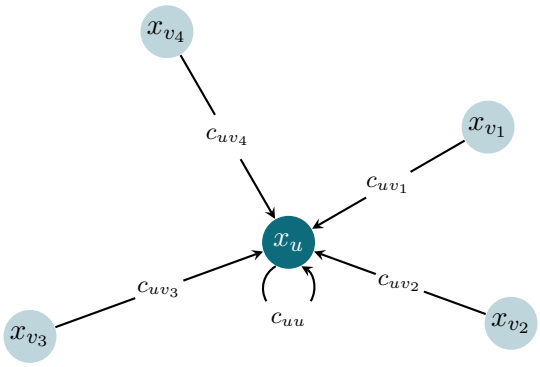
$$h_{i,j} = P(\sigma(x + \psi(x))) \tag{2.11}$$

This approach addresses the vanishing gradient problem, allowing for increased network depth.

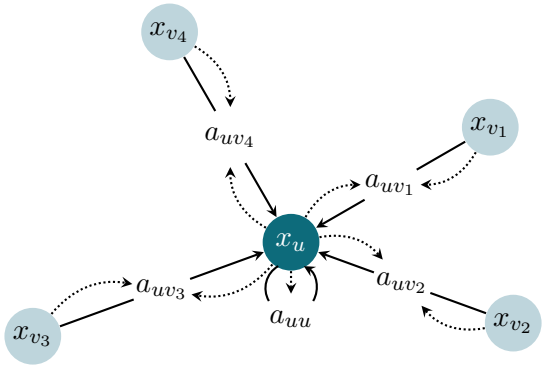
2.3.2 Graph Neural Networks

For 3D point clouds as a discretization of a manifold, no specific order of the points can be assumed, so the central property for applying traditional CNNs is not given. Therefore, in this section, layers that are designed for more general graphs and can be applied to point clouds or meshes are introduced. These layers are known as Graph Neural Network (GNN) layers and they primarily involve applying permutation-invariant functions within a local neighborhood to process the available graph information. While the coordinates of data points in CNNs are implicitly utilized through convolution with a filter, graph neural networks can explicitly exploit the positions of the points as features.

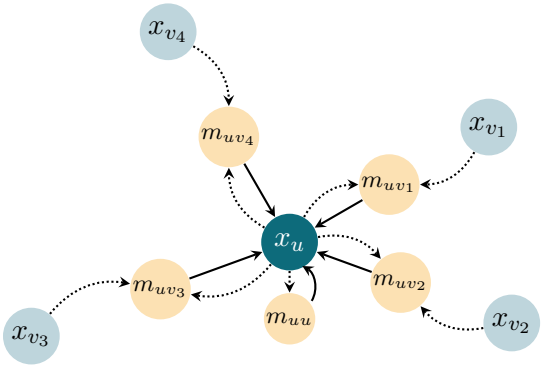
The neighborhood features are aggregated by a symmetric function $\bigoplus$ and subsequently update the node u through a learnable function ϕ. To maintain permutation invariance, the aggregation function must also be permutation-invariant. Typically a sum, mean, or maximum function is used for this purpose.



(a) Convolutional graph layer.



(b) Attentional graph layer.



(c) Message-passing graph layer.

Fig. 2.7: An illustration of the different concepts for GNN layers to analyze the local neighborhood around a node u with its associated feature vector x_u. As an example, a neighborhood consisting of four nodes is visualized. a) Each neighboring node is assigned a constant weight c_{uv} and multiplied with its respective feature vector. b) Information from receiver x_u and sender x_v is used to calculate the attention $\alpha_{uv} = a(x_u, x_v)$, which is then multiplied with the sender's feature vector. c) The feature vectors of the senders are not directly transmitted to the receiver but indirectly through a message vector $m_{uv} = \psi(x_u, x_v)$.

For these types of GNN layers, there are various concepts for analyzing information within local neighborhoods, as depicted in Fig. 2.7. For a more in-depth understanding of this field, we refer interested readers to the work of Bronstein et al. [2021], whose fundamental concepts we outline below, accompanied by examples and potential applications.

Convolutional Graph Layers

The convolutional graph layer, Fig. 2.7a, conceptually follows the structure of a CNN layer. In this context, for a point u, the information from neighboring points is weighted multiplicatively before aggregation. These weights are intended to quantify

the significance of the contribution of an adjacent point v to the currently considered point u. Furthermore, it is important to note that the features of neighboring points can be transformed through a learnable function ψ. This leads to the following expression:

$$h_u = \phi \left(x_u, \bigoplus_{v \in \mathcal{N}_u} c_{uv} \psi(x_v) \right). \tag{2.12}$$

Point-Wise Graph Layers

A well-known example of such a layer is PointNet [Qi et al., 2016], whose underlying key idea is to approximate a general function on a point cloud by employing a symmetric function on the transformed points of the cloud. In this approach, each point in the cloud is considered individually, meaning that neighborhoods are defined as $\mathcal{N}_u = \{u\}$. In this specific case, Eq. 2.12 simplifies to:

$$h_u = \psi(x_u). \tag{2.13}$$

This highly simplified representation of neighborhoods in a point cloud simultaneously ensures the permutation invariance of the method. In this case, the learnable function ψ is defined as a multi-layer perceptron. The entire network utilizes multiple PointNet-modules to extract features from all points within a point cloud. These features are aggregated through a max-pool operation and combined into a global feature vector. Depending on the task to be accomplished, the global feature vector is used differently. For a classification task, another MLP is employed to directly transform the global feature into the desired output. In the case of point cloud segmentation, the generated global features are concatenated with the local individual point features. Subsequently, two additional PointNet-modules generate per-point scores.

While the individual point-wise consideration allows for efficient feature computation, it also leads to the network neglecting local structures, resulting in the omission of crucial information.

Attentional Graph Layers

The attentional layer, Fig. 2.7b, extends the general concept of convolutional layers and uses a function $a(x_u, x_v)$ instead of fixed weights c_{uv}. This function computes attention based on the features of the two nodes u and v. The mathematical framework for such layers can be described as follows:

$$h_u = \phi \left(x_u, \bigoplus_{v \in \mathcal{N}_u} a(x_u, x_v) \psi(x_v) \right). \tag{2.14}$$

It is evident that a convolutional layer is a special case of an attentional graph layer.

In natural language processing, attention mechanisms in transformer networks have achieved impressive results and revolutionized the research field [Dai et al., 2019; Devlin et al., 2019; Vaswani et al., 2017]. The fundamental idea is to define a mapping function that transforms a query along with a set of key-value pairs into attention. The query is an enhanced representation of the input word sequence. In conjunction with the keys, which are generated for each word and determine their importances, it is used to search for relevant information with the context of a specific question to be answered. The values represent the actual information associated with the words. These values are weighted by the queries and keys. Within the transformer network, this information is collectively used to draw conclusions about context and relevant information. To account for word order and the overall structure of the input sequence, this approach is extended with positional encoding concepts.

This idea can also be applied to graphs to focus attention on relevant structures. A well-known approach is the PointTransformer layer [Zhao et al., 2021], which utilizes the following function to compute self-attention for two nodes u and v:

$$a(x_u, x_v) = \rho(\Theta_\gamma(\phi(x_u) - \sigma(x_v) + \delta_{uv})). \tag{2.15}$$

In this process, the feature vectors x_u and x_v are initially transformed by the learnable functions ϕ and σ, followed by computing their difference and adding the positional encoding δ. Subsequently, the features are mapped with the encoding function Θ_γ and normalized through the function ρ, which can be for instance a softmax function. The positional encoding is defined as the difference of positions, followed by a mapping using the learnable function Θ_δ:

$$\delta_{uv} = \Theta_\delta(u - v). \tag{2.16}$$

Both encoding functions Θ_δ and Θ_γ are defined by the authors as MLPs with two linear layers and the non-linear ReLU function. With the feature transformation $\psi_u(x_v) = \alpha(x_v + \delta_{uv})$ and summation as the aggregation function, the entire layer is obtained through the expression

$$h_u = \sum_{v \in \mathcal{N}_u} a(x_u, x_v) \odot \alpha(x_v + \delta_{uv}). \tag{2.17}$$

Within the entire Transformer network, the presented PointTransformer layers are embedded in an encoder-decoder architecture. Up and downsampling are performed to consider both local and global relevant structures of the point cloud to capture detailed semantic structures.

Message-Passing Graph Layers

Another concept for gathering information within the local neighborhood of a graph considers edges as a means of sending messages, as shown in Fig. 2.7c. The following expression describes this concept:

$$h_u = \phi\left(x_u, \bigoplus_{v \in \mathcal{N}_u} \psi(x_u, x_v)\right) \tag{2.18}$$

Initially, a message $\psi(x_u, x_v)$ is computed based on a learnable function ψ and the information content of both, the sender x_v and the receiver x_u. The subsequent aggregation operator can be interpreted as the transmission of the sent messages.

The preceding concepts can be considered specializations of message-passing, with this concept being the most generic. However, creating and processing messages can be memory and time-consuming, making the specialized cases a good choice for many applications.

In the Dynamic Graph CNN, presented by Wang et al. [2018], the message-passing concept is used to extract features from point clouds to draw semantic inferences and solve various tasks. In the so-called EdgeConv operation, edge features are defined as $e_{uv} = \psi_\Theta(x_u, x_v)$, where ψ_Θ represents a non-linear function with learnable parameters Θ. With a symmetric aggregation operation, we can thus generally obtain

$$h_u = \bigoplus_{v \in \mathcal{N}_u} \psi_\Theta(x_u, x_v)), \tag{2.19}$$

wherein a wide variety of edge features can be utilized in principle. For instance, if one uses

$$\psi_\Theta(x_u, x_v) = \psi_\Theta(x_u), \tag{2.20}$$

the previously introduced PointNet module is obtained, which solely evaluates global features while ignoring local structures. To account for local structures, the option

$$\psi_\Theta(x_u, x_v) = \psi_\Theta(x_v) \tag{2.21}$$

can be used. However, in this case only local structures are encoded and there is no consideration of the global structure. Therefore, the authors propose encoding both local and global information in the messages using the function

$$\psi_\Theta(x_u, x_v) = \psi_\Theta(x_u, x_v - x_u). \tag{2.22}$$

Specifically, the authors utilize the edge features

$$e_{uvm} = \mathrm{ReLU}(\Theta_m(x_v - x_u) + \Lambda_m x_u) \tag{2.23}$$

with the learnable parameters Θ_m and Λ_m. Thus, with a max function as the aggregation operator, the EdgeConv operation can be defined as:

$$h_{um} = \max_{v \in \mathcal{N}_u} e_{uvm}. \tag{2.24}$$

To apply the EdgeConv operation, edges are required in addition to the 3D point cloud. A straightforward approach for generating the set of edges is the k-nearest neighbor method with Euclidean distances. Using feature distances instead, this methodology can also be applied after an EdgeConv operation to dynamically adapt the graph to the new feature space.

Chapter 3

Hand Pose Estimation on 3D Point Clouds

In this first methodological chapter a novel approach is introduced that enables the learning of meaningful features on 3D point clouds to infer the positions of specific hand keypoints and thereby estimate hand poses. The hand pose, together with hand movement, is a crucial component of a human hand gesture and thus an essential part of hand analysis. At the core of the new approach is a so-called support point cloud that is defined around the incoming 3D hand point cloud. This focuses the space for possible keypoint positions and offers a stable structure, especially in the presence of hand occlusions. Through experiments on two publicly available datasets, the accuracy of the proposed method is demonstrated and the impact of the support point cloud is investigated. Additionally, possible variants to establish semantic connections between support and hand points are evaluated. The method and aspects of the experiments were presented and published [Hermes et al., 2022] as a contribution to the conference *Bildverarbeitung für die Medizin 2022*.

3.1 Introduction

The determination of the hand pose is becoming increasingly important in computer assisted surgery [Hein et al., 2021] as well as for interaction in virtual reality and augmented reality systems. In medical applications an important factor is contactless interaction with devices either to ensure sterile working conditions or to enable treatments from a distance. This requires accurate and robust determination of the hand pose in real time, which is still challenging despite the progress made so far.

Until recently, 3D hand pose estimation has been dominated by methods relying on depth images [Du et al., 2019; Oberweger et al., 2015, 2017; Wan et al., 2017, 2018; Wang et al., 2018; Xiong et al., 2019; Zhang et al., 2020; Zhou et al., 2018b]. Depth maps enable the processing by powerful CNNs while simultaneously including 3D information [Chatzis et al., 2020]. However, they still constitute a projection of 3D data onto a 2D plane, so direct regression of a 3D pose is a highly nonlinear task that is difficult to learn.

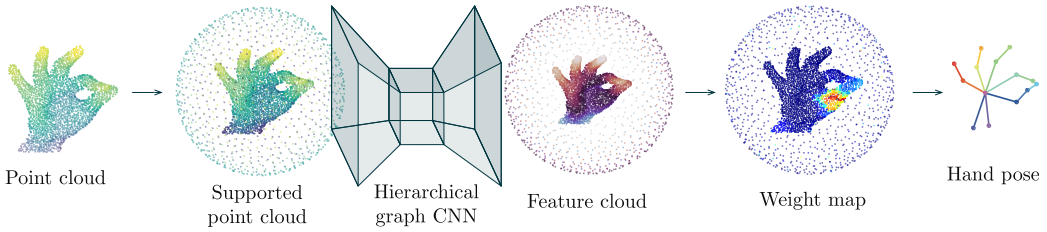


Fig. 3.1: Overview of the proposed method. We extend the input point cloud by a support point set. The supported point cloud is fed into a hierarchical graph CNN that connects the hand and the support point set and shares their information. The resulting features are transferred into a weight map to infer the final 3D hand pose as weighted sum over all points.

On the contrary 3D point clouds preserve the original 3D structure and thus enable a more natural representation for 3D pose estimation. This has been leveraged in multiple works [Chen et al., 2018b; Ge et al., 2018b; Li et al., 2019a; Moon et al., 2018; Wang et al., 2018], which directly regress the 3D pose from 3D geometric input domains [Bronstein et al., 2016]. Specifically for raw 3D point clouds as input, most of the methods predict the 3D pose by estimating offset vectors or fields with respect to the input points. This procedure suffers from two problems: First, estimating offset vectors is still a complex problem that requires complex network architectures. Second, missing hand parts in the point cloud due to occlusions are still a major problem, since joints with larger distance to the point cloud are more prone to faulty offset predictions. This is a severe limitation in clinical practice, since the focus is on the patient and his treatment, and an optimal camera position is only secondary. Therefore, self-occlusion is a common issue in medical applications.

We address these challenges by making the following three contributions (as illustrated in Fig. 3.1): Firstly, we propose to estimate the position of a joint as the weighted sum of all input points. So instead of an offset vector, we only need to learn a scalar weight for each input point. This is a substantially simpler task that can be solved with a less complex network. However, this procedure only allows estimating joints inside the convex hull of the input cloud, which is quite limited. Therefore, secondly, we propose to complement the input point cloud by a support point set, which is arranged around the input cloud and thus enables a more comprehensive prediction of joint locations. To jointly process the unified set, we develop a hierarchical graph CNN whose graph structure enables information flow between hand points and support points. In particular, geometric information is propagated from the hand points to the surrounding support points, which in turn can improve joint prediction.

3.1.1 Related Work

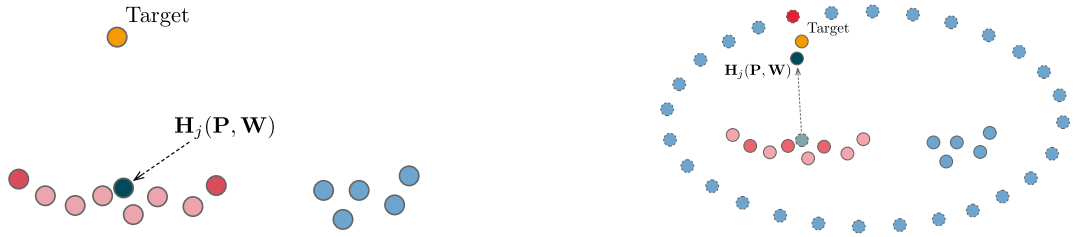
In this section, the related work in the field of hand pose estimation based on 3D information is briefly summarized. To enhance structure, existing methods are categorized according to the underlying modality.

3D hand pose from depth images. Depth images, while not a natural representation of a three-dimensional scene and containing less spatial information than 3D point clouds, can be processed by CNNs due to their underlying grid structure. Wan et al. [2018] employ two stacks of hourglass modules to generate features that are transformed into 2D heatmaps, 3D heatmaps and unit vector fields to infer the 3D pose. This approach leverages both the 2D information of the depth map for localization, as well as the depth for the spatial estimation of the keypoints. Du et al. [2019] apply a similar approach to extract heatmaps from the depth image, but they divide the pose determination into two subtasks: determining the palm pose and determining the finger pose. The respective poses are predicted in two separate branches, while cross connections ensure that each branch benefits from the information of the other subtask.

3D geometric domains for hand pose estimation. Point clouds can be generated from depth maps, providing a natural representation of the captured scene. In the presented *Point-to-Point Regression PointNet* (P2P) [Ge et al., 2018b], PointNet layers [Qi et al., 2016] are employed at different resolutions of a point cloud to extract local and global spatial features from the input data structure. From these features, point-wise estimates of distances and directions to the respective keypoints are generated to infer the 3D pose. Moon et al. [2018] transfer the depth map into a 3D voxelized grid. This allows the use of 3D CNNs that are employed in an encoder-decoder architecture to estimate per-voxel likelihood for the keypoints. While this methodology leverages the spatial information effectively, the use of 3D CNNs imposes significant computational overhead.

3.2 Methods

We aim to estimate the 3D hand pose which is equivalent to finding the location of all hand joints. The input to our proposed method is a set of points $\mathbf{P} \in \mathbb{R}^{N_p \times 3}$ with $N_p \in \mathbb{N}_+$ being the number of points. We assume that the given point set only contains a hand, where parts of the hand can be missing due to occlusions. The output is a set of joint positions $\mathbf{H} \in \mathbb{R}^{J \times 3}$ with the number of joints $J \in \mathbb{N}_+$.



(a) Point set. **(b)** Supported point set.

Fig. 3.2: On the left an input point cloud and a target joint ■ are shown. During estimation we aim to find a weight (■ and ■) for each input point such that the weighted sum of all input points yields the joint position. The right side shows an example estimation and the resulting joint position $\mathbf{H}(\mathbf{P}, \mathbf{W})$ ■ for a supported point set.

3.2.1 Support point sets

A core concept of our method is to estimate the position of the hand joints as the weighted sum over all input points $p_i \in \mathbf{P}$ with $i = 1, ..., N_p$. Specifically, we define the position of joint j as:

$$\mathbf{H}_j(\mathbf{P}, \mathbf{W}) = \frac{\sum_{i=1}^{N_p} w_{ij}\mathbf{p}i}{\sum_{i=1}^{N_p} w_{ij}}, \tag{3.1}$$

where the weights $w_{ij} \in [0, 1]$, represented by $\mathbf{W}$, need to be inferred from the input cloud. However, this joint representation has the limitation that it only maps to positions that lie within the convex hull of the given set of points $\mathbf{P}$. Figure 3.2a visualizes this problem for an example point cloud and a joint. Even though the estimation ignores unimportant points by assigning them weights close to zero, the estimated joint position is far away from the real position, since in this example the joint is located outside of the convex hull of the point cloud. When point clouds are generated from a single view depth image, some parts of the hand are occluded and not reflected in the point cloud. These occluded hand joints mostly cannot be mapped using representation (3.1).

In order to solve this problem we create an additional point set to expand the convex hull of the hand. We define a *support point set* $\mathbf{U}$ by randomly choosing $N_p \in \mathbb{N}_+$ points from the unit sphere $\{x \in \mathbb{R}^3 : \|x\|_2 = 1\}$. To better capture the shape of a hand we scale these support points depending on the standard deviation vector σ of the hand point set and translate them towards the center μ:

$$\mathbf{B} = d(\sigma \circ \mathbf{U}) + \mu, \tag{3.2}$$

where $\circ$ denotes the Hadamard product and d a scaling factor $d \in \mathbb{R}$. This support set is unified with the original input cloud, yielding a supported hand point set $\mathbf{P}_{\text{sup}} = \mathbf{P} \cup \mathbf{B}$.

Figure 3.2b visualizes such a supported point set. As the convex hull is now enlarged, the joint representation (3.1) can directly map to the true position of the joint.

3.2.2 Network architecture

While many hierarchical network architectures use a PointNet [Qi et al., 2016, 2017] to compute features, we propose a hierarchical Graph CNN [Wang et al., 2018] to learn weight maps on supported point sets. A graph has the advantage that it can connect the points of the two semantically different point clouds and thus enables an exchange of information. The combination with the hierarchical network structure [Qi et al., 2017] ensures that we capture local as well as global features of both sets.

Set abstraction layers are usually used to transfer the features of a point set to a coarser resolution [Qi et al., 2017]. We propose a modified *GraphAbstraction* layer that takes into account both the hand point set and the support point set. Given an input point set of size N_{in} that consists of $N_{\text{in}}/2$ hand points and $N_{\text{in}}/2$ support points around the hand. The new resolution N_{out} is obtained using farthest-point-sampling on the hand points and support points separately. This ensures that the subsampled point set consists of 50% hand points and the other 50% support points.

Point cloud-based graph networks [Wang et al., 2018] commonly build the graph based on the K-nearest-neighbor method. However, in our case, this would result in hand and support points not being connected due to their distance. Therefore we modify this approach and enforce that each hand point connects to some support points and the other way around. The input point set and the graph connections are then fed into an EdgeConvolution layer followed by a Multi-Layer-Perceptron (MLP) to obtain the final features of the coarser resolution.

Our complete network architecture is shown in Fig. 3.3. It accepts any D-dimensional supported point set as input, where $D \in \mathbb{N}_+$. Within the first GraphAbstraction step the input cloud is subsampled to $N_1 = 1024$ feature points of dimension $C_1 = 32$ that are concatenated to the 3D positions. The two following GraphAbstraction layers process these features to an $N_2 \times (D + C_2)$ ($N_2 = 256$ and $C_2 = 128$) and subsequently to an $N_3 \times (D + C_3)$ ($N_3 = 64$ and $C_3 = 256$) dimensional feature tensor. We use the same *FeaturePropagation* layers as proposed in [Ge et al., 2018b] to combine the features of two different resolutions. To this end, the features of a coarser resolution N_{r+1} are interpolated to a finer resolution N_r using nearest-neighbor-interpolation, concatenated with the features of the GraphAbstraction layer GA_r and fed into an MLP to obtain a feature tensor of dimension $N_r \times C_r$. The final features with input resolution are fed into an MLP that outputs one weight map for each hand joint.

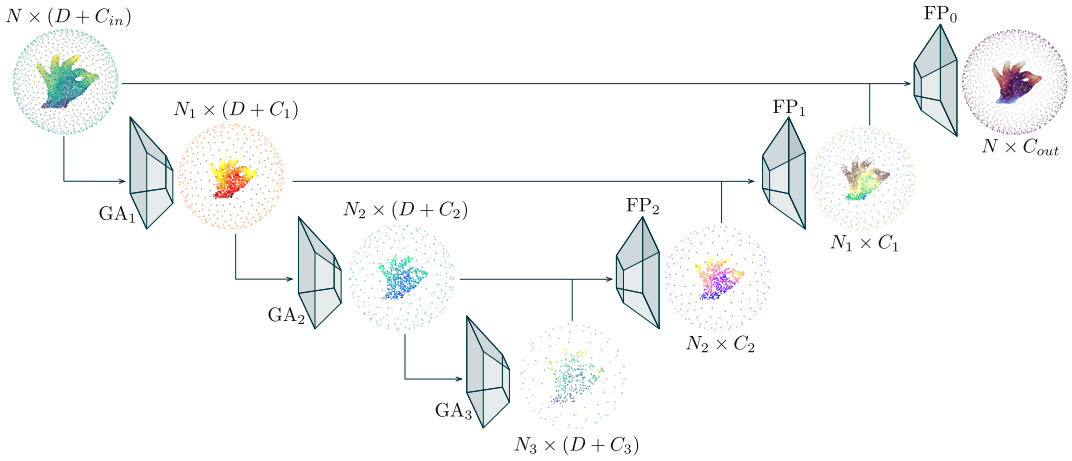


Fig. 3.3: An illustration of the proposed network architecture. The GraphAbstraction layers (GA) downsample a point set to a coarser resolution and output its features. In the FeaturePropagation layers (FP) the features of a coarser resolution are interpolated to the higher resolution. These interpolated features are concatenated with the output of the corresponding GA layer and afterwards fed into a Multi-Layer-Perceptron (MLP). The features of the highest resolution (FP0) are fed into another MLP to obtain the weight map.

We apply deep supervision when training our hierarchical network by adding MLPs to the output of the FeaturePropagation layers, each generating a weight map for all joints. This yields the definition of the following loss function for each training sample:

$$\mathcal{L}(\mathbf{P}_{\text{sup}}, \mathbf{H}_{\mathbf{t}}, \mathbf{W}) = \sum_{f=0}^{F-1} \sum_{j=1}^{J} \left\| \mathbf{H}_j(\mathbf{P}_{\text{sup}}, \mathbf{W}_f) - \mathbf{H}_{\mathbf{t}j} \right\|^2, \tag{3.3}$$

with $F \in \mathbb{N}_+$ the number of FeaturePropagation layers, the generated weight map of a FeaturePropagation layer $\mathbf{W}_f$ and the target positions of the hand joints $\mathbf{H}_{\mathbf{t}}$.

3.3 Experiments and Results

Dataset. We evaluate our proposed method on two publicly available datasets. The NYU [Tompson et al., 2014] hand pose dataset contains 8252 depth maps of a single person for testing and 72 757 depth maps of two users for training. Besides the depth image the dataset contains the 3D position of 36 hand joints. For a better comparison we follow [Oberweger et al., 2015; Tompson et al., 2014] and perform evaluation on a subset of 14 joints. We transfer the depth image into a 3D point cloud using the camera intrinsic parameters and use farthest point sampling to obtain a point cloud with 1024 points. We then perform a principal component analysis (PCA) to obtain an

orthonormal basis where the variance of the points is small along the axes and map the point cloud to that basis. In a last step we normalize the cloud by dividing it by the longest possible distance along one of the axes and shift it to the root by subtracting the mean of the cloud.

The MSRA [Sun et al., 2015] hand gesture dataset contains data from nine different subjects who performed 17 different gestures in a sequence, with each sequence comprising 500 frames per gesture. The data for each frame includes a depth map and the ground truth positions of 21 hand keypoints. We process the data in the same manner as the NYU dataset, initially using the camera intrinsic parameters to generate a point cloud from the depth data. The point cloud is then reduced to 1024 points using farthest point sampling. Subsequently, the point cloud is normalized through PCA, as done also for the NYU dataset.

Implementation Details. Our network is trained and tested on a computer with a GeForce® RTX 2080 Ti 11GB GPU. For the implementations of the deep neural network modules we use the PyTorch framework. On graph creation we connect each point in the cloud with a total of $K = 8$ other points. To enforce connections between hand and support point set, we connect each hand point with one support point and seven other hand points. Each support point is connected to seven hand points and one other support point. We extend the EdgeConvolution layer by replacing the standard max pooling aggregation function by a combination of multiple aggregation functions (PNA) [Corso et al., 2020]. During the training of our proposed networks we use the Adam optimizer with the initial learning rate 0.001, $\beta_1 = 0.5$, $\beta_2 = 0.999$ and weight decay $5e^{-4}$. Additionally the learning rate is halved every 30 epochs and training is stopped after 100 epochs.

Based on the number of trained parameters, the reduction of network complexity can be deduced. The specified network contains a total of 1 736 413 trainable parameters and thus has a size of 6.95 MB. Compared to the P2P network [Ge et al., 2018b], which has a total size of 17.2 MB, our network is much smaller.

Metrics. To demonstrate the accuracy of the proposed approach we compared it with seven state-of-the-art methods on the NYU [Tompson et al., 2014] dataset by computing two metrics. First is the mean endpoint error (EPE) as the Euclidean distance between the ground truth and the estimated joint positions. The second metric is the proportion of successfully predicted frames, where the highest joint error is below a certain threshold [Oberweger et al., 2015].

3.3.1 Impact of the Support points

The support point cloud is one of the main components of the presented method, consequently in this part different definition possibilities for such a support will be investigated. The presented method was trained and evaluated on the NYU dataset, where the support point cloud was defined in different ways. For comparison, the

Table 3.1: Endpoint errors of out method with different ways to define the support point cloud. Training and testing was performed on the NYU dataset. As the perfect support utilizes ground truth information, we mark it with * and do not count it as best result.

Support	none	hull ($d = 4.0$)	hull ($d = 4.5$)	hull ($d = 5.0$)	perfect
EPE [mm]	11.64	9.66	**9.30**	9.43	8.83*

training without a support is defined as baseline. The results of this investigation are shown in Table 3.1. As expected, our method without support achieves the worst results because the keypoints to be determined are often not part of the convex hull of the hand point cloud and thus cannot be represented. The method defined in Eq. 3.2 for calculating a hull support already achieves much better results with 9.3 mm, using a scaling factor $d = 4.5$. However, if the ground truth positions of the keypoints are used for the standard deviation instead of the points of the cloud, named perfect support in the table, then the error can be reduced to 8.83 mm. Even if this methodology is not applicable in reality due to the use of ground truth information, this result shows that even the hull support does not include all keypoints in the convex hull and thus can be improved.

3.3.2 Graph Connections

As we use a graph CNN on two semantically different point sets, it is important to connect the different point sets in a suitable manner. With the following experiments we investigate how many connections the hand points need to the support points. We hypothesized that connections within the support set are not required because the support points are intended to encode the directional information of the hand points. Table 3.2 shows the impact of varying the amount of connections between the support

Table 3.2: The impacts of the number of connections between the hand point set and the support point set on the mean error on a subset of the MSRA dataset.

$K_{Hand \to Support}$	0	0	0	0	0	1	1	1	1	1	4	4	4
$K_{Support \to Hand}$	0	1	4	7	8	0	1	4	7	8	0	1	4
EPE (mm)	8.5	8.4	8.5	8.4	8.4	8.7	8.7	8.5	**8.3**	8.5	8.6	8.6	8.5

$K_{Hand \to Support}$	4	4	7	7	7	7	7	8	8	8	8	8
$K_{Support \to Hand}$	7	8	0	1	4	7	8	0	1	4	7	8
EPE (mm)	8.4	8.4	8.5	8.7	8.6	8.5	8.4	8.5	8.6	8.4	8.5	8.6

and the hand point set. For each point a total of $K = 8$ connections were searched. The points were forced to connect with a specified number of closest points of the opposing point set and the remaining connections were searched in the own point set. We train the different settings on a small subset of the MSRA dataset (three subjects each performing two gestures) for 80 iterations and evaluate on the full test subset. Within the table we see that we achieve the lowest overall mean error when the hand points are connected to one support point and to seven other hand points. This result demonstrates that the directional information could be successfully encoded in the support points, making it sufficient for each hand point to establish a connection to one support point. On the other hand, it is beneficial for each support point to be connected to at least one other support point. We conclude that the local structure of the convex hull and thus the support points also contribute to the direction estimation and should be considered.

3.3.3 Hand Pose Estimation

Utilizing the previously obtained information, we train our proposed network on both datasets. For the generation of the support point cloud, we use a convex hull from Eq. 3.2 with a scaling factor of $d = 4.5$. Furthermore, we connect each hand point with the nearest support point and the seven nearest hand points. Conversely, we connect each support point with the nearest support point and the seven nearest hand points. As can be seen in Fig. 3.4, with an average error of 9.3 mm our method achieves comparable state-of-the-art accuracy on the NYU dataset. Compared to the other methods, this means an average reduction of the error by 20%. Only P2P [Ge et al., 2018b] achieves better results than our approach due to suitable post-processing. Without this post-processing, P2P [Ge et al., 2018b] achieves an error of 9.5 mm which

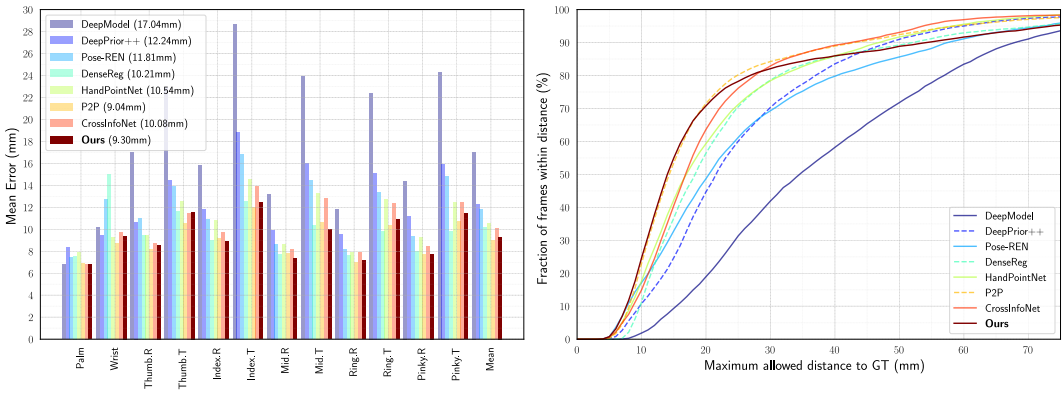


Fig. 3.4: Evaluation of the proposed method and state-of-the-art methods on the NYU dataset [Tompson et al., 2014]. The figure shows the average error of each joint (top) and the percentage of successfully predicted hand poses (bottom).

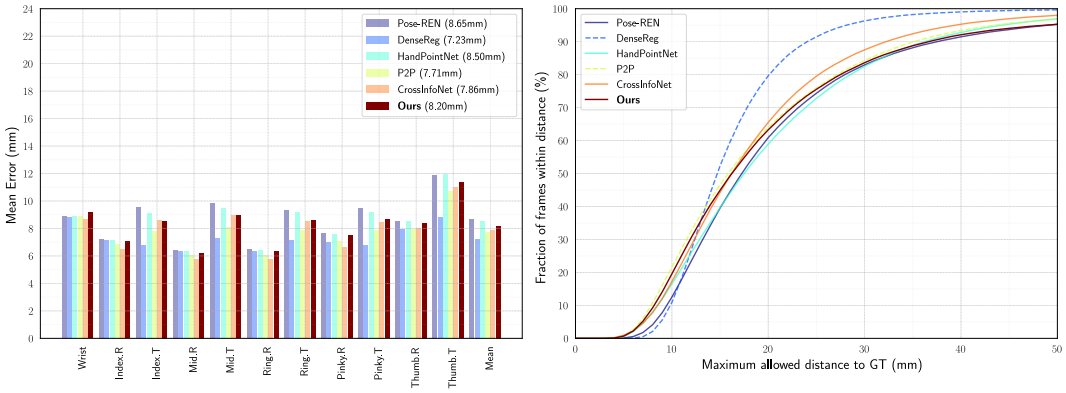


Fig. 3.5: Evaluation of the proposed method and state-of-the-art methods on the MSRA dataset [Sun et al., 2015]. The figure shows the average error of each joint (top) and the percentage of successfully predicted hand poses (bottom).

leads to the assumption that our method may achieve similar or even better results with suitable optimization.

The high accuracy of our method is also reflected in the percentage of successfully predicted frames. Up to a maximum allowed distance of 19 mm, we outperform all other methods. With a threshold of 10 mm, we correctly predict 25% of the frames, while the next best approach with the same threshold achieves only 23%. Furthermore, we achieve a proportion of 50% correctly predicted frames with a threshold of 14 mm. When the threshold is above 19 mm, P2P [Ge et al., 2018b] has a slightly higher percentage of correct predictions than the proposed method. The same is true for CrossInfoNet [Du et al., 2019] when the threshold is above 29 mm.

The results of our method on the MSRA dataset are depicted in Fig. 3.5. With an EPE of 8.2 mm, our approach performs better than Pose-REN [Chen et al., 2018a] and HandPointNet [Ge et al., 2018a] but is surpassed by CrossInfoNet [Du et al., 2019], P2P [Ge et al., 2018b] and DenseReg [Wan et al., 2018]. Examining accuracy provides additional insights. It is evident that our method processes frames successfully for a threshold of up to 16 mm, performing only slightly worse than P2P and outperforming all other methods. However, as the threshold increases, our method is eventually surpassed by the others. We conclude that while our approach achieves many particularly precise hand pose estimations on the MSRA dataset, the proportion of outliers is greater compared to other methods. However, it can be expected that this proportion can be significantly reduced with appropriate post-processing. We would also like to refer to Ge et al. [2018b], who identified frames in the MSRA dataset where the annotated ground truth of the keypoint positions is inaccurately provided, reducing the reliability of this dataset.

3.4 Discussion and Conclusion

In this work, we proposed to estimate the 3D hand pose from a point cloud as the union of the input points and a support point set. This enables us to predict the weight maps with an efficient hierarchical graph CNN, which we tailored for optimal information flow between both point sets. That way, we address the problem of occluded joints and reduce the complexity of the optimization problem. Our results demonstrate that the method achieves comparable state-of-the-art accuracy while also reducing the network complexity. By precisely estimating the hand joint positions, the gestures of a person can be detected. This enables contactless interaction with devices, which in turn eases maintaining sterile working conditions.

Chapter 4

Scene Flow Estimation on Realistically Deformable Objects

The second methodological chapter of this thesis addresses the generation and analysis of motion information within a scene to gain temporal insights into the dynamics of an object of interest. One way for analyzing object motion is the estimation of scene flow, as introduced in Chapter 2. While several promising methods have been developed in this domain in recent years, they have predominantly been evaluated on datasets with objects that are affected by global translation and rotation. However, since hands in particular undergo deformations in the form of pose changes, this chapter introduces a new benchmark that evaluates existing methods on realistically deformed objects. This aims to get new insights in the field of motion analysis. Parts of the ensuing benchmark have been published in volume 95 of the *Journal of Visual Communication and Image Representation* [Hermes et al., 2023].

4.1 Introduction

The estimation of motion fields of points in a scene is still a challenging task in computer vision. In various application areas, the analysis of the movement is an important component. In the automotive sector, it is important to analyze and evaluate the dynamic scene using radar sensors, as well as to detect the movement of objects to ensure the earliest possible reaction of the system [Battrawy et al., 2019; Lenz et al., 2011; Menze et al., 2018]. Furthermore, applications in the field of human interaction in particular benefit from the motion information [Liu et al., 2019b; Vedula et al., 1999; Wang et al., 2022].

Generally, motion analysis is divided into two different types. The optical flow refers to a projection of velocities of objects in the three-dimensional world to an image plane and thus describes the pixel-wise motion of adjacent frames [Horn et al., 1981]. In contrast, scene flow directly describes the motion of every point in a scene utilizing a

three-dimensional flow field [Vedula et al., 1999]. A naive first interpretation of the scene flow is:

$$\mathrm{f} = P_t - P_s, \tag{4.1}$$

with source and target point clouds $P_s, P_t \in \mathbb{R}^{N\times 3}$. For each point in the source cloud, the corresponding point in the target cloud is searched to determine the scene flow, assuming both clouds have the same size and sort order. However, this assumption is not realistic, since the data acquisition is always subject to noise and in general, point clouds are unordered and have different sizes. A formulation that better reflects reality is

$$\mathrm{f} = \underset{\tilde{\mathrm{f}}\in\mathbb{R}^{N_s\times 3}}{\arg\min}\, D(P_s + \tilde{\mathrm{f}};\ P_t), \tag{4.2}$$

with a suitable dissimilarity function $D : (\mathbb{R}^{N_s\times 3};\ \mathbb{R}^{N_t\times 3}) \to \mathbb{R}$. This means we seek the flow f such that the transformed source cloud $\hat{P}_s = P_s + \mathrm{f}$ is as similar as possible to the target cloud P_t. Also in this case, hard correspondences between point clouds are sought, which is generally not an optimal solution due to the data acquisition process. This can be prevented by transforming the target point cloud into a more general representation $R(P_t)$, such as a surface representation, in order to also allow soft correspondences as a solution:

$$\mathrm{f} = \underset{\tilde{\mathrm{f}}\in\mathbb{R}^{N_s\times 3}}{\arg\min}\, D(P_s + \tilde{\mathrm{f}};\ R(P_t)). \tag{4.3}$$

Within the last few years, deep learning-based methods have established themselves in many tasks on point clouds [Chen et al., 2019a; Fan et al., 2019; Lin et al., 2021; Min et al., 2020; Owoyemi et al., 2018; Sarode et al., 2019; Wang et al., 2018; Zhou et al., 2018a]. Also, for this specific task, they have become increasingly important and currently dominate the state-of-the-art.

4.1.1 Related Work

Especially for deep learning-based methods, the creation of meaningful benchmarks is essential for the development of new methods, as they allow researchers to see which patterns the networks can learn. A basic requirement is therefore that the datasets used for evaluation represent reality as well as possible to avoid the creation of a bias towards a perfect, unrealistic environment.

Most benchmarks of scene flow methods [Gu et al., 2019; Liu et al., 2019b; Puy et al., 2020; Tishchenko et al., 2020; Wu et al., 2020] build on the two datasets FlyingThings3D [Mayer et al., 2016] and KITTI [Menze et al., 2015b, 2018]. The FlyingThings3D dataset contains stereo frames with ground truth data that are generated within synthetically

created scenarios. Each scene is based on randomly chosen components but taken along a defined scheme. To generate a sequence from the scene, the objects and the camera are translated and rotated along linear 3D trajectories. KITTI also contains stereo frames, but these were recorded using calibrated cameras on a car from a real scene on the road. The ground truth data was recorded with a laser scanner, which however only generates sparse data. To increase the precision of the ground truth data, 3D models of cars were fitted to the point cloud to achieve a denser resolution. In the end, the ground truth data is still an approximation.

Apart from scaling, both datasets as well as other datasets from the autonomous driving sector such as Waymo Open Dataset [Sun et al., 2020] or GTA-SF [Jin et al., 2022] only contain scenarios where objects are affected by rigid transformations. This makes the datasets very suitable for training and evaluation with respect to other tasks, such as point cloud registration [Ao et al., 2021; Poiesi et al., (early access) 2022]. On the other hand, the representation of the world through piecewise rigid transformations is a strong simplification. Humans in particular can only be poorly represented with this assumption, so it is important to use datasets with realistically deformed objects for a benchmark in order to be able to make statements about use cases such as hand or body pose tracking.

Several previous works have already released benchmarks on deformed data Bhatnagar et al., 2020; Deng et al., 2022; Feng et al., 2021; Li et al., 2022; Shen et al., 2021, but the focus was on registration procedures and some established standard methods such as ICP [Besl et al., 1992] and CPD [Myronenko et al., 2010]. Li et al. [2022] propose to use sequences from DeformingThings4D [Li et al., 2021b] for a benchmark, but due to both the low overlap between source and target and the large global movement, the benchmark is better suited for registration procedures than for scene flow methods. Furthermore, these characteristics make it difficult to draw conclusions for realistic tracking applications. Shen et al. [2021] propose a benchmark on the PVT1010 dataset which contains 1010 lung vessel trees between exhale and inhale. However, the limitation of the used dataset is that ground truth labels are only available for 10 cases and with 300, only a few landmarks are available.

Given this lack of appropriate benchmarks for scene flow methods on realistic deformed data, we set up a suitable benchmark to obtain important new insights.

4.1.2 Contributions

In the following we shortly summarize our contributions of this work.

- Preparation of datasets for training and evaluation of scene flow estimation approaches. The datasets that are commonly used for training and evaluation mostly contain scenarios with objects that are affected by rigid transformations. We use various existing datasets, which were created mainly for human pose estimation and part segmentation, and specifically prepare them for scene flow estimation with objects

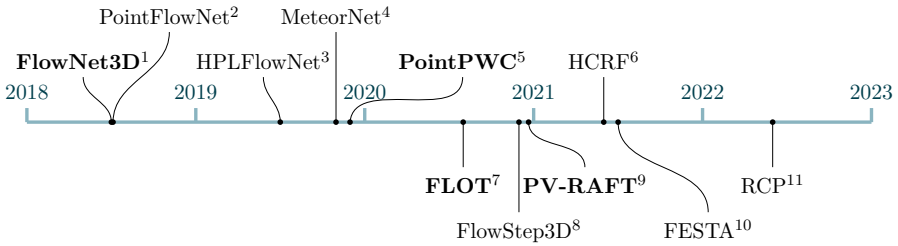
that are also affected by realistic deformations and therefore pose an even greater challenge.

- Introduction of new metrics to ensure the separate analysis of the methods regarding correspondences as well as similarity of the point clouds.
- Creation of a benchmark of scene flow estimation approaches on deformed objects. This involves training and evaluation of existing approaches on our prepared datasets to analyze their performance in scenarios with deformed objects. We aim to help researchers, especially in the field of object tracking with a deeper understanding of the scene flow estimation methods.
- A comprehensive analysis of the runtime of different methods. The runtime is often left out of benchmarks. However, since it is very important in real-time applications, we also explore dependencies of the speed of the methods.

The rest of the chapter is organized as follows. In Sec. 4.2 we shortly review a selection of existing methods for scene flow estimation. Afterwards a performance comparison on new datasets and further experiments are presented in Sec. 4.3. Finally, we conclude and shortly discuss the results in Sec. 4.4.

4.2 Methods

In recent years, researchers have made great progress in the field of scene flow estimation and developed several new methods [Behl et al., 2018; Gojcic et al., 2021; Gu et al., 2019; Kittenplon et al., 2020; Li et al., 2021a; Liu et al., 2019b,c; Puy et al., 2020; Teed et al., 2020; Wu et al., 2020]. In general, deep learning-based methods for determining



[1] [Liu et al., 2019b]
[2] [Behl et al., 2018]
[3] [Gu et al., 2019]
[4] [Liu et al., 2019c]
[5] [Wu et al., 2020]
[6] [Li et al., 2021a]
[7] [Puy et al., 2020]
[8] [Kittenplon et al., 2020]
[9] [Wei et al., 2021]
[10] [Wang et al., 2021]
[11] [Gu et al., 2022]

Fig. 4.1: Chronological overview of one-shot methods (top) and iterative methods (bottom) for scene flow estimation on 3D point clouds. The methods we chose for further analysis are marked in bold type.

correlations between 3D points can be clustered into one-shot solutions and iterative methods. While initially the data-driven one-shot methods [Behl et al., 2018; Gu et al., 2019; Li et al., 2021a; Liu et al., 2019b,c; Wu et al., 2020] were developed to combine the features of both point clouds to determine the scene flow directly, with the advent of algorithm unrolling techniques [Li et al., 2019d; Monga et al., 2021] the iterative methods [Gu et al., 2022; Kittenplon et al., 2020; Puy et al., 2020; Wang et al., 2021; Wei et al., 2021] have also shown their success by combining feature learning and application of prior knowledge. A chronological overview of the development of scene flow estimation methods on 3D point clouds is shown in Fig. 4.1.
We have carefully chosen the most significant methods of these two clusters, which represent milestones in the estimation of scene flow and therefore find their place in most benchmarks. Within this section, we briefly review our selection of approaches. Aiming for a detailed performance analysis, we will also limit the experiments to the presented methods.

4.2.1 FlowNet3D

The FlowNet3D [Liu et al., 2019c] network was one of the earliest deep neural networks that learn scene flow estimation from point clouds in an end-to-end fashion. As can be seen from Fig. 4.2, the basic concept is similar to PointNet++ [Qi et al., 2017] extended by a flow embedding layer in order to find correlations between two point clouds. First, features are generated in the set conv layer with shared weights for both the source cloud and the target cloud, following the hierarchical network architecture

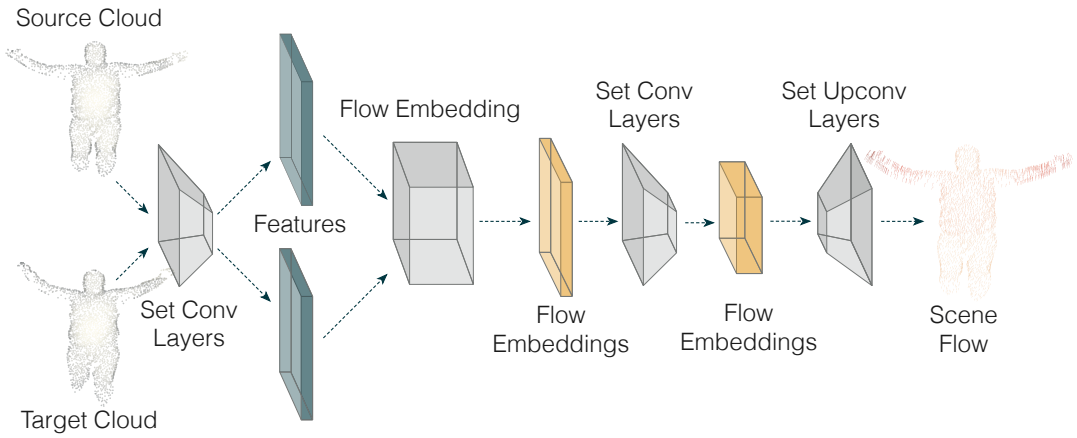


Fig. 4.2: Overview of the FlowNet3D network architecture. Set conv layers are used to create local features for source and target cloud. The features of both clouds are used to produce flow embeddings, which are fed into set upconv layer to generate the final flow predictions.

of the PointNet++. Subsequently, in the flow embedding layer, feature similarities and spatial relationships of the points of both point clouds are combined to encode the movement of the points. This is achieved by first searching for neighboring points of the target cloud for each point of the source cloud and afterwards using a neural layer that learns to weigh the neighbors for potential flow vectors. In the flow refinement step, the flow embeddings are upsampled to the resolution of the input point clouds and the final scene flow is predicted by another neural layer.

4.2.2 FLOT

Puy et al. [2020] interpret the point clouds as graphs within their published method called FLOT and substitute the problem of finding correspondences between point clouds into a problem of matching two graphs. This allows the use of techniques from optimal transport theory, which among other things also deals with comparisons and matching of graphs [Maretic et al., 2019; Titouan et al., 2019]. Conceptually, all points of the source point cloud are assigned a mass, which should be transported to the target point cloud in the best possible way, where each displacement has certain costs. The goal is then to develop a transport plan that minimizes costs and meets given conditions such as mass conservation. As Fig. 4.3 shows, the authors first use convolutional layers with shared weights to extract features on the source and the target point cloud. This allows subsequently to define the displacement cost, depending on the similarity of the features. Here, a cosine distance between the features is used to ensure a reduction of the displacement costs with increasing feature similarity. Within the

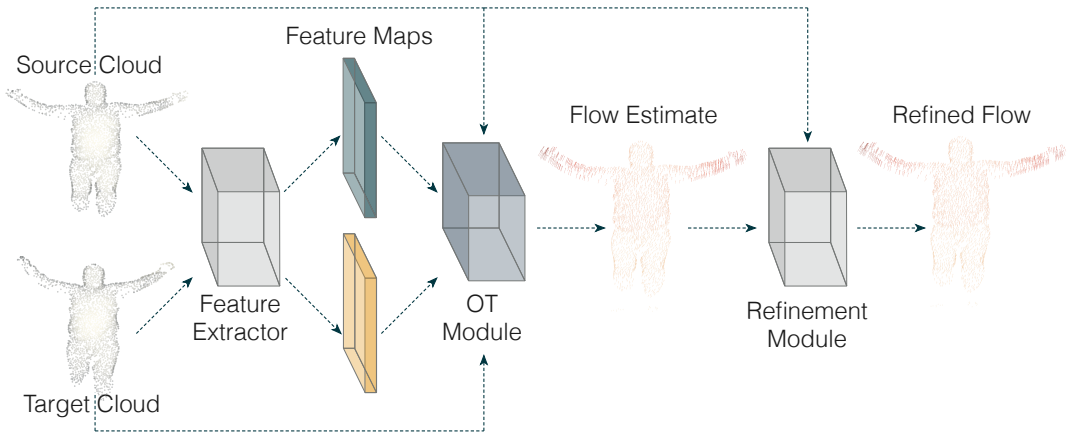


Fig. 4.3: Overview of the FLOT network. Convolutional layers extract context features of the source and the target point clouds with shared weights. Subsequently, the features are fed into the Optimal Transport (OT) module to obtain a transport plan which is used to compute a first flow estimate. The final scene flow is then computed in the refinement module.

Optimal Transport (OT) module, unrolled iterations of the Sinkhorn algorithm derived for transport problems [Chizat et al., 2016; Cuturi, 2013] are used to output a transport plan, which substitutes a first flow estimate. Since many real-world circumstances are not accounted for in the mathematical model, such as parts being occluded and therefore not visible in the target cloud, the flow estimate is refined using further convolutional layers.

4.2.3 PV-RAFT

PV-RAFT [Wei et al., 2021] is an extension of the Recurrent All-Pairs Field Transform (RAFT) [Teed et al., 2020], a published state-of-the-art deep network architecture for 2D optical flow estimation. The approach is to first extract per-pixel features from the source and target images that can be used to identify similarities in the two images. Those similarities between pixels in the two images are summarized in a correlation pyramid with the addition that the correlation of an already shifted image $I_s + f$ can also be interpolated. Starting with a predefined initial flow, a gated activation unit based on the GRU cell [Cho et al., 2014] is fed with the information of the features and the current correlation to iteratively estimate the flow between source and target images. However, many parts of the given methodology cannot be readily applied to 3D point clouds because of the irregularity of the data and the lack of neighborhood

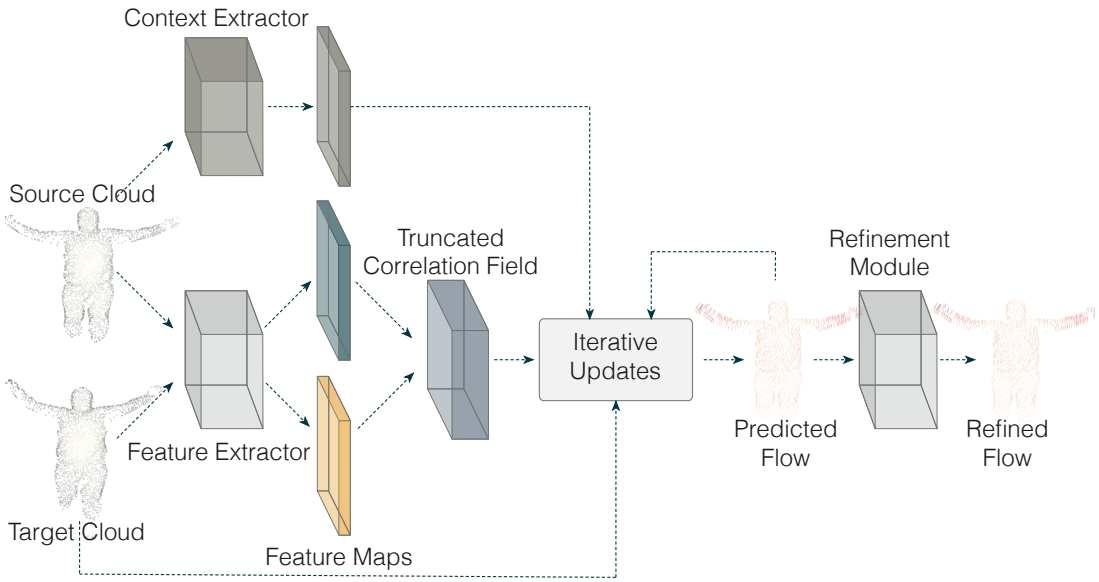


Fig. 4.4: Overview of the PV-RAFT method. After extracting the context and the features of source and target cloud, a truncated correlation field is generated and, together with the context, fed into a gated activation unit that iteratively predicts the 3D flow.

information. An overview of PV-RAFT, the extension of the described methodology to 3D point clouds, is shown in Fig. 4.4.

First, both context and features of the point clouds are extracted using a PointNet++ [Qi et al., 2017]. Assuming that the features contain similarity information, the correlation field is created by simple matrix dot product:

$$\mathcal{C} = E_\theta(P_s) \cdot E_\theta(P_t), \tag{4.4}$$

with $E_\theta(P_s)$ and $E_\theta(P_t)$ being the features of the source and the target point cloud. To reduce the required memory and computing time, the correlation volume is truncated afterwards to the top-M highest correlations, resulting in the truncated correlation field $\mathcal{C}_M \in \mathbb{R}^{N_s \times M}$, where N_s defines the number of points in the source point cloud. The correlation volume is computed only once and can also be used to compute the correlation between the transformed source cloud $Q = P_s + f$ and the target cloud, by searching the neighbors of Q in P_t. This step is performed in the published procedure using either a k-nearest-neighbor search to capture the local information or by searching voxel neighbors in cubes that are defined with different sizes and centers. The latter ensures to also integrate the long-range correlation information into the network. Equipped with context, features and correlation information, the flow is predicted iteratively in a final step. Given the current flow prediction f_{t-1}, the correlation features between the transformed source cloud $Q_{t-1} = P_s + f_{t-1}$ and the target cloud P_t are computed in the update step as the addition of the two previously described correlation lookups. The gated activation unit uses these correlation features, the context and the hidden states from previous iterations as input to compute the updated flow estimate f_t.

4.2.4 PointPWC

In their method called PointPWC, Wu et al. [2020] use a hierarchical architecture with different resolutions to estimate the scene flow, as shown in Fig. 4.5. Features of different resolutions are created for source and target point clouds. To obtain the features of a coarser resolution, the points are downsampled using farthest-point sampling and a PointConv [Wu et al., 2019] produces the features of the lower resolution. Starting from the coarsest resolution, a refined flow is now generated level by level using the published layers, until the initial resolution is reached. Each refinement step has the same structure. At first, the coarse flow is upsampled using an inverse distance weighted interpolation. This upsampled flow is used to warp the features of finer resolution as an element-wise addition. To find the correspondences between the source and target points, a cost volume layer is learned, which receives the warped source features and the target features as input. Cost volumes have already been successfully learned on 2D image data and optical flow estimation [Sun et al., 2018; Xu et al., 2017], but the

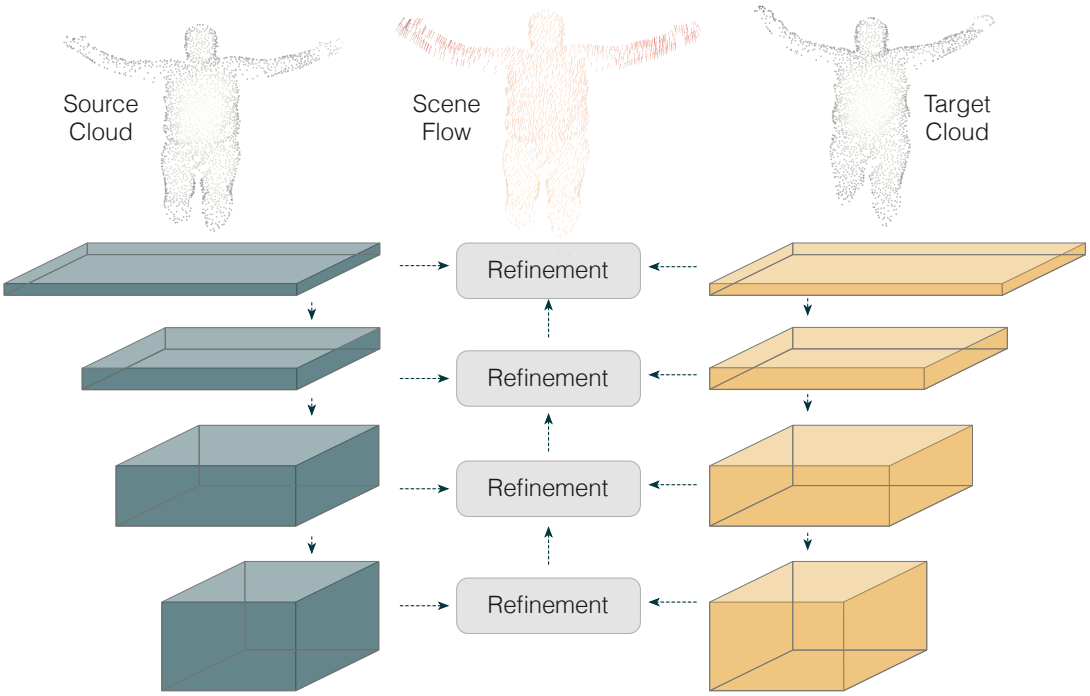


Fig. 4.5: The overview of the PointPWC network. In the hierarchical architecture, features are generated for the source and the target cloud on different resolutions. Starting from the sparsest resolution, the features are used to get a first sparse flow estimate. At each more dense resolution, the flow is refined and upsampled until the input resolution flow estimate is computed.

transition to 3D point clouds poses some problems because the data are unstructured and therefore had to be solved by the authors. Conceptually, point-to-patch costs are first computed by finding the neighboring points in P_t for each point $p \in P_s$. The features of these points are concatenated with the direction vectors and fed into a PointConv to learn the costs. Subsequently, these costs are further aggregated in the respective neighborhoods in P_s using PointConv to learn the final patch-to-patch costs. In the flow predictor, the costs estimated by the cost volume layer, together with the features of the source cloud, the upsampled flow of the previous layer and the upsampled features of the scene flow predictor of the previous layer are fed into multiple layers of PointConv and MLP to determine the flow of one level.

Our selection of methods consists of two iterative and two one-shot methods, whereby the methods presented differ partly in the definition of the problem to be solved, but especially in the network structure. All four methods process features that are first extracted from the point clouds. In FLOT they are used to match two graphs, while within PV-RAFT a gated activation unit is used to iteratively approximate the flow.

Table 4.1: Properties of the created scene flow datasets. The average object size is defined as the maximum possible distance between two points in the source point clouds. The estimated deformation percentage uses the deformed adaptations of the datasets presented in Section 4.3.5 to relate the length of the deformation vector to the length of the entire flow vector.

Name	Subset	N frames	∅ Object size [mm]	∅ Flow length [mm]	∅ Deformation [%]
nyu	train	72 558	240 (± 21)	8.87 (± 8.57)	78.7
	test	8150	220 (± 26)	14.17 (± 13.62)	84.0
nyu synth	train	72 558	237 (± 19)	8.94 (± 8.60)	78.3
	test	8150	217 (± 23)	14.25 (± 13.61)	83.5
dynamic faust	train	35 916	1674 (± 91)	65.37 (± 38.28)	68.9
	test	4659	1817 (± 81)	64.02 (± 35.37)	69.1
surreal	train	19 030	1690 (± 173)	91.02 (± 136.71)	58.4
	test	6770	1685 (± 162)	154.51 (± 174.05)	67.0

In PointPWC, features at different resolutions are extracted to determine a refined estimate of the scene flow. Thus, the methods represent very different starting points for possible future developments of new methods and are therefore particularly suitable for a detailed analysis.

4.3 Experiments and Results

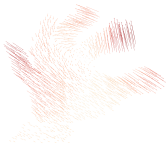
4.3.1 Datasets

The creation of scene flow datasets is generally a process that requires significant effort because 3D motion fields contain a lot of information that is difficult to capture manually. For this reason, large synthetic datasets are used for training models to reduce the required effort. However, compared to other research areas such as optical flow estimation, there are fewer datasets available for training and evaluation. To overcome the limitation of available scene flow data with realistically moving and deformed objects, we will conduct our experiments on datasets that we specifically prepared for scene flow estimation. The datasets are based on human hand data from NYU [Tompson et al., 2014] and human body data from Dynamic FAUST [Bogo et al., 2017] and SURREAL [Varol et al., 2017]. For an overview of the adapted datasets see Table 4.1 or for visualization of examples see Fig. 4.6.

NYU. Tompson et al. [2014] captured RGB-D data of hands from three different views and fitted a hand model into the point cloud to obtain 42 3D positions of keypoints on the hand that characterize its pose. Besides those keypoints, the dataset provides real and synthetic depth images for each frame of a sequence. Therefore, we created two scene flow datasets, one with real hand data and the other one using synthetic data.

(a) NYU real source

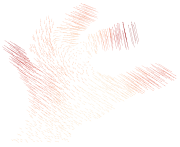
(b) NYU real target



(c) NYU real flow

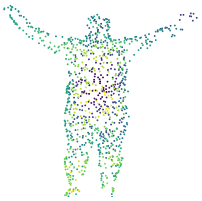
(d) NYU synthetic source

(e) NYU synthetic target



(f) NYU synthetic flow

(g) Dynamic FAUST source



(h) Dynamic FAUST target

(i) Dynamic FAUST flow

(j) SURREAL source

(k) SURREAL target

(l) SURREAL flow

Fig. 4.6: Exemplary representation of a selected frame from the respective datasets. The source (left) and target (middle) point clouds are shown, as well as the ground truth scene flow (right).

First, we use the camera parameters to convert the depth images of the sequences into point clouds and use the positions of the hand keypoints to restrict the point cloud to the area of the hand. Farthest-point sampling is then used to limit the number of points to 1024. Subsequently, point cloud pairs are formed from the video sequences by assigning to each frame the point cloud of the fifth closest frame in time. We have determined this time interval empirically to make the scene flow problem neither too easy nor too difficult. Let the ground truth positions of the keypoints K_s and K_t be given for each pair of point clouds P_s, P_t. Then we define the sparse flow of the keypoints $F_K = K_t - K_s$ and determine the flow of the pair for $\forall p \in P_s$ as:

$$F(p) = \left(\sum_{k \in \mathcal{N}(p, K_s)} \|k - p\|_2^2 \right) \sum_{k \in \mathcal{N}(p, K_s)} \frac{F_K(k)}{\|k - p\|_2^2}, \tag{4.5}$$

with $\mathcal{N}(p, K_s)$ being the neighborhood of p in K_s. We empirically choose a size of three for the neighborhoods. In total we obtain 14 sequences for the test set and 34 sequences for the training subset, each with varying numbers of point cloud pairs but resulting in a total of 8150 pairs for testing and 72558 for training.

Dynamic FAUST. The Dynamic FAUST dataset [Bogo et al., 2017] contains full 3D scans of moving humans, as well as dense ground truth correspondences that are generated by fitting an SMPL model [Loper et al., 2015] to the 3D scans. The data was captured at 60 fps over time with ten subjects that had to perform up to 14 sequences. The subjects are divided into five female and five male subjects. We chose a male subject that performs 14 different sequences as a test subset and all remaining subjects as the training set. In the first step, we use farthest-point sampling to extract subsampled point clouds of a fixed size of 4096 points. Following the data generation protocol of NYU, we built point cloud pairs by assigning each frame the point cloud of the fifth closest frame in time and used equation (4.5) to estimate the 3D motion field. We chose a neighborhood size of one because the motion field of the SMPL model is much more densely populated than the keypoints at NYU. To make the dataset more challenging, we randomly sample the source and the target point clouds to 1024 points within our experiments, to avoid frames where $P_s + f_t$ is the permuted version of the target cloud P_t. In addition to this, we translate and rotate P_t to increase the average flow length.

SURREAL. Varol et al. [2017] created the synthetic humans for real tasks dataset, in which depth images are synthetically rendered from sequences of human motion capture data by using an SMPL body model to ensure realism. In addition to the depth images the dataset provides the ground truth pose parameters of the SMPL models. The comprehensive dataset offers 55001 clips within 1964 sequences containing 115 different subjects for a training subset and 12528 clips within 703 sequences containing 30 different subjects, whereby every clip contains mostly 100 frames. To keep the

training times of the networks low, we limit ourselves to the first clip per valid sequence for both training and test subset and also limit the clips to 15 temporally contiguous frames. Since not every clip has 15 temporally contiguous frames with a sufficiently large point cloud, because the subject was outside the field of view of the camera, we obtain a total of 1903 clips for the training subset and 677 clips for the test set.
Similar to NYU, we use the camera parameters to generate 3D point clouds from the depth images and use farthest-point sampling to limit the number of points to 1024. We then define ten point cloud pairs per clip by assigning the point cloud of the fifth closest frame in time to each frame. For each pair, we use the corresponding SMPL pose parameters provided by the dataset to generate the registered 3D vertices of the body model, which then in turn can be used as keypoints for Eq. 4.5 to estimate the scene flow, where we again choose a neighborhood size of one.

4.3.2 Implementation details

The training and evaluations were run on a PC with Intel® Xeon® CPU E5-2637 v4 with 3.50GHz and a GeForce® GTX 1080 Ti GPU. In the training of the methods we used a point cloud size of 1024 for all datasets, utilized the Adam [Kingma et al., 2014] optimizer with an initial learning rate of 0.001, and roughly followed the procedure of the respective published works. In general, the selected methods are easy to train and require only a few data-specific parameters. Since we are particularly interested in dataset-independent estimation of the scene flow and to ensure a fair comparison, we have refrained from fine-tuning all hyperparameters specifically for the datasets. However, since the units and the scaling of the objects in the new datasets differ, we adjusted the size and scaling-specific hyperparameters such as search radii or distance limits.

For FLOT used a batch size of four on a total of 40 epochs. Also, we performed one iteration of the Sinkhorn algorithm for every dataset. Within the transport module, we chose a maximum transport distance of 200 mm for the NYU datasets and 1200 mm for Dynamic FAUST and SURREAL.

The PV-RAFT network was trained using a batch size of one for 20 epochs for the backbone and iterative module and the refinement module for another 10 epochs. We chose a cube resolution of three and set the truncation number of the correlation field to 512. However, due to the different scalings of the objects we built the level pyramid using a base radius of 25 mm for NYU and 150 mm for Dynamic FAUST and SURREAL.

While the authors of the PointPWC used only a part of the training set first and then fine-tuned the model on the whole set to speed up the training, we trained the model for all 800 epochs on the full training sets with a batch size of 8 for all datasets.

Since the architecture of FlowNet3D requires setting the radii in the four set conv, the flow embedding and the three set upconv layers, we have set these dataset-specific

to (5, 10, 20, 40, 100, 24, 12, 6) for NYU and to (30, 60, 120, 240, 600, 144, 72, 36) for Dynamic FAUST and SURREAL.

As the choice of parameters is crucial for CPD, we use a random selection of frames of the training data to find an appropriate set of parameters for the specific datasets. We therefore empirically set $w = 0$, $\lambda = 0.2$ and $\beta = 100$ for both NYU datasets, $w = 0$, $\lambda = 0.03$ and $\beta = 400$ for Dynamic FAUST and $w = 0$, $\lambda = 0.03$ and $\beta = 500$ for the SURREAL dataset.

4.3.3 Metrics

The *endpoint-error* (EPE) is defined as the average Euclidean distance between the predicted 3D flow vectors and the ground truth flow. Although this metric gives a good overview of the general quality of the scene flow estimation, it does not show how many frames were processed particularly badly (outliers) or particularly well (successful) by the network. *Accuracy* addresses this point by describing the proportion of successfully estimated frames, where successful means that the EPE of all points is less than a given threshold. Most existing works [Jin et al., 2022; Puy et al., 2020; Tishchenko et al., 2020; Wei et al., 2021; Wu et al., 2019] use the three metrics *ACC Strict*, *ACC Relax* and *Outliers* for their evaluation. In the following we explain why these metrics are not ideally suitable in general but only for specific datasets.

The goal of the metrics is to find a characteristic value for different accuracy gradations as well as for particularly bad predictions of the methods. According to this objective, ACC Strict is defined as the fraction of points with an EPE $< 5\,\text{mm}$ or a relative error $< 5\%$, ACC Relax is the fraction of points with EPE $< 10\,\text{mm}$ or a relative error $< 10\%$ and the Outliers are defined as the fraction of points with an EPE $> 30\,\text{mm}$ or a relative error $> 10\%$.

However, requiring a threshold for the metrics makes comparison for many different datasets much more difficult. For particularly difficult datasets, the thresholds for ACC Strict and Relax must be adjusted to be meaningful for the dataset itself, but you lose comparability to simpler datasets. Another problem with the metrics is linking the EPE condition to the relative error by an or. This can cause points to be classified as successfully detected in the ACC Strict metric with an EPE $< 5\,\text{mm}$, but at the same time as an outlier with a relative error $> 10\%$. This behavior can be observed in particular in the datasets we present, such as Tab. 4.2 shows.

Both FLOT and PV-RAFT show high accuracy with an ACC Strict value of more than 85% for the real hand data from the NYU dataset, but the percentage of outliers is just as high with 85% as well. We justify this by the local deformations in the hand, while a large fraction of the hand experiences little motion and thus has very small flow vectors. For the parts with short flow vectors, the relative error is high, and this effect is amplified when noise occurs in the data generation.

Table 4.2: The ACC Strict, ACC Relax and the Outlier values on the presented datasets. All metrics were averaged over the frames of the respective test sets.

		NYU real	NYU synth	Dyn. FAUST	SURREAL
ACC Strict [%] ↑	**CPD**	75.64	87.84	11.74	15.98
	FlowNet3D	77.96	91.87	04.79	12.85
	FLOT	**85.51**	97.45	42.33	**49.89**
	PointPWC	64.75	83.22	**54.67**	27.61
	PV-RAFT	85.12	**97.49**	39.41	19.12
ACC Relax [%] ↑	**CPD**	93.07	97.05	38.34	28.85
	FlowNet3D	93.81	97.74	24.32	41.99
	FLOT	**96.83**	**99.37**	**78.42**	**79.09**
	PointPWC	86.32	94.91	76.69	52.25
	PV-RAFT	96.69	99.12	72.86	51.39
Outliers [%] ↓	**CPD**	88.59	75.51	87.68	93.76
	FlowNet3D	93.29	67.03	91.99	84.47
	FLOT	**84.32**	50.56	52.85	**54.83**
	PointPWC	98.31	91.40	**42.54**	79.40
	PV-RAFT	85.51	**47.77**	56.56	64.00

A look at the ACC Relax values for the NYU synth hand data shows that the thresholds for these metrics are inappropriate. All methods have values above 90% and in the case of FLOT and PV-RAFT the difference to the ACC Strict values is less than 2%. For this dataset, the metric has little significance. Due to the reasons described above, the ACC Relax and Outlier metrics are not included in our evaluations. Nevertheless, in order to be able to make a comparison with existing benchmarks, we also evaluate the ACC Strict metric in our experiments.

Although EPE is a well-established metric for evaluating scene flow, it combines both the similarity of the transformed point cloud to the target point cloud and the correspondence between all points in the source and target cloud within one value. This means a high EPE does not indicate whether the target point cloud is poorly matched or if it is well mapped but the correspondences were poorly determined. Therefore we additionally provide similarity and correspondence measures separately as follows:

$$\text{dis} = \frac{1}{N_t} \sum_{p \in P_t} \min_{\tilde{p} \in P_s + f} \|p - \tilde{p}\|_2 \tag{4.6}$$

$$\text{cor} = \frac{1}{N_s} \sum_{i=1}^{N_s} K(I({P_s}^i + f^i, P_t), I(P_s^i + f_{gt}^i, P_t)) \tag{4.7}$$

$$K(i,j) = \begin{cases} 1, & \text{for } i = j \\ 0, & \text{for } i \neq j \end{cases} \tag{4.8}$$

$$I(p,P) = \arg\min_{\tilde{p} \in P} \|p - \tilde{p}\|_2^2. \tag{4.9}$$

The *dissimilarity* measure (dis) is defined as the mean Euclidean distance of all points of the target cloud to the respective closest point of the transformed point cloud, which corresponds to a one-sided chamfer distance. We do not use the usual chamfer distance here, as this would also be high for good solutions if the target cloud contains structures that are hidden in the source point cloud. The *correspondence* measure (cor) on the other hand is defined as the fraction of points from the transformed source cloud that have the same nearest neighbor in the target point cloud as the point cloud transformed with the true flow.

4.3.4 Performance

For a comprehensive summary of the presented methods, we analyzed their performance regarding two aspects. The accuracy, which is the main component of previous evaluations, besides the runtime of the trained networks. Therefore we trained all models on the four contributed datasets for scene flow with realistically deforming objects. Before the advent of deep learning-based methods, some methods that do not require prior learning were also established. For the broadest possible comparison, we therefore also include the frequently used Coherent Point Drift (CPD) for non-rigid point registration [Myronenko et al., 2010] in our evaluation.

Runtime. In real-world applications, the runtime of algorithms is one of the most important and limiting factors, along with accuracy. Although various previous publications have examined the runtime on the KITTI dataset [Menze et al., 2015a] with 4096 points per cloud [Ding et al., 2022; Hur et al., 2020, 2021], a more extensive investigation is missing. In general, the runtime of the networks for estimating the scene flow is strongly dependent on the sizes of the input point clouds. Therefore, we ran the models on the Dynamic FAUST dataset, but with varying size of the source and target point cloud. The results of the evaluation are visualized in Fig. 4.7. It can be seen that the runtimes scale quite differently with the size of the point clouds. For small clouds, FLOT is the fastest network, but it is caught up by FlowNet3D due to the strong scaling, which we deduce to the fact that FLOT calculates all-to-all

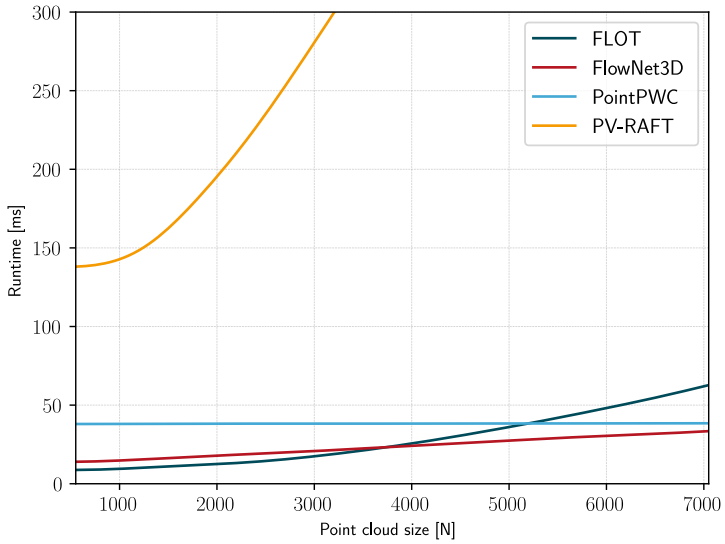


Fig. 4.7: Shown are the runtimes of the presented methods, depending on the size of the input point clouds. The evaluation was performed on the Dynamic FAUST dataset, which was scaled down and up for this scenario. The runtimes were averaged for better visualization.

correspondences. As the size of the point clouds increases, this becomes much more complex than the neighborhood correlations of FlowNet3D. Even if the runtime curve of FlowNet3D increases slightly with the increase in the number of points, at the end of the experiment it remains the fastest method due to the lower complexity of the network. PV-RAFT has consistently the highest runtime, which is due to the complexity of the method. What is striking in this experiment, however, is the almost constant runtime of PointPWC. Through the proposed cost volume layer, 3D convolutions or the generation of correlation volumes, which are much more computationally intensive, can be avoided and thus result in an architecture that is almost independent of the size of the point clouds. With a further increase in the number of points, PointPWC will also catch up with FlowNet3D and thus represents the fastest method for large point clouds.

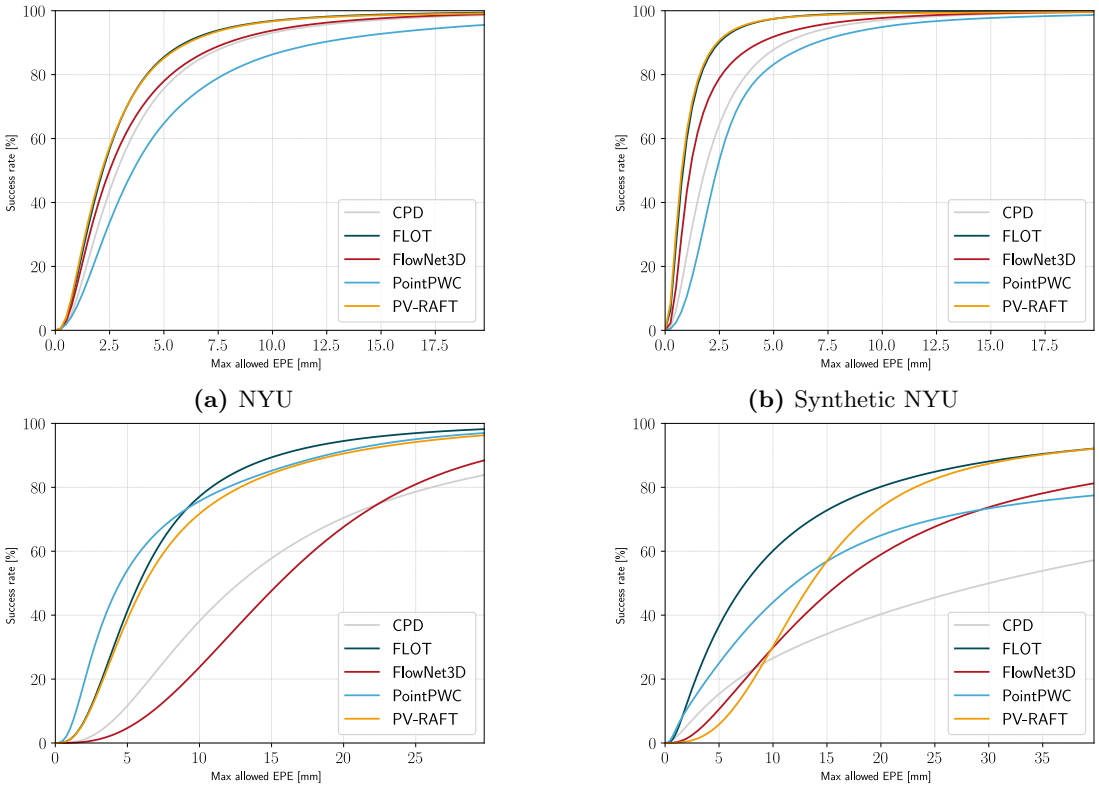
Accuracy. Next, we examined the accuracy of the presented methods. Here, the networks were run on the respective test subsets of the scene flow datasets and the results were evaluated using two presented metrics.

Table 4.3 shows the three metrics of the evaluated methods on the four datasets. It is noticeable that the EPE is generally lower for the two NYU datasets, which is due to the much lower average length of the flow compared to the other datasets. Both PV-RAFT and FLOT achieve very low EPE on the real and synthetic hand datasets. The non-learning-based CPD method achieves a slightly worse EPE than the

Table 4.3: The endpoint-error (EPE), dissimilarity, correspondence and ACC Strict measures of the evaluated methods. All metrics were averaged over the frames of the respective test sets of the presented datasets.

		NYU real	NYU synth	Dyn. FAUST	SURREAL
EPE [mm]↓	**Initial**	14.17	14.25	64.02	154.51
	CPD	4.12	2.73	19.32	92.30
	FlowNet3D	3.76	2.03	17.91	30.20
	FLOT	**3.07**	**1.28**	7.82	**15.90**
	PointPWC	5.77	3.56	**7.78**	54.13
	PV-RAFT	3.12	**1.28**	9.10	22.07
dissimilarity [mm] ↓	**Initial**	8.18	7.98	41.05	94.37
	CPD	2.99	2.46	24.01	52.61
	FlowNet3D	3.33	2.60	23.31	22.21
	FLOT	**2.95**	**2.33**	**21.02**	**14.42**
	PointPWC	3.65	2.93	21.17	34.45
	PV-RAFT	3.08	2.39	21.70	19.22
correspondence [%] ↑	**Initial**	21.94	22.63	22.62	23.46
	CPD	40.92	55.19	65.87	34.05
	FlowNet3D	45.93	67.19	63.23	46.65
	FLOT	50.41	76.76	83.17	**66.29**
	PointPWC	35.51	49.81	**83.97**	52.09
	PV-RAFT	**50.87**	**77.65**	80.91	54.41
ACC Strict [%] ↑	**CPD**	75.64	87.84	11.74	15.98
	FlowNet3D	77.96	91.87	04.79	12.85
	FLOT	**85.51**	97.45	42.33	**49.89**
	PointPWC	64.75	83.22	**54.67**	27.61
	PV-RAFT	85.12	**97.49**	39.41	19.12

FlowNet3D with 4.12mm on real and 2.73mm on synthetic hand data but achieves a dissimilarity measure close to the best deep learning-based methods. Surprisingly, the error of FlowNet3D and CPD is lower than that of PointPWC. Since it is mentioned in the PointPWC publication among other things that it is difficult for the network to find correspondences in the presence of strong deformations, the assumption can be made that this comparatively large error is due to the strong deformation part of the datasets (see Table 4.1). A look at the correspondence measure confirms this thesis because, with the proportions of 35.51% on the real dataset and 49.81% on the synthetic dataset, PointPWC found by far the least number of corresponding points. For the Dynamic FAUST dataset, on the other hand, with 7.78 mm PointPWC has the lowest EPE along with FLOT (7.82 mm). Unlike the other datasets, which only have camera projections of the objects, the Dynamic FAUST dataset has complete scans of the bodies. Apparently, for the PointPWC network, this favors the search for



(a) NYU

(b) Synthetic NYU

(c) Dynamic FAUST

(d) SURREAL dataset

Fig. 4.8: The accuracy of the presented methods on the testing subset of the presented scene flow datasets.

good correspondences. PV-RAFT also performs well on this dataset with an error of 9.1 mm and only FlowNet3D and CPD stand out with the comparatively high error of 17.91 mm and 19.32 mm which was to be expected due to the low complexity of the FlowNet3D network and due to the many variations in the dataset which are difficult to model with a fixed set of parameters. On the last dataset, the endpoint errors of the methods differ greatly. With an error of 15.9 mm FLOT performs best, followed by PV-RAFT with an EPE of 22.07 mm and FlowNet3D with 30.2 mm. Surprisingly, PointPWC has a comparably high error of 54.13 mm. The SURREAL dataset has the largest averaged lengths of the flow vectors, which means that the source and target clouds are further apart and may also be additionally rotated. It can therefore be assumed that PointPWC cannot estimate the flow well if the target cloud has undergone a larger translation or rotation. The probabilistic method CPD was not able to achieve adequate results on this dataset.

A look at the accuracy (Fig. 4.8) allows us to draw conclusions about the number of outliers and successfully estimated frames of the methods. Also here it becomes

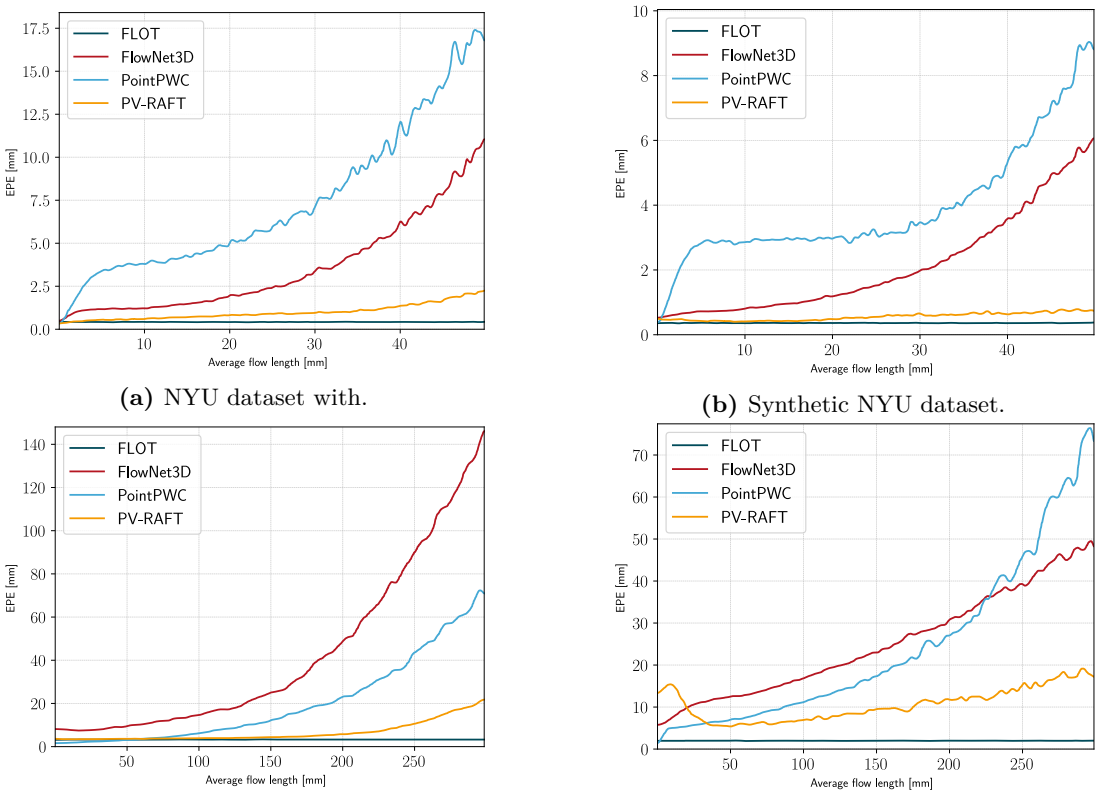
clear that FLOT and PV-RAFT achieve the most precise results on the real as well as on the synthetic NYU dataset and hardly differ from each other in quality. The results of PointPWC stand out somewhat, which at a threshold of 2.5mm successfully handles only 35% on the real data and 55% of the frames on the synthetic data. At the same threshold, the other methods achieve percentages of over 50% on the real data and over 80% on synthetic data. On the Dynamic FAUST dataset, on the other hand, PointPWC has the largest proportion of successful estimates up to a threshold of about 8mm, after which it is only caught up by FLOT. Both the pioneer method FlowNet3D and CPD are unable to produce accurate results on this dataset and their proportions of successful frames are far below the other methods. Also on the SURREAL dataset, the proportion of successful frames initially increases most rapidly for PointPWC as the threshold is increased, but is caught up by all three other methods due to the poorer processing of outliers. The proportion of extreme outliers above 35mm is almost identical for FLOT and PV-RAFT, but due to the higher proportion of successful frames at lower threshold values, FLOT performed best overall on this dataset.

4.3.5 Analysis of specific tasks

Now that we have examined the general accuracy of the scene flow estimations of the four methods, we will carry out a more detailed analysis to identify the strengths and weaknesses of each method. In general, the individual scene flow vectors can be divided into three components translation, rotation and deformation. Depending on the problem, the different components have different shares of the total flow vector. In order to examine how well the methods can solve the individual subtasks, we used the already trained models and evaluated them on the presented datasets, whereby the source point clouds are kept, but the target clouds specifically were adapted for the different tasks. Within all three adaptions, the resulting target point cloud was permuted.

In addition to accurately detecting the flow components, it is also important that the methods continue to produce accurate results on new data. Therefore, in the last part of this section, we investigate the generalization capabilities of the methods by evaluating them under domain gaps between training and testing data.

Translation. Especially for hand datasets, objects often experience translation in addition to deformation, as this is a large component of gesture. Even if a larger translation can be counteracted with suitable preprocessing, the networks have to detect and solve smaller translations. To investigate the presented methods with respect to this task, we processed the datasets so that they only contain translations. For this purpose, new target clouds were generated for all source clouds by shifting the source clouds by generated vectors. Both the direction and the length of these vectors were randomly generated, with an upper limit on the length depending on the dataset. Afterwards the already trained methods were evaluated on these adapted datasets by



(a) NYU dataset with.

(b) Synthetic NYU dataset.

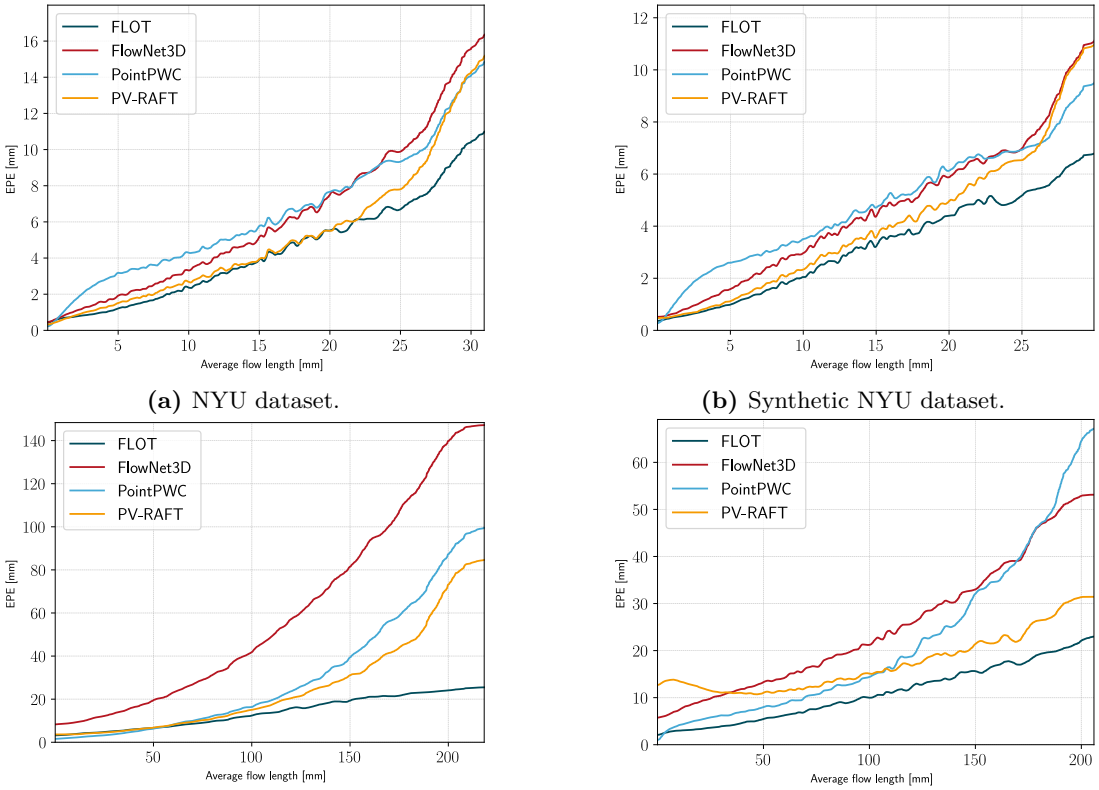
(c) Dynamic FAUST dataset.

(d) SURREAL dataset.

Fig. 4.9: Analysis of the presented methods on point clouds that are translated. The images show the averaged endpoint-error depending on the length of the translation.

computing the EPE, depending on the length of the translation vector. The results are shown in Fig. 4.9. Particularly striking is the constant error of FLOT on all four datasets. Even at high lengths of the displacement vector, the error remains almost constant. This can be attributed to the translation invariant PointNet++ modules, which are used in FLOT to generate features. Since the transport cost is defined by similarities of these features, the optimal transport model becomes translation-invariant. For the other methods, the error is usually also low for small displacements but often increases exponentially with an increase in the length of the displacement vector. An exception to this is PV-RAFT, for which the error on the synthetic NYU and the SURREAL dataset increases only linearly. PointPWC has the smallest error on all four datasets for extremely small displacements but has a rapid increase in error on the NYU datasets at a length of the vector of about 3mm. Subsequently, the error remains constant again and then goes into exponential growth with an increase in the length of the displacement and performs worst on the two datasets. On the Dynamic

FAUST dataset, PointPWC is best up to a displacement length of about 60mm, for larger displacements PV-RAFT and FLOT are somewhat more accurate.



(a) NYU dataset.

(b) Synthetic NYU dataset.

(c) Dynamic FAUST dataset.

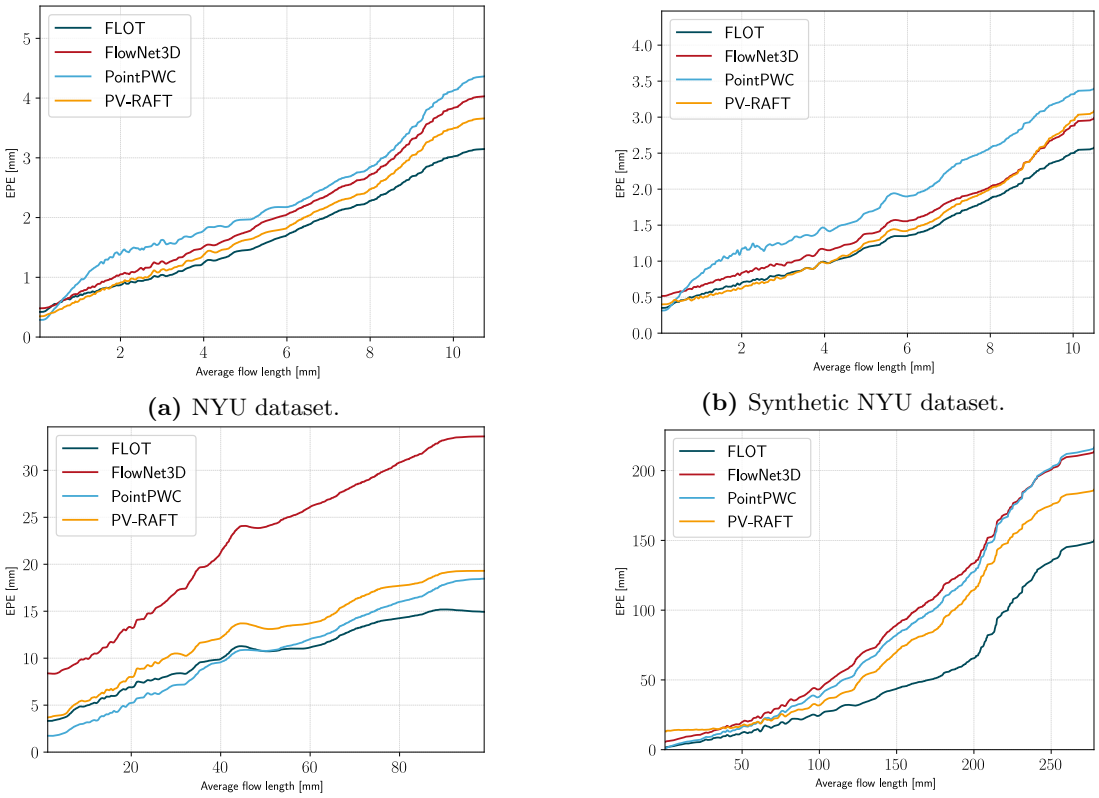
(d) SURREAL dataset.

Fig. 4.10: Analysis of the presented methods on point clouds that are rotated. The images show the averaged endpoint-error depending on the mean flow length that corresponds to the size of the rotation angle.

Rotation. In addition to displacement, the objects often experience a general rotation within a time interval. While this aspect is less important for the whole body, it is especially important for hand data. For this reason, we have examined the presented methods to see how well they can handle rotations of different degrees. Therefore, we generated new target clouds that correspond to a rotated version of the source cloud. First, a random normal vector was generated for each frame and defined as the rotation axis. Then, a random angle in the range between 0 and 30 degrees was determined and the rotation was performed accordingly. As in the previous subtask, the already trained models were evaluated on this data with respect to the EPE. Figure 4.10 shows the results for rotated point clouds, where the EPE is shown as a function of the average flow length, which in turn corresponds to the size of the rotation angle. On the real and synthetic hand data, the errors are generally close to

each other, with FLOT and PV-RAFT achieving slightly more accurate results than the two contenders. It is noticeable, however, that the error of all methods increases linearly with the length of the flow and thus with the size of the rotation angle. This is true up to a length of 25mm, after which the error changes to exponential growth. This allows the conclusion that this dataset contains many rotations, as already suspected, and that the models were trained accordingly. Particularly large rotations are not included in the training dataset and therefore the error increases much faster here. On the Dynamic FAUST dataset, PointPWC, PV-RAFT and FLOT initially show a quite similar accuracy, with a small lead of PointPWC. This suggests that PointPWC learns the rotations occurring in the training dataset best. For larger rotations that are not part of the training, the growth of the error for PV-RAFT and PointPWC turns into exponential growth and is therefore outperformed by the linear growth of FLOT. This leads to the conclusion that FLOT can best generalize rotations of the point clouds and therefore is able to find reasonable solutions even for unseen rotations. On the SURREAL dataset, PointPWC starts with the most accurate solutions for particularly small rotations but is quickly outperformed by PV-RAFT and FLOT, as the error increases exponentially with the increasing size of the rotation angle. FLOT gives the most accurate results on the dataset, followed at the end by PV-RAFT, which however has the most problems with smaller rotation angles.

Deformation. Since rotations and translations are already part of other datasets, deformation is of particular interest as the third subtask investigated. As in the two previous tasks, we have processed the source cloud to obtain a target cloud that only contains the transformation to be investigated. In order to focus on natural deformations in this evaluation, we did not generate them randomly but instead fell back on deformations that already occur in the dataset. To achieve this, the datasets require different pre-processing. For the real and synthetic hand datasets, we used the ground truth positions of the keypoints already provided by the NYU dataset to calculate deformations as naturally as possible. For this purpose, all keypoints of the target cloud that belong to the palm were mapped to the corresponding keypoints of the source cloud as best as possible using least square error, with the remaining keypoints undergoing the same transformation. We have selected the palm keypoints for the calculation of the mapping, as these are anatomically rarely subject to deformation and are usually transformed together. Since the correspondences of the keypoints are known, a deformation field can be calculated that maps the keypoints of the source cloud to the keypoints of the target cloud. Using nearest-neighbors interpolation, all points of the source cloud are finally deformed. The same process was carried out for the SURREAL dataset, with the addition that the SMPL model provides significantly more registered points and thus a denser deformation field can be determined. Furthermore, a global rotation and a root position are also specified as parameters, which simplifies the mapping of the keypoints. Since rotation and translation were artificially added by us in Dynamic FAUST, adaptation for this task is trivial.



(a) NYU dataset.

(b) Synthetic NYU dataset.

(c) Dynamic FAUST dataset.

(d) SURREAL dataset.

Fig. 4.11: Analysis of the presented methods on point clouds that are deformed. The images show the averaged endpoint-error depending on the mean flow length that corresponds to the amount of deforming.

The results of the already trained models for these deformed points clouds are visualized in Fig. 4.11. For the hand datasets, all four methods have a quite similar dependence of the EPE on the length of the flow and thus on the degree of deformation. Surprisingly, PointPWC has the lowest EPE for particularly small deformations, but as the degree of deformation increases, the EPE rises quite rapidly for a short time before it takes on the linear course of the other methods. Thus, for both real and synthetic hand data, FLOT followed by PV-RAFT produce the most accurate results overall. It is also worth noting that the error of all four methods shows variations at the same frames, which suggests that particularly difficult deformations pose problems for all four methods in a similar way.
On the Dynamic FAUST dataset, PointPWC achieves the most accurate results up to an average flow length of 50mm, above which FLOT has a slightly lower EPE. With an EPE that is always at least twice that of FLOT and PointPWC, FlowNet3D has problems finding reasonable solutions on this dataset. For the SURREAL dataset,

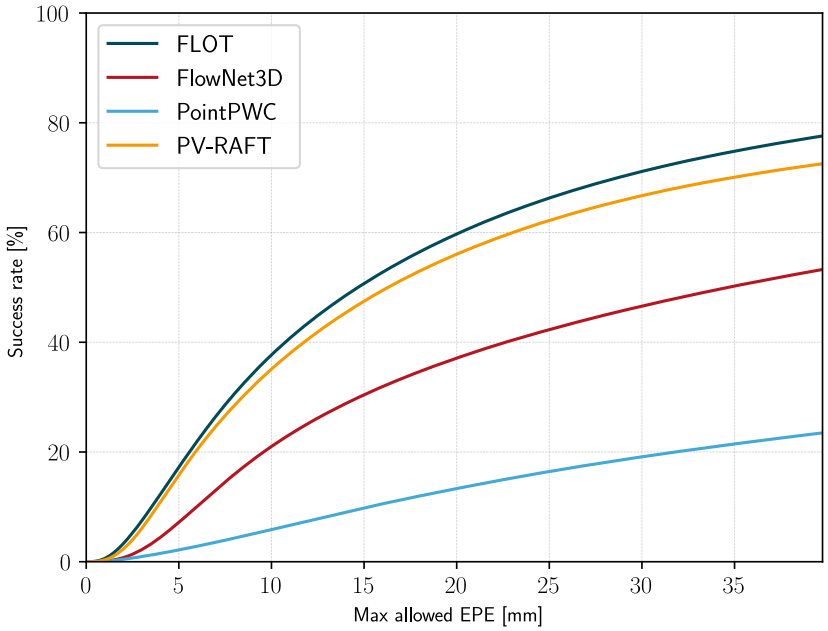


Fig. 4.12: The accuracy of the evaluated methods that were trained on the Dynamic FAUST dataset and evaluated on the SURREAL dataset.

FLOT consistently achieves the lowest EPE. The other three methods achieve similar results, with PV-RAFT tending to be slightly more accurate than the two counterparts.

Generalization. In order to analyze the generalization abilities of the methods, we used the networks that were trained on the Dynamic FAUST dataset and evaluated them on the SURREAL dataset. As Dynamic FAUST contains full-body scans while SURREAL includes camera projections of the point clouds, there is a domain gap that the networks must overcome. However, both datasets contain body point clouds that were generated using the SMPL model, therefore this task is demanding but solvable. A look at the accuracy in Fig. 4.12 shows that FLOT closely followed by PV-RAFT solves the generalization task with the highest accuracy. Both FlowNet3D and PointPWC have low accuracy in estimating scene flow on domain gap data. The impression that the two methods are not able to cope with the generalization task is confirmed by the three metrics EPE, dis and cor in Table 4.4. The EPE is very high for both methods, even higher than the mean scene flow length of the dataset for PointPWC with 171.0 mm. Furthermore, both methods find only few correspondences between the source and target cloud with less than 30%. Surprisingly, PV-RAFT and FLOT find almost equal correspondences in the point clouds, although the EPE for PV-RAFT is much higher with 72.8 mm compared to FLOT with only 41.0 mm.

Table 4.4: The Endpoint-error, dissimilarity and the correspondence measures of the evaluated methods that were trained on the Dynamic FAUST dataset and evaluated on the SURREAL dataset. For better comparability, the results of the in-domain training are shown again in brackets.

	↓**EPE** [mm]	↓**dis.** [mm]	↑**cor.** [%]
Initial	154.5	94.4	23.5
FlowNet3D	91.4 (30.2)	40.5 (22.2)	28.8 (46.7)
FLOT	**41.0** (15.9)	**14.9** (14.4)	**45.0** (66.3)
PointPWC	171.0 (54.1)	73.0 (34.5)	12.1 (52.1)
PV-RAFT	72.8 (22.1)	23.5 (19.2)	42.5 (54.4)

4.4 Discussion and Conclusion

Scene flow estimation on deformable objects is an extremely challenging problem in the area of motion analysis of a scene or object. In this paper, we have reviewed the main methods and trained and evaluated them on specially prepared datasets with deforming objects. This allows us to draw conclusions in the area of human hand and body tracking. Through a comprehensive analysis of the performance of the presented methods, we were able to gather important information about the current state of research and the main results can be summarized as follows.

1. When the data shows strong variations of translation in addition to deformations and rotation, an optimal transport approach in combination with modules that generate translation invariant features, such as a PointNet++, is perfectly suited to successfully solve this subtask. Since the transport costs for the mass of all points increase equally with translations the optimal transport model determines the scene flow very reliably and almost independently of the length and direction of the shift vectors.

2. If time is a limiting factor in the entire application, learning cost volumes is an excellent way to reduce the runtimes for calculating correspondences. In addition, the runtime becomes almost independent of the size of the incoming point clouds, which means that a precise estimate of the runtime can always be given even for variable sizes of point clouds.

3. The determination of correspondences between source and target point clouds can be performed very precisely with the existing methods. However, there is still a strong dependence on the quality of the available data. For example, only about 45% of the correspondences can be successfully determined for real hand data. If the data is generated synthetically, more than 70% of the correspondences are found.

For synthetic full-body scans, the methods even achieve successful correspondence determinations of over 80%.

4. Occlusions of body parts pose varying degrees of problems for the methods. If parts are missing due to the camera projection, PointPWC tends to perform somewhat worse compared to the other methods. However, if a complete scan of the object is available, as would be the case with a fusion of multiple sensors, PointPWC solves the problem with the highest accuracy.

5. The generalization abilities of the methods on deformed objects are very limited. None of the methods has managed to achieve an endpoint error comparable to that without this domain shift. Surprisingly, FLOT and PV-RAFT were able to find above 40% of the correspondences despite a much higher EPE and are thus not far from the specifically trained models, which are 66% for FLOT and 52% for PV-RAFT.

In conclusion, with new datasets and a new benchmark, we were able to gain significant information from state-of-the-art scene flow methods on real human data, revealing previously unknown strengths but also weaknesses. We are convinced that our presented datasets will be integrated into future benchmarks in order to monitor the progress of scene flow estimation in reality. Also, we think that our obtained information will have a great impact on the development of new methods with temporal information, especially tracking algorithms of real human data.

Chapter 5

Incorporating Temporal Information into 3D Hand Pose Estimation

After gathering crucial information on motion analysis in the previous chapter, the third methodological chapter aims to leverage the acquired temporal information for improved hand pose estimation. In this regard, we introduce a new network designed to extract temporal information from a sequence of estimated hand poses to generate enhanced pose outputs. Specifically, an existing scene flow network is optimized for hand skeleton movement and integrated into a temporal module for pose refinement. This temporal module, complemented with a spatial module that examines incoming poses for anatomical plausibility, forms the conceptual framework of the new network. Parts of this chapter have been accepted for publication in the proceedings of the *VISAPP 2024* conference.

5.1 Introduction

3D human hand pose estimation is becoming increasingly important especially in the fields of virtual and augmented reality, as it enables an intuitive way for a human-machine interaction and provides a realistic user experience [Buckingham, 2021; Voigt-Antons et al., 2020]. In this way, interaction possibilities such as the use of gesture language [Lenman et al., 2002; Nielsen et al., 2004; Quek, 1996] or object manipulation [Bowman, 2002; Buchmann et al., 2004; Tokgoz et al., 2003] can be brought into applications. The use of 3D point clouds has the general advantage of being more independent from the camera setup, for example the data of multiple depth sensors can be fused in a dense 3D point cloud [Hu et al., 2021] and thus be used.

Earlier approaches attached gloves or additional sensors to the user's hand to obtain precise data for estimating the hand pose [Ma et al., 2011; Stiefmeier et al., 2006; Wang et al., 2009] with the disadvantage that this also entails a restriction in the movement of a user. This led to the development of purely image-based methods for hand pose estimation, which advanced significantly with the progress of deep learning [Chatzis et al., 2020]. A successful but error prone approach is to predict the pose of a hand at an individual point in time [Hermes et al., 2022; Malik et al., 2019; Moon et al., 2018;

Oberweger et al., 2017; Poier et al., 2018; Rezaei et al., 2023; Wan et al., 2018; Zhang et al., 2020]. While performant, the largest remaining challenge for these methods is to deal with frequently occurring self-occlusions [Barsoum, 2016], which make it difficult to determine the hand pose precisely and negatively influence human-machine interaction.

The integration of temporal information is one way to remedy this problem, as it enables to propagate information from previous frames in which the presently occluded parts were visible. Different approaches to determine temporal correspondences in a scene already exist and are currently being actively researched. Scene flow methods analyze the motion of a scene given by a sequence of 3D point clouds and enable a more comprehensive understanding of the dynamics in the scene [Gu et al., 2019; Hur et al., 2020; Liu et al., 2019a; Mittal et al., 2020; Puy et al., 2020; Wang et al., 2020; Wei et al., 2021; Wu et al., 2020]. While these methods can accurately estimate point-wise correspondences, it remains unclear and has not been investigated yet how 3D pose estimation can benefit from scene flow.

We strongly believe that incorporating temporal information to pose estimation methods in the form of scene flow is a key to overcome the occlusion problem and achieve accurate pose predictions. Therefore we propose, to our knowledge, the first network that simultaneously estimates skeleton scene flow and 3D hand poses. Given the frame-wise predictions of an arbitrary pose estimator, the key idea is to analyze the motion of the bone structure between consecutive points in time with a scene flow module to improve the pose predictions of the estimator. To this end, we introduce a module, which adapts an arbitrary scene flow network to determine the skeleton motion and thus transfers previous bone motion knowledge to current time. This is done through a novel supervision strategy for scene flow networks, which defines a bone flow loss function to encourage skeleton-aware flow predictions. To increase generalizability, this module is complemented by a method, which learns to recognize spatial relationships of a hand and transfers a specific pose into a lower-dimensional pose space and constraining it to anatomical plausibility.
Particularly noteworthy is the independence of our method in the choice of networks used to determine the scene flow and to estimate the initial hand pose. Thus, it is agnostic to a specific implementation choice, makes use of pretrained models and can continue to benefit from the future development of both fields by replacing them with newer networks.

In the following, we briefly summarize the main contributions of this work.

- Proposal of a new network that combines the information in a 3D point cloud sequence to improve existing per-frame pose predictions. Once trained, the network can be used out of the box to improve new per-frame pose prediction methods.

- Proposal of a new module that learns spatial pose relations and transfers poses into a lower-dimensional space to increase the generalization capabilities of a network.

- Introduction of a method that optimizes existing scene flow prediction networks to specifically estimate skeleton-based movement.

- Extensive evaluation on a publicly available hand dataset to demonstrate the accuracy of our method and to obtain new insights in 3D hand pose estimation.

To the best of our knowledge we are the first to propose a network that integrates temporal information with scene flow for pose estimation using 3D point clouds only without resorting to external camera based information. Besides the convincing qualitative results of our method, we demonstrate its generalization capabilities in a comprehensive evaluation and show that we achieve significant improvements for 3D hand pose estimation.

5.1.1 Related Work

In this section we briefly summarize previous work in the area of incorporating temporal information into pose estimation, which is most related to this work. In general, the techniques for incorporating temporal information differ depending on the available input modality. Obviously, regular grid data such as RGB or depth images require different approaches than 3D point clouds.

3D pose from single frames. Besides the methods that employ depth maps [Moon et al., 2018; Oberweger et al., 2017; Wan et al., 2018; Xiong et al., 2019], there are also some existing methods that use 3D point clouds to estimate the hand pose frame-by-frame [Ge et al., 2018a,b; Hermes et al., 2022; Li et al., 2019a]. Ge et al. [2018a,b] use PointNet [Qi et al., 2016] layers in a hierarchical structure to capture local and global features to regress the hand pose. Hermes et al. [2022] extend the hand point cloud by support points to then extract features through graph CNNs [Wang et al., 2018] to determine the hand pose. Another method to extract features in point clouds are Residual Permutation Equivariant Layers as proposed by Li et al. [2019a]. In their work, point-wise poses are predicted and then the final pose is determined by voting.

3D hand tracking. Chen et al. [2022] presented a 3D hand tracking method in which the hand position is regressed for a current hand point cloud by updating the previous pose estimate. The previous pose is used to determine a global hand pose based on which a transformation of all point clouds in a canonical space is performed. This, in turn, is used to extract features and thus determine the current pose. Since this is a tracking method, the main idea is to transform an already found previous pose to the currently available data in the best possible way, which is different from our goal to create an improved pose estimate based on previous data. The difference in the objective is also reflected in the evaluation, since the authors use the ground truth pose in an initialization process to extract hand shape information and to obtain

a suitable initial pose to be tracked. Our method does not require initialization and therefore does not use ground truth information during evaluation.

Optical flow-based pose estimation. Other existing methods use the information extracted by optical flow to achieve an improved pose estimate [Alldieck et al., 2017; Arko et al., 2022; Liu et al., 2021; Pfister et al., 2015; Schwarz et al., 2012]. The computation of optical flow is a well-established technique to estimate the motion of pixels between consecutive frames in an image sequence. By analyzing the displacement of pixels, valuable information about the movement and velocity of objects within the scene can be obtained. Schwarz et al. [2012] use graph geodesic distances to establish correspondences between the 3D positions of human joints. This spatial information is combined with the temporal information that is extracted from intensity images using optical flow in order to refine the detected landmarks and thus to create a pose estimate by fitting a skeleton body model. Pfister et al. [2015] process the information of a whole sequence of color images. First, convolutional networks are used to predict heatmaps for the keypoints for each frame of the sequence. Using optical flow, the heatmaps are warped to the current frame and a subsequent temporal pooling generates the final heatmap from which the keypoint positions are inferred. A two-way improvement of pose and optical flow estimation is presented in the work of Arko et al. [2022]. First, the pose is used to fine-tune the optical flow estimation to better fit the human pose. With this optimized flow an improvement of the pose can be achieved in return. Since optical flow can only be estimated on 2D grid data, the application of these methodologies to 3D point clouds is not directly possible.

Scene flow estimation. One way to analyze the motion and dynamics of a scene given by 3D point clouds is to estimate the scene flow. Scene flow methods [Li et al., 2021a; Liu et al., 2019a; Puy et al., 2020; Wang et al., 2021; Wei et al., 2021; Wu et al., 2020] predict the correspondences of the points of two 3D point clouds captured at two successive points in time. However, it has never been used in 3D hand pose estimation and it is uncertain until now how the pose estimation can benefit from this information.

5.2 Methods

Pose tracking deals with the detection of specific keypoints, given a sequence of temporally consecutive visual data. In the 3D case, we assume the consecutive acquisition of the object to be tracked at $N \in \mathbb{N}$ time points $t = \{t_1, ..., t_N\}$ which leads to a sequence of point clouds $\mathcal{C} = \{C_1, ..., C_N\}$ with $C_i \in \mathbb{R}^{L_{C_i} \times 3}$. Furthermore, we define t_N as the current point in time, thus the times t_i with $i < N$ have already passed. The objective is to find the 3D positions of the keypoints $K_N \in \mathbb{R}^{L_K \times 3}$ at the current time t_N, using the information of the whole sequence:

$$K_N = \varphi(\mathcal{C}). \tag{5.1}$$

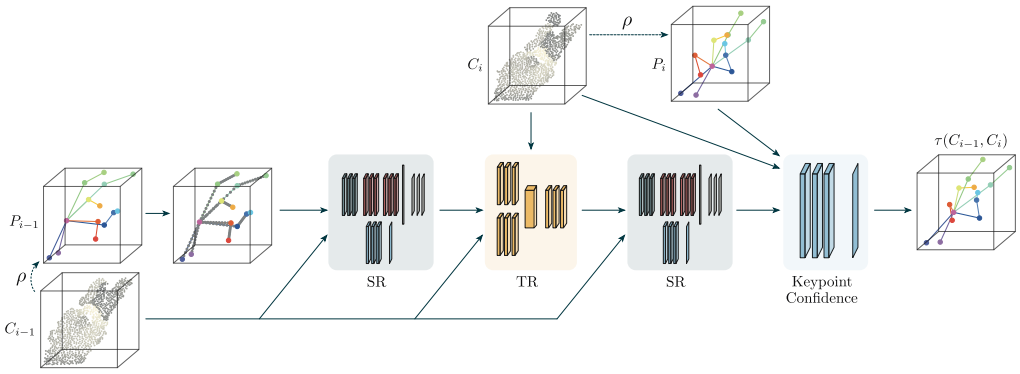


Fig. 5.1: Overview of the presented network. The previous estimate P_{i-1} is extend by bone points and fed into a Spatial Refinement module (SR) with the previous cloud. The result is fed into a Temporal Refinement module (TR) together with both the current and previous point clouds C_i and C_{i-1}. After a consecutive spatial refinement, the final pose estimate is generated in a weighting module.

As we want to improve the estimation of an existing pose estimator, we further assume that an already trained network ρ is available, which computes a prediction of the keypoints, given a single point cloud $\rho(C_i) = P_i$ with $P \in \mathbb{R}^{L_K \times 3}$. Our network always processes two consecutive frames, so by applying the model several times, formula Eq. 5.1 expands to:

$$\varphi(\mathcal{C}) = \tau(C_{N-1}, C_N) \tag{5.2}$$

with the recursion

$$\tau(C_{i-1}, C_i) = \begin{cases} \tau_P(\tau(C_{i-2}, C_{i-1}), C_{i-1}, C_i) & i > 2 \\ \tau_\rho(C_1, C_2) & i = 2 \end{cases} \tag{5.3}$$

$$\tau_P(\cdot, C_{i-1}, C_i) = \begin{cases} \tau_P(\tau(C_{i-2}, C_{i-1}), C_{i-1}, C_i) & i > 2 \\ \tau_\rho(C_1, C_2) & i = 2 \end{cases} \tag{5.4}$$

where $\tau_P(\cdot, C_{i-1}, C_i)$ expects the previous pose estimate as input, while $\tau_\rho(C_{i-1}, C_i)$ uses ρ to generate the pose estimates from the point clouds.

Figure 5.1 shows a general overview of our model. First, the previous pose estimate P_{i-1} is extended by points that are supposed to represent the bone structure by constructing a connection graph and defining equidistant points on the connections. The pose data is then converted into a lower-dimensional pose space in the *Spatial Refinement* (SR) module, preventing anatomically implausible poses. The constrained pose is transferred to the current point in time with additional input of the current

data within the *Temporal Refinement* (TR) module. After penalizing unrealistic poses again in another SR module, the pose to be output is determined by confidence-based weighting of the improved pose and the existing estimate. Conceptually, our network thus consists mainly of two modules, which are intended to establish spatial as well as temporal correspondences to improve a given pose estimate. Both modules will therefore be described in more detail hereafter.

5.2.1 Spatial Correlation Regression

Many of the existing hand pose estimation methods learn to determine keypoints without explicitly constraining the correlation between anatomically important points. To address this issue, we present a *Spatial Refinement* (SR) module which estimates an anatomically reasonable pose, given any point cloud C_i with an existing prediction P_i. Conceptually, the module is based on the principle of Principal Component Analysis (PCA), extended by a confidence measure in order to keep already good predictions. An overview is shown in Fig. 5.2. The upper branch of the visualization shows the generation of the anatomically reasonable pose estimate. First, the local features of the point cloud are extracted at the specific keypoints. As in PointNet++ [Qi et al., 2017], the neighborhood in the point cloud is determined for each keypoint and the features of the local regions are extracted using successive PointNet [Qi et al., 2016] convolutions. In a next step, all keypoints are themselves defined as neighborhoods for each individual keypoint and translated into new features through further successive convolutions. This ensures to also capture global features of the predicted pose. A subsequent abstraction layer outputs a single feature vector of the pose which is used within multiple linear layers to estimate weights for a given set of principal components. Multiplying these weights with a set of the most important principal components results in an anatomically plausible pose estimate Ψ. By using only a subset of the principal components the resulting pose is transferred into a lower-dimensional pose space to avoid unnatural hand pose outliers.

The lower branch of the visualization shows the generation of confidence values for the existing prediction. As in the upper branch, the nearest neighbors in the point cloud are determined for each keypoint in order to extract local features through subsequent convolutions. The local features are clamped into confidence values between 0.01 and 1 by a linear layer. At this point, we prevent confidence values of 0 to avoid getting stuck in local extrema. The confidence values are used for linear interpolation between the initially predicted keypoints P_i and the refined pose Ψ, resulting in a final pose estimate Φ.

By transferring the existing pose into a lower dimensional space learned by PCA, a restriction of the possible keypoint positions explicitly accompanies it. In the definition of the loss function to be optimized, we further restrict the space of possible poses and

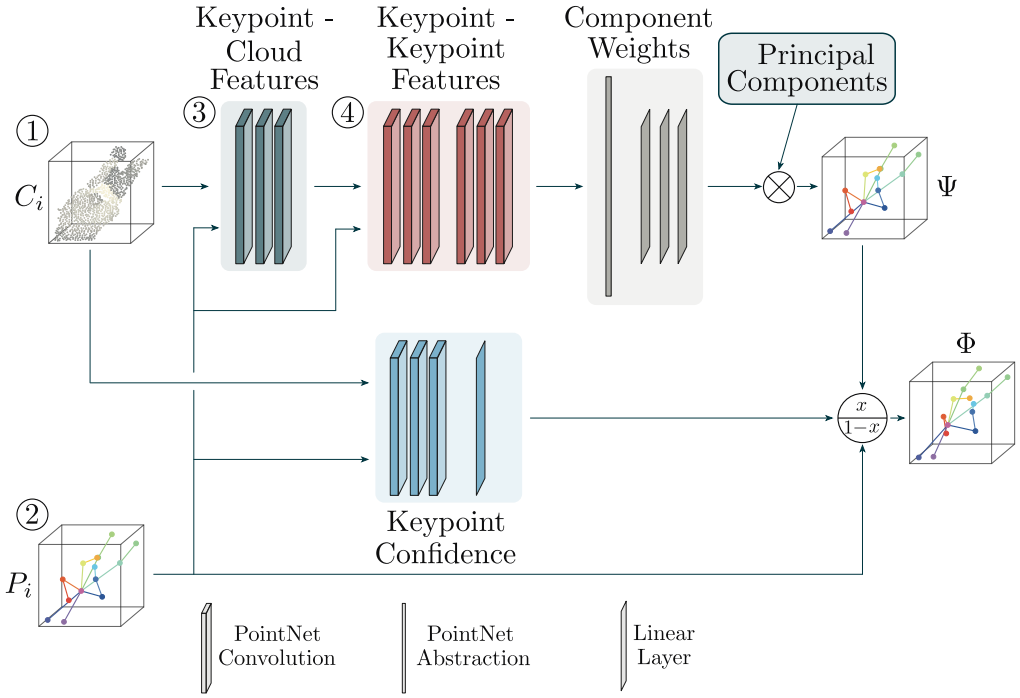


Fig. 5.2: Overview of the Spatial Refinement module. 1.) Given a point cloud C_i and 2.) the corresponding keypoint predictions P_i, 3.) cloud features are generated using each keypoints closest neighbors in the point cloud. 4.) The resulting features are extended by a new dimension that contains all other keypoint features, resulting in keypoint-to-keypoint features after multiple convolutions. In a last step these features are converted into weights which are multiplied with the principal components to output a suitable pose Ψ. This pose is linearly interpolated with the existing prediction using the weights from the Keypoint Confidence module.

require that the bone structure of the resulting pose is anatomically correct and that the 3D positions of the keypoints match the ground truth.

Assume the ground truth 3D positions of the keypoints $\hat{P} \in \mathbb{R}^{N_K \times 3}$ and further a set of index tuples $B = \{(h, j) \mid h,\ j \in 1, ..., N_K \text{ with } h \neq j\}$ for spanning a bone vector with $P_{B_i^h} - P_{B_i^j}$. Then we define

$$\mathcal{L}_{bones}(P, \hat{P}) = \frac{1}{|B|} \sum_{i=1}^{|B|} \|(P_{B_i^h} - P_{B_i^j}) - (\hat{P}_{B_i^h} - \hat{P}_{B_i^j})\|_2^2 \tag{5.5}$$

as a bone loss function which enforces that the bone vectors of the resulting pose correspond to the ground truth without taking into account the correct position in space. We further define

$$\mathcal{L}_{position}(P, \hat{P}) = \frac{1}{N_K} \sum_{i=1}^{N_K} \|P_i - \hat{P}_i\|_2^2 \tag{5.6}$$

as position loss function that ensures that the 3D positions of a given prediction P are correct. This results in the following loss function to be minimised for the network:

$$\begin{aligned} \mathcal{L}_{spatial}(\Psi, \Phi, \hat{P}) = \alpha \mathcal{L}_{bones}(\Psi, \hat{P}) + \beta \mathcal{L}_{position}(\Psi, \hat{P}) \\ + \beta \mathcal{L}_{position}(\Phi, \hat{P}), \end{aligned} \tag{5.7}$$

with the hyperparameters $\alpha, \beta \in \mathbb{R}$, that weigh the bone loss against the spatially constrained pose estimate Ψ and the interpolated resulting pose Φ.

5.2.2 Temporal Correlation Regression

One of the biggest difficulties with pose estimation is the occlusion of important parts of the object. This issue is particularly prominent in the determination of hand poses and constitutes a major challenge. We present the *Temporal Refinement* (TR) module to tackle this problem. If an important part of the object is occluded at the current time t_N, then we assume that the part was visible at a previous point in time. The aim of this module is to transfer the information from previous frames to the current time in order to compensate for the missing information of the hidden structure and to ensure a more precise determination of the pose. An overview of the module is visualized in Fig. 5.3. Given two successively captured point clouds C_{i-1} and C_i, as well as their corresponding pose predictions P_{i-1} and P_i, the module outputs an improved pose estimate Θ that contains the temporal information of both points in time. This is based on a scene flow network, which determines the correspondences between the previous pose and the current prediction, subject to a small error. To better represent the skeletal structure of the object, we extend the previous pose P_{i-1} by points that lie on the bone vectors. Subsequently, the additional points are combined with the point cloud and the scene flow is determined. The scene flow can then be used to compute a new pose estimate by adding P_{i-1} and the corresponding flow subset. To prevent existing correct estimates from being discarded, we use estimated confidence values as in the Spatial Refinement module to linearly interpolate between the existing prediction P_i and the new pose estimate and receive a temporal refined pose estimate Θ.

Bone flow. A crucial part of the module is that the scene flow network is optimized to recognize the movement of the skeleton in particular. We therefore propose a bone flow refinement process during the training, which is visualized by the dashed lines. Arko et al. [2022] already optimized optical flow using 2D body pose predictions

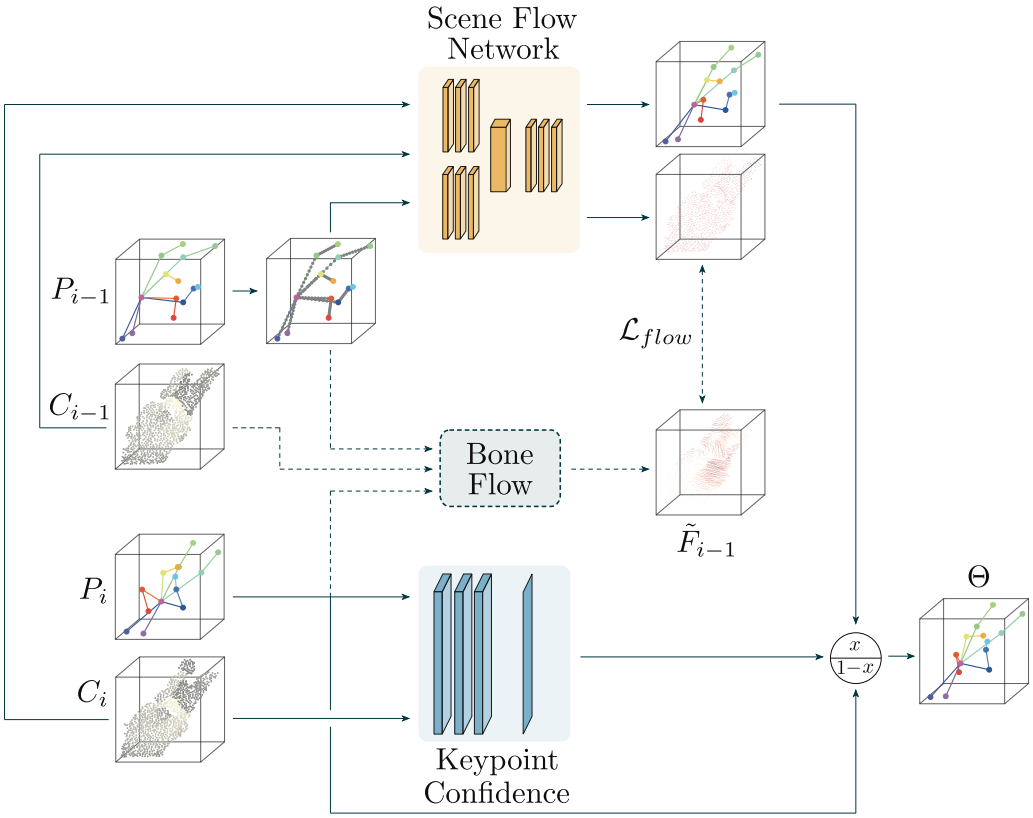


Fig. 5.3: Structure of the Temporal Refinement module that receives two successively captured point clouds C_{i-1} and C_i as input and the corresponding pose predictions P_{i-1} and P_i. The predictions of the earlier captured cloud are extended by points that lie on bone vectors and fed into the scene flow network, together with both clouds. The scene flow network in turn outputs the flow of the earlier point cloud and a new pose estimate which is the addition of the flow and the prediction P_{i-1}. The final pose estimate Θ is received by linear interpolation between the new pose estimate and prediction P_i, using the confidence values of the Keypoint Confidence module. During training, a refined bone flow $\tilde{F}_{i-1}$ is computed that integrates the movement of the bones of the object to be tracked. This process is visualized with dashed lines.

and computing a pixel-wise flow of a stick-man, which is integrated into the ground truth optical flow. Since we do not use camera data throughout the method and thus no projected 2D predictions are available, the method is not readily applicable. We therefore propose to extend the same methodology for 3D poses.

We assume that the predictions P_i can be used to construct a bone-like graph consisting of bone vectors. For each of these vectors we specify an anatomically suitable third keypoint to construct a coordinate system using the cross product. For two predictions P_{i-1} and P_i we obtain the sets of bone vectors $\mathcal{B}_{i-1}$ and $\mathcal{B}_i$ as well as the

corresponding coordinate systems. Let the point cloud C_{i-1} and the corresponding ground truth scene flow $\hat{F}_{i-1}$ be given. The following process produces a bone-motion specific flow $\tilde{F}_{i-1}$:

1. For each point of the cloud C_{i-1} find the closest bone vector in $\mathcal{B}_{i-1}$.
2. Translate each point to the current point in time i by transferring it into the coordinate system that belongs to the closest bone vector. Afterwards transfer it back to world coordinate system but using the coordinate system of the corresponding bone vector from B_i.
3. Compute the bone flow as vectors from the points towards their translated positions.
4. Linearly interpolate between the bone flow and the ground truth flow $\hat{F}_{i-1}$, where the distances of the points to the closest bone vectors are used to compute weights for interpolation.

To accommodate this revised flow in training, a suitable loss function must be selected. The requirements for a loss function to be optimized during training are that the flow network outputs the improved flow, as well as that the 3D positions of the keypoints of the resulting pose Θ are correctly determined. We therefore define the loss as:

$$\mathcal{L}_{temporal}(\Theta, F, \hat{P}_i, \tilde{F}) = \mathcal{L}_{position}(\Theta, \hat{P}_i) + \lambda \mathcal{L}_{flow}(F, \tilde{F}), \tag{5.8}$$

with the adjusting scalar $\lambda \in \mathbb{R}$, the network's pose estimate Θ, the estimated flow F, the ground truth keypoint positions $\hat{P}_i$ and the refined bone flow $\tilde{F}$. We further define the flow loss as:

$$\mathcal{L}_{flow}(F, \tilde{F}) = \frac{1}{|F|} \sum_{i=1}^{|F|} \frac{1}{3} \|F_i - \tilde{F}_i\|_1 \tag{5.9}$$

The data terms Eq. 5.7 and Eq. 5.8 of the presented modules lead to a convex optimization model to be minimized during training.

5.3 Experiments and Results

In this section, we provide a comprehensive analysis of our method by conducting a quantitative evaluation.

Dataset. To the best of our knowledge, the most widely used datasets for evaluating per-frame 3D hand pose estimation methods are NYU [Tompson et al., 2014], MSRA [Sun et al., 2015], and ICVL [Tang et al., 2014]. Since the MSRA dataset is less informative due to some erroneous annotations [Ge et al., 2018b] and a saturation of average joint errors in the ICVL dataset [Wan et al., 2018], we focus this evaluation

on the NYU dataset. Moreover, the NYU dataset contains more complex movements, making it the most challenging of the three datasets [Chen et al., 2018a]. It provides depth data of hands and ground truth 3D positions of 42 keypoints of the hand. However, we stick to the common practice for hand pose estimation methods and use only 14 keypoints [Ge et al., 2018b; Hermes et al., 2022; Wan et al., 2018]. For training, 72K frames in 34 sequences are available and the evaluation is carried out on 8K frames in 14 sequences. It should also be mentioned that we do not use the depth data at any time, but only point clouds with 1024 points extracted from it using farthest point sampling.

Implementation Details. For training of the modules we use the adjusting scalars $\alpha = \beta = \lambda = 1$, as well as an Adam [Kingma et al., 2014] optimizer with an initial learning rate of 0.001. The Spatial Refinement module was trained independently with an initial learning rate of 0.0001. Both modules were trained using a batch size of 16 for a total of 40 epochs. During the tests we focus on a single network for the calculation of the scene flow, in order to be able to exclude the factor of different quality grades in the scene flow determination. Therefore we follow the work of Hermes et al. [2023] and select a pre-trained FLOT [Puy et al., 2020] network to determine the scene flow, due to its ability to handle deformations well. Furthermore, we refer to their approach and use the 42 keypoint positions to calculate a ground truth scene flow. All trainings and evaluations were run on a PC with a GeForce® GTX 1080 Ti GPU.

Metrics. As a measure of error, we consistently use the widely used *Average End Point Error* (EPE) to ensure comparability with previous work. For a more precise statement, in some places we additionally provide the *Accuracy* measure, which is defined as the proportion of successfully determined frames whose EPE of all points is below a certain threshold value.

5.3.1 Impact of Scene Flow Refinement

In a first ablation study we investigate the influence of the proposed bone flow refinement process. We performed the following experiments on the NYU dataset, using only a subset of the training set. As per-frame pose prediction we used the results of the support points-based method by Hermes et al. [2022], that achieves an EPE of 9.44 mm.

Two successive captured point clouds as well as their corresponding pose predictions were always used as input to improve the 3D key point positions of the temporally more recent acquisition. Table 5.1 provides an overview of the results of the tests. The trained FLOT network was used as baseline to determine the scene flow of the concatenation of the point cloud and the predictions of the previous frame. The addition of the previous pose with the scene flow results in the new hand pose. This methodology achieves an EPE of 9.92 mm and thus even degrades the per-frame prediction by 4.6%.

Next, the scene flow network was optimized using the bone flow refinement process presented, while maintaining the same calculation of the hand pose. This yields an

Table 5.1: Results of various experiments with the presented bone flow refinement process. Training and testing was performed on the NYU dataset using the support points-based hand pose predictions. The trained scene flow network was chosen as baseline without further optimization.

Method	**EPE** [mm]
per-frame	9.48
common flow	$9.92_{\uparrow 4.6\%}$
refined flow	$9.65_{\uparrow 1.8\%}$
refined flow + distance weights	$9.32_{\downarrow 1.7\%}$
refined flow + learned weights	$\mathbf{9.07}_{\downarrow 4.3\%}$

error of 9.65 mm and therefore still results in higher errors as the per-frame predictions, but improves the baseline results by 0.27 mm.

Under the assumption that some keypoints can already be estimated well with the method, but others cannot, only keypoints of the new pose that improve the existing per-frame prediction should influence the resulting pose. In a next experiment, we therefore chose a linear interpolation between the two poses. The weighting is based on the distance of a keypoint to the nearest point of the point cloud. If no point of the cloud is close to a keypoint, so the part was occluded during the acquisition, the pose estimate determined through the scene flow is weighted higher. This methodology achieves an error of 9.32 mm and thus improves the per-frame prediction by a small amount. As a final experiment, we did not calculate the weights for the interpolation ourselves, but learned them during training. This resulted in the lowest error of 9.07 mm and improved the existing per-frame prediction by 4.3%.

The experiments have thus shown that the presented bone flow refinement process improves the usual scene flow for tracking a pose. Furthermore, they show that scene flow is well suited for tracking certain keypoints, especially occluded keypoints with missing information. If enough information is available for the position of a keypoint, per-frame pose prediction is superior.

5.3.2 Influence of the Temporal Distance

In a second ablation study, we investigate the dependence of a current pose prediction on the temporal distance of additional information. That is in particular, we are interested in how long a previous pose prediction can lie in the past in order to still have an additional value for a current estimate. We trained our proposed network on the NYU dataset using the predictions from the support points-based method again. During training, we use Eq. 5.2 - Eq. 5.4 to process a sequence of five frames, whereby the gradient calculation and thus the optimization of the network was only carried out

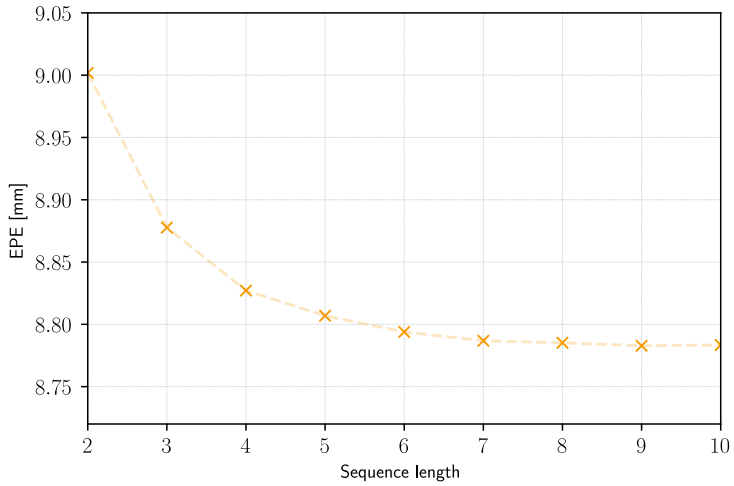


Fig. 5.4: Correlation of the number of frames used in our network to the EPE of the resulting hand poses.

for the temporally current frame. The evaluation was then performed using different sequence lengths N. The results of the experiment are shown in Fig. 5.4. First, the accuracy of the pose estimation increases with the use of more previous frames. From about seven frames used, the error of the pose predictions stagnates. From this we conclude that subsequent frames are too far away in time from the current frame and thus have only little to no information content for determining the pose.

5.3.3 Hand Pose Estimation on NYU Dataset

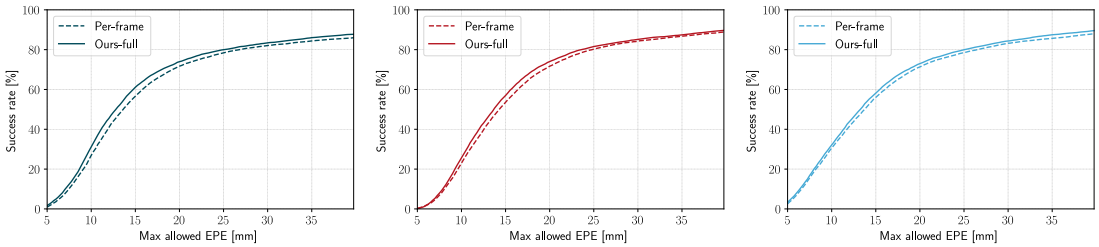
This experiment shows the different contributions of the presented modules of our network. We refer to per-frame predictions of different methods for a quantitative evaluation and analyze the behavior of our network in the presence of noisy pose estimates as inputs. Furthermore, we check the behavior of our proposed method when only little training data is available.

The different modules as well as the whole network were trained using the predictions determined by the support points-based network. Evaluation was additionally performed on the pose predictions of P2P [Ge et al., 2018b] and V2V [Moon et al., 2018], both of which yield state-of-the-art per-frame results. The EPE is listed in Tab. 5.2, based on consecutive execution of our network to process the information of five frames.

An individual analysis of the SR module shows that limiting to a spatial consideration does not significantly improve the pose estimation. For the support predictions only an improvement of 1.1% is achieved, for P2P the results are constant and for V2V even worse than the per-frame prediction. Bringing in temporal information has a greater

Table 5.2: EPE [mm] evaluation on the NYU dataset. The training is based on the predictions from the support points-based method, while evaluation was performed with the information of five frames using the predictions of various hand pose estimation methods. The best results are highlighted.

	Support	**P2P**	**V2V**
Per-frame	9.48	9.05	8.42
Ours-spatial	$9.38_{\downarrow 1.1\%}$	$9.05_{\downarrow 0\%}$	$8.44_{\uparrow 0.2\%}$
Ours-temporal	$\mathbf{8.76}_{\downarrow 7.6\%}$	$8.65_{\downarrow 4.4\%}$	$8.24_{\downarrow 2.1\%}$
Ours-full	$8.81_{\downarrow 7.1\%}$	$\mathbf{8.61}_{\downarrow 4.9\%}$	$\mathbf{8.07}_{\downarrow 4.2\%}$



(a) Support-based predictions. **(b)** P2P predictions. **(c)** V2V predictions.

Fig. 5.5: Accuracy measures of our proposed method. Shown are the results for the used per-frame predictions (dashed lines) and the refined results of our network using spatial and temporal information (solid lines).

impact and improves the per-frame predictions of the three considered methods. The combination of temporal information and spatial correlations (Ours-full) continues to improve the results for P2P slightly and noticeably for V2V, but marginally worsens the results of the support-based prediction. Since the estimated poses of P2P and V2V were not part of the training, we infer from the results that the SR module noticeably improves the already good generalization ability of the TR module. Furthermore, it can be assumed that the network adapted slightly too much to the predictions in the training data during the training. With p-values far below 0.05, a paired t-test proves that the difference between the EPE means of the per-frame predictions and the results of our network is statistically significant.

For a more detailed quantitative analysis, we calculated the accuracy measure for the obtained results, which is shown in Fig. 5.5. From the figure, it can be seen that our proposed method increases the number of successfully processed frames for all three considered methods of per-frame pose estimation. Furthermore, a runtime of over 40 fps is achieved for the processing of two point clouds, including the time for the scene flow network, which demonstrates high computationally efficiency and real-time capability.

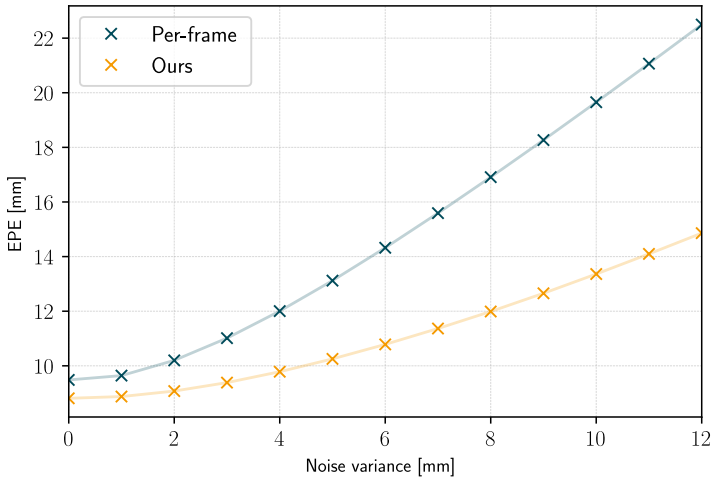


Fig. 5.6: Error evolution of our method on ingoing pose predictions corrupted with Gaussian noise of increasing variance. The upper curve shows the EPE of the per-frame predictions under the effect of the noise and the lower curve shows the results of our network with the noisy pose predictions as input.

Evaluation on corrupted input poses. In real scenarios, external influences can always cause noise in the pose detection. Since the incoming pose predictions are an important ingredient for our method, we perform an evaluation with corrupted incoming poses to investigate the behavior of our method with noisy data. For this purpose, we applied Gaussian noise with varying variance to the pose predictions of the method based on support points. We used the same network as in the previous experiment for further evaluations, which was trained with non-noisy pose predictions as input. The results of the evaluation are shown in Fig. 5.6. The upper curve shows the evolution of the EPE of the noisy per-frame predictions as the variance of the noise increases, and thus serves as a baseline for this experiment. The lower curve shows the EPE of our method, based on the respective noisy pose predictions as input. Both error curves increase linearly with the increase of the variance of the noise, but the slope of the curve of our method is significantly lower and thus achieves a smaller increase of the EPE as the baseline. Thus, our network achieves a reduction of the error by 10% for small variances and even more than 30% for large variances of the noise. These results show that our proposed network can counteract the error of noisy poses.

Training on a small subset. In a further experiment, we investigated the hypothesis that the Spatial Refinement module improves the generalization ability in more depth. Therefore, we repeated the previous experiment but limited the training data to the first 12 sequences, which reduced the number of training frames to under 19%. The EPE of the proposed modules are listed in Table 5.3. As before, the SR module

Table 5.3: EPE [mm] evaluation on the NYU dataset. The training was performed on reduced subset that contains only 19% of the frames, while evaluation was performed on the full test subset with the information of five frames using the predictions of various hand pose estimation methods. The best results are highlighted in bold.

	Support	**P2P**	**V2V**
Per-frame	9.48	9.05	8.42
Ours-spatial	$9.37_{\downarrow 1.2\%}$	$9.04_{\downarrow 0.1\%}$	$8.41_{\downarrow 0.1\%}$
Ours-temporal	$9.07_{\downarrow 4.3\%}$	$8.87_{\downarrow 2.0\%}$	$8.50_{\uparrow 1.0\%}$
Ours-full	$\mathbf{8.96}_{\downarrow 5.5\%}$	$\mathbf{8.75}_{\downarrow 3.3\%}$	$\mathbf{8.18}_{\downarrow 2.9\%}$

individually does not have much effect on the error of the 3D keypoint positions. The temporal refinement module suffers from the lack of available data and performs worse than before, even worsening the per-frame predictions of the V2V posenet with an EPE of 8.50 mm. The combination of temporal and spatial information, on the contrary, noticeably improves all existing per-frame predictions. We therefore conclude from the results that transferring the pose predictions into a lower dimensional hand space enables the TR module to produce an improved pose estimate even with little available training data. This ensures that our network can produce suitable pose estimates even on unknown data.

5.3.4 Comparison with HandTrackNet

In their work, Chen et al. [2022] introduce the HandTrackNet method for tracking hand poses using 3D point clouds. Given the previous hand keypoint positions and the current 3D hand point cloud, both inputs are transformed into a canonical space, whereas the transformation is calculated using an initial hand pose and the current 3D positions of the palm keypoints. Subsequently, features are extracted on the basis of which the previous pose is placed in the current hand point cloud. The aim of this process is to transfer a previous pose into a current 3D hand point cloud in the best possible way, whereas our method improves an existing pose estimate based on previous data. As these objectives differ from each other, a direct comparison is not readily possible. Nevertheless, we explore the differences in these two approaches in a joint evaluation.

For the evaluation, the already pre-trained HandTrackNet was used and tested on the NYU [Tompson et al., 2014] dataset. Since the procedure requires the position of six palm keypoints, we extended the 14 keypoints by the six required palm keypoints that are also part of the ground truth in the NYU dataset. The error calculation was carried out on the 14 keypoints as before.

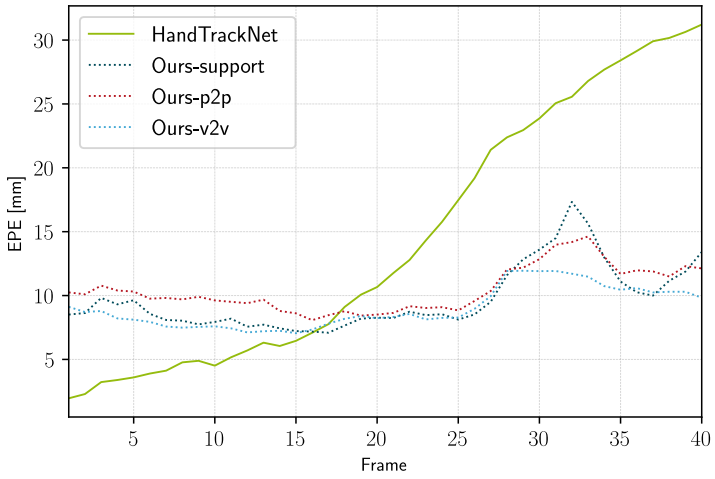
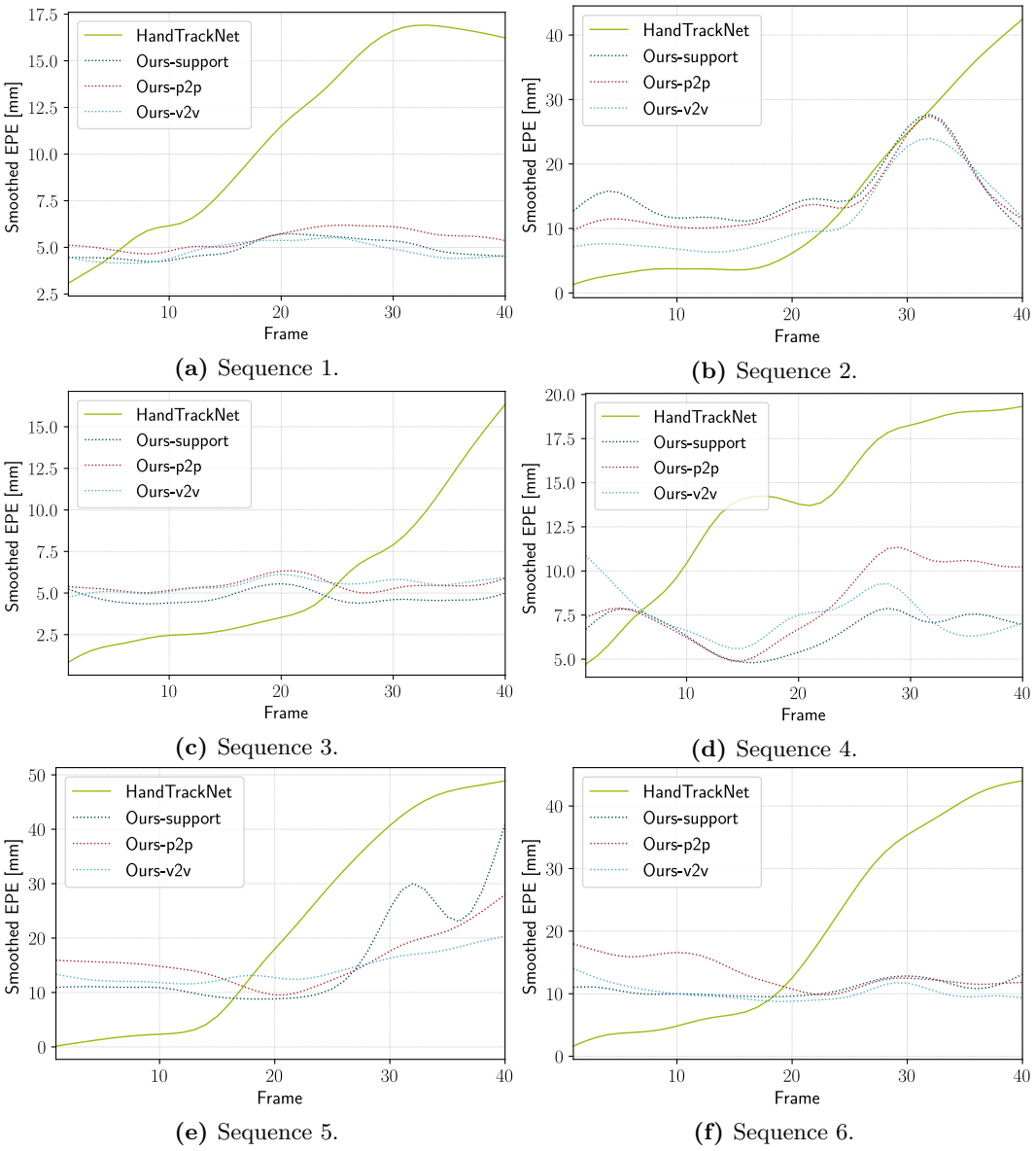


Fig. 5.7: EPE average over the six exemplary sequences of 40 frames. The pre-trained HandTrackNet was initialized with the ground truth pose. Our trained network was evaluated with the pose estimations predicted by the support point-based method, P2P and V2V.

As a first experiment, similar to the evaluation setup in Sec. 5.3.3 we had HandTrackNet estimate the hand poses based on five previous frames, using the pose predictions of the support points-based method as initial pose. This resulted in an EPE of 9.67 mm which is worse than the per-frame predictions with an error of 9.48 mm. Since the method only tracks a pose and does not improve it, this result was to be expected, because the initial pose was already faulty and the tracking accumulated the errors.

In a second experiment, we initialized the HandTrackNet on six exemplary sequences of 40 frames each with a ground truth pose and tracked them continuously. On the same sequences, we evaluated our from Sec. 5.3.3 trained network, using different pose estimators [Ge et al., 2018b; Hermes et al., 2022; Moon et al., 2018]. The per-frame EPE is shown in Fig. 5.8 for the six sequences, where the errors were smoothed for better visualization. For a quick overview, the average of the six sequences is additionally shown in Fig. 5.7. From the results can be seen that the error of HandTrackNet is small at the beginning because the initial pose corresponds to the ground truth, whereas our method has a higher error because it falls back on erroneous pose estimations. Since the tracking error is accumulated, the EPE for HandTrackNet generally increases continuously, whereas the error of our method varies due to the sequence, but a general growth is not recognizable. The progression in the second sequence is particularly interesting, as problems occur here with all methods from frame 25 onwards. The tracking cannot recover from these problems and the error increases steadily. Since our method always uses new pose estimates, the method recovers and the error decreases.



(a) Sequence 1. **(b)** Sequence 2.

(c) Sequence 3. **(d)** Sequence 4.

(e) Sequence 5. **(f)** Sequence 6.

Fig. 5.8: Smoothed per-frame EPE for six exemplary sequences of 40 frames. The pre-trained HandTrackNet was initialized with the ground truth pose. Our trained network was evaluated with the pose estimations predicted by the support point-based method, P2P and V2V.

5.4 Discussion and Conclusion

We propose a method for incorporating temporal information into hand pose estimation from a sequence of 3D point clouds. To the best of our knowledge, we are the first to

utilize scene flow for an improved pose estimation. In this context, we introduce a process that optimizes arbitrary scene flow networks for the application of pose estimation by focusing on the motion of the bone structure. This procedure is supplemented by a module that recognizes the spatial pose correlations and transfers existing poses into a lower-dimensional pose space in order to avoid unnatural outliers and to improve the generalization capability. In a comprehensive evaluation we demonstrate that the proposed method significantly improves existing frame-by-frame methods for 3D hand pose estimation. We further show that the network is able to reduce the negative influence of noisy pose predictions and that our network has such good generalization capabilities that it can significantly improve other pose estimation methods without further training.

In conclusion, we demonstrated the effective improvement of hand pose estimation, especially for occluded keypoints, through a sequence of past frames. As a future perspective, we are convinced that the methods we have presented for incorporating temporal information into hand pose estimation will yield even greater advantages when not only past frames are analyzed but also when information from future frames is taken into account.

Chapter 6

Hand Gesture Recognition with Predicted 3D Keypoints

In the final methodological chapter of this work, the previously developed methodologies for hand pose estimation are evaluated concerning potential applications. Specifically, we have chosen gesture recognition as a downstream application, representing a significant and practical domain that enables touchless human-machine interaction. Firstly, the current state-of-the-art in this field will be introduced, followed by a detailed description of a representative lightweight network for gesture recognition to provide a solid entry point into this field. In an extensive evaluation, we find that our gathered knowledge in the domain of hand pose estimation is rich in potential for gesture recognition. Utilizing the introduced techniques for estimating keypoint positions and using them to identify gestures demonstrates the favorable benefits of our pose predictions for gesture classification. During the experiments, comparability with existing evaluations is ensured to maximize the acquisition of new insights in this field.

6.1 Introduction

Human gesture plays a central role in communication, encompassing various components such as facial expressions, body posture, and hand movements. While recognizing the importance of other aspects, this work specifically focuses on hand gestures as a means of communication. Technological advancements have enabled hand gestures to serve not only in human interaction but also as a crucial component of touchless human-computer interaction systems. This potential application is evident in sectors such as the automotive industry [Pickering et al., 2007] or augmented and virtual reality [Li et al., 2019b; Sagayam et al., 2016].

In general, various gestures vary significantly from each other and also the same gesture varies substantially between different performers. When aiming to describe the entirety of hand gestures, decomposing them into a pose and a global movement is likely the most fitting approach. Here, both the pose and the movement can be either static or dynamic [Oudah et al., 2020]. For instance, in a swipe gesture the movement is dynamic, while the pose mostly remains static throughout the sequence. Conversely,

in a pinch gesture the thumb and index finger come together and then separate within the sequence. In this case, there might not be a movement of the hand, but the pose changes, making it dynamic.

Therefore, detecting a gesture algorithmically is a challenging task. Due to the significant success of deep learning in the field of computer vision, numerous new approaches have been developed to recognize hand gestures based on visual sequence data. These methods can be categorized into image-based techniques and skeleton-based methods. Image-based techniques directly utilize data produced by sensors, such as RGB (cite) or depth data [Molchanov et al., 2015]. Bigalke et al. [2021] generate a sequence of 3D point clouds from data captured by a depth camera and create features for both the global position of the hand and the local pose, based on basic point sets. These features are then processed in an LSTM to detect the gesture within the sequence.

Conversely, skeleton-based methods, as the name suggests, utilize a hand skeleton as input modality [Chen et al., 2019b; Zhang et al., 2020]. In contrast to the previously presented methods, this requires the processing of significantly fewer points and therefore substantially reduces the complexity of the task. In line with the thematic background of this chapter, special emphasis will be placed on these methods in the introduction. A commonly used approach involves dividing the information contained in hand poses into pose and movement information, processing them separately and ultimately inferring the gesture from both pieces of information. Liu et al. [2020] convert the skeleton sequence into a hand posture evolution volume, representing the 3D visualization of all skeletons in the sequence in an abstract sense. Subsequently, a 3D CNN is used to extract pose features. Additionally, hand movement is encoded in a 2D map based on the central hand point and the fingertips, enabling the extraction of hand movement features through 2D CNNs. The final prediction is generated by averaging both sets of features.

Taking a similar information decomposition approach, Shi et al. [2020] leverage the self-attention mechanism of the transformer [Vaswani et al., 2017] to extract spatial and temporal information from the sequence of hand skeletons within a spatial and temporal attention module. These information are then used to derive the final prediction.

Another approach involves interpreting the hand skeletons as graphs, allowing the extraction of spatio-temporal features through Graph Convolutional Networks to draw conclusions about the gesture [Cheng et al., 2020; Yang et al., 2022a]. Liu et al. [2023] expand upon the conventional approach of a statically defined adjacency matrix by dynamically learning a temporal adjacency matrix and a channel-dependent adjacency matrix, enabling a more accurate description of the connections between different skeletons within the sequences.

Without loss of generality, hand skeleton-based methodologies are trained and evaluated on datasets providing both gesture labels and ground truth positions of hand keypoints. In practice, however, ground truth hand keypoints are not available. Therefore, within this part of the thesis, we aim to investigate the influence of using

network-predicted hand poses on the quality of gesture recognition. Specifically, we aim to draw conclusions regarding the suitability of the methods we propose for estimating hand poses within the domain of gesture recognition. The structure of this chapter is as follows: Initially, a representative method leveraging hand keypoint positions for gesture recognition is presented. Subsequently, we employ this method to determine the accuracies achievable in gesture recognition using the methodologies outlined in Chapter 3 and Chapter 5 for determining hand poses. Finally, we discuss the insights gathered and identify remaining questions or areas for further exploration.

6.2 Methods

Following the overarching aim of this chapter to analyze whether our estimated hand poses are well-suited for gesture recognition, we refrain from developing a new method and opt to utilize an existing algorithm for evaluation. Since hand pose estimation already consumes a significant portion of computational resources, gesture recognition should have minimal complexity to ensure the overall framework remains real-time capable. With a frame rate of up to 3500 fps on a GPU or 2000 fps on a CPU, coupled with state-of-the-art performance, the DD-Net by Yang et al. [2020] stands out as an extremely lightweight network, making it a suitable choice for the subsequent experiments. In this section, we will briefly introduce the network. For more precise information, interested readers are referred to the work of the authors.

An essential component of the DD-Net involves generating various features to best distinguish between different gestures. This is based on the assumption that each gesture exhibits different variances or invariances. For instance, in a pinch gesture where the thumb and index finger come together and then move apart, the relations between the keypoints of the index finger and thumb change, while other keypoints might undergo minimal movement and possibly maintain their spatial positions. In contrast, in a swipe gesture the relations between the keypoints are largely preserved while the entire hand is affected by translation and rotation. To capture these differences in gestures, it is crucial to use features that are rotation- and translation-invariant and features that describe the global spatial position.

For the latter, Euclidean features are suitable as they directly describe the global positions of the respective keypoints. For features that are rotation- and translation-invariant, the authors propose Joint Collection Distances (JCD) as features. These are defined as the Euclidean distances between each keypoint and all other keypoints. This results in a symmetric distance matrix that essentially describes the relations between individual keypoints. To save memory, the symmetry of the distance matrix can be exploited by storing the contained information in a lower triangular matrix.

Figure 6.1 illustrates the concepts of the two features. The visualization depicts a hand pose at two different time points, t_1 and t_2, experiencing a rotation and potentially

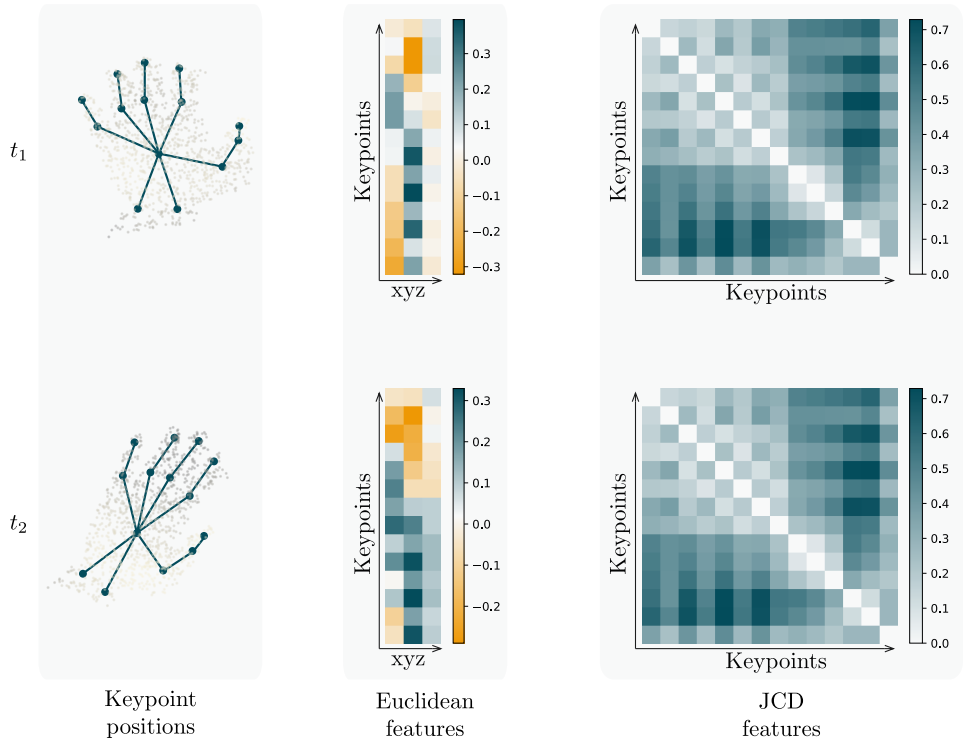


Fig. 6.1: A visualization of the distinct features processed in the DD-Net. On the left, hand poses with their respective keypoint positions are exemplarily depicted at two time points, t_1 and t_2. During the time interval $\Delta t = t_2 - t_1$, the hand is only affected by a rotation. The Euclidean features (middle) exhibit differences due to the spatial change in keypoint positions, whereas the JCD features (right) remain identical by definition.

a translation within the time difference $\Delta t = t_2 - t_1$. The Euclidean features undergo substantial changes, reflecting alterations in the spatial positions of all keypoints. On the other hand, the JCD features remain identical at both time points, indicating that the hand pose itself has not changed.

Equipped with the two introduced features, a network for gesture recognition can now be defined. An overview of the entire DD-Net is depicted in Fig. 6.2. The authors assume that each sequence to be processed consists of $T = 32$ frames. In the first step, each individual frame is used to compute the features, resulting in 32 Euclidean features and 32 JCD features. As visible in the figure, the JCD features are directly input into an embedding, whereas the Euclidean features undergo a temporal difference computation, representing the movement of the $K \in \mathbb{N}$ keypoints. For fast movements, the difference between each second frame t_n, $n = 3, 5, ...$, and the frame two time

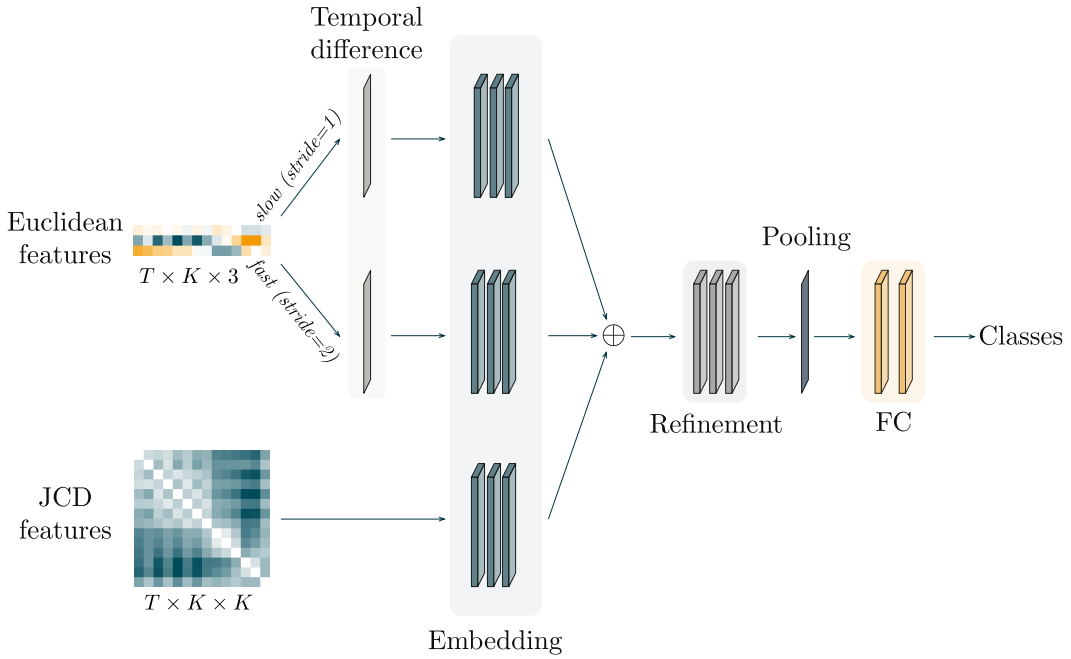


Fig. 6.2: An overview of the DD-Net is depicted. The Euclidean features, processed with a temporal stride of two for fast movements and a stride of one for slow movements, along with the JCD features, are each handled in a feature embedding module. This embedding primarily comprises three ConvNet layers. Subsequently, all three features are concatenated and further processed in a refinement module. A final max-pooling operation, followed by two fully connected layers, ultimately leads to the desired classes.

points earlier t_{n-2} is computed. A similar procedure is employed for detecting slow movements using each frame and its previous frame t_n-1 for the difference. This yields 15 motion features for fast movements and 31 for slow ones. To achieve a uniform size, the motion features are padded to 32 and the missing frames are determined through linear interpolation.

To process uncorrelated keypoint positions on the one hand and reduce the influence of noise in the given poses on the other, the authors feed the features into specific embedding modules. Each embedding module comprises three consecutive Conv1D operations. However, for the features related to fast movements, an additional Max-Pooling operation is appended since they cover twice the temporal length. After this, a concatenation brings all features together. A Subsequent refinement step, consisting of additional Conv1D layers, followed by Global Average Pooling and two Fully Connected layers, ultimately leads to the gesture recognized by the network. It is also worth noting that the size of the model can be adjusted by a single parameter, allowing control over its complexity.

6.3 Experiments and Results

In this section, the presented DD-Net will be evaluated using pseudo ground truth poses, the per-frame poses from Chapter 3 and the tracking-based poses from Chapter 5. This evaluation aims to draw conclusions regarding whether the quality of the estimated hand poses is sufficient for a proper gesture recognition.

Datasets. For evaluation purposes, we refer to existing works [Hou et al., 2019; Li et al., 2019c] and utilize the widely used public datasets DHG [De Smedt et al., 2016] and SHREC [De Smedt et al., 2017]. The DHG dataset comprises sequences of 14 gestures performed by 20 subjects, each executed five times in two distinct ways. Once with the entire hand and once using a single finger. This results in a total of 2800 different sequences. Similarly, the SHREC dataset includes 14 gestures performed in the same two ways, with data recorded from 28 subjects who executed the gestures between 1 and 10 times. The total number of sequences is likewise 2800. The individual frames of these sequences were captured using an Intel RealSense Depth camera, providing depth data and the positions of 22 hand keypoints as pseudo ground truth by Intel RealSense. We utilize the depth data to generate 3D point clouds using camera intrinsics, extract the hand region based on the keypoint positions, and subsequently reduce the number of points in the cloud to 1024 using farthest-point-sampling.

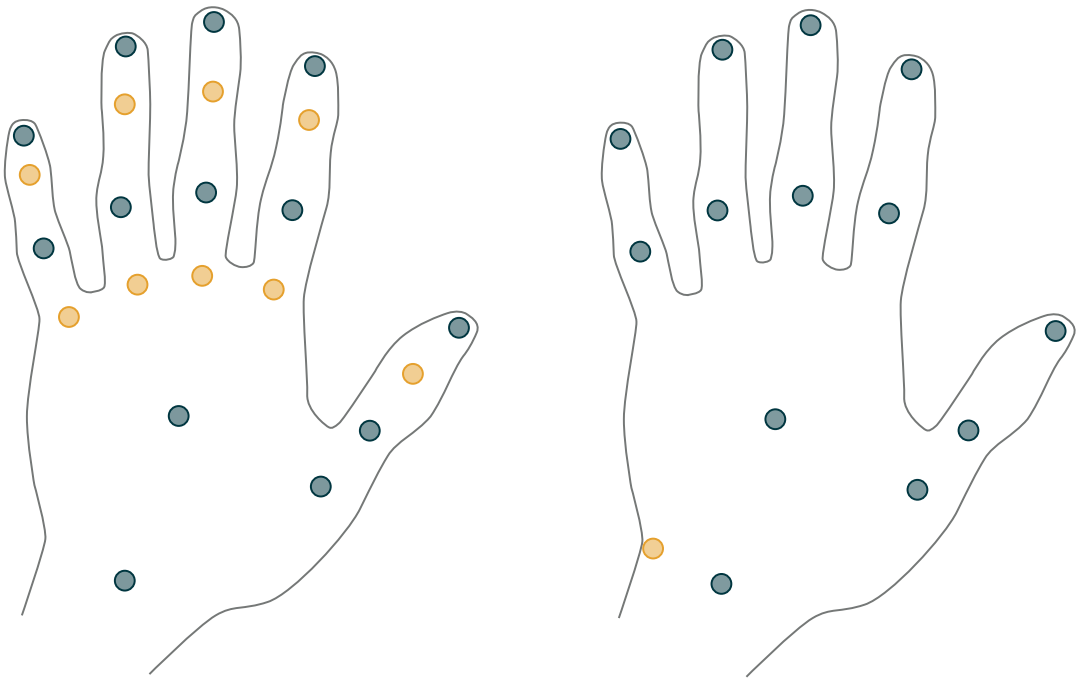
Both datasets distinguish between 14 labeled gestures and 28 labeled gestures. In the latter case, there is additional differentiation in the labels depending on whether the gesture is performed with one finger or the entire hand. Within the dataset descriptions, the 14 labels are used to test algorithms for consistent hand shape recognition when the same gesture is performed with different fingers. On the other hand, the 28 labels require a finer distinction of movement.

It is worth noting that the SHREC dataset includes a designated test subset of approximately 30% of the dataset. In contrast, test splits for the DHG dataset must be defined independently. Therefore, we follow the leave-one-out principle and use one subject for evaluation and the remaining for training.

Implementation Details. In general, we adhered to the PyTorch implementation of the DD-Net provided by Yang et al. [2020] to ensure optimal comparability of results and also set the parameter for the complexity of the DD-Net to 64. The individual training sessions were conducted for 199 epochs with a batch size of 64 and an Adam [Kingma et al., 2014] optimizer with an initial learning rate of 0.01. All trainings and evaluations were run on a PC with a GeForce® GTX 1080 Ti GPU.

6.3.1 Data Preparation

In the experiments, we aim to investigate the impact on the classification accuracy when estimated keypoint positions are used instead of pseudo ground truth hand keypoint positions. The estimations will be generated using the methods developed

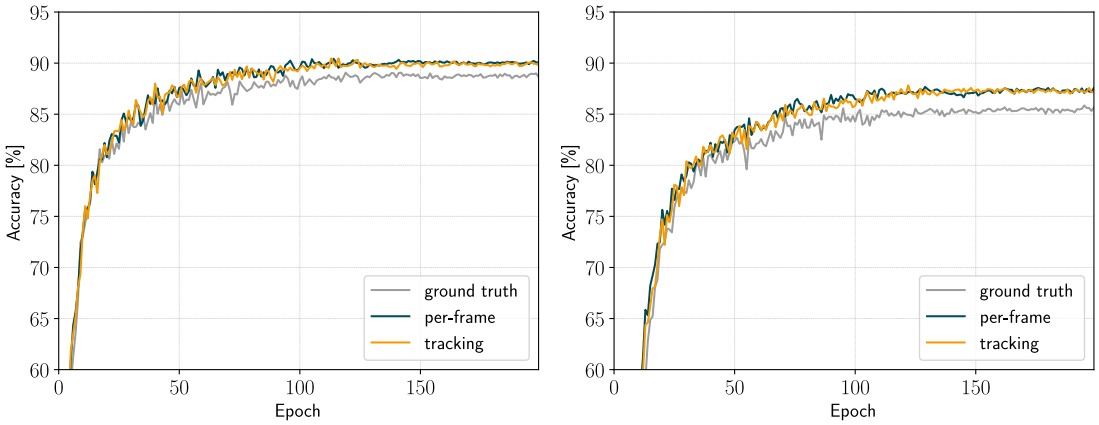


(a) SHREC & DHG. **(b)** NYU.

Fig. 6.3: Representation of different hand poses from the SHREC & DHG datasets (left) and the NYU dataset (right). To standardize all datasets to an identical set of points with same anatomical correspondences, only a subset of possible keypoints (dark green) is considered while the remaining keypoints are ignored (orange).

in the previous chapters. Since the pseudo ground truth hand poses were obtained through camera-integrated hand skeleton tracking, it is assumed that these positions are imperfect. For this reason and to simulate a realistic out-of-domain scenario, we will not retrain the methods presented in Chapter 3 and Chapter 5 but instead use the networks already trained on the NYU [Tompson et al., 2014] dataset. This implies that the networks were trained on data from a different sensor, exhibiting different noise characteristics which make it necessary to overcome a domain gap. Moreover, it is expected that not all gestures performed in SHREC and DHG datasets are covered in the NYU training set, posing a challenge to the network's generalization capabilities.

For simplicity, moving forward we will refer to the approach developed in Chapter 3 and trained on the NYU dataset as the *per-frame* method. Additionally, the algorithm utilizing temporal information from Chapter 5 and also trained on the NYU dataset will be termed as the *tracking* method. We utilize both methods to obtain different estimations for the hand keypoints. The pre-trained per-frame network can directly determine the hand pose from the respective point clouds. Conversely, the tracking



(a) 14 labels. **(b)** 28 labels.

Fig. 6.4: The accuracy of the DD-Net on the DHG dataset is depicted for each training epoch using pseudo ground truth, per-frame, or tracking keypoint positions. The evaluation of the 14 and 28 labeled gestures is shown on the left and right, respectively.

method receives these per-frame estimations to track the hand pose for the respective sequences.

Using the NYU dataset for training the hand pose estimation methods introduces another challenge because the keypoint definitions in NYU differ from those in SHREC and DHG. While NYU evaluates on 14 keypoints, both SHREC and DHG provide 22 keypoints that can be used during training and evaluation. To ensure a meaningful comparison, the number of used keypoints needs adjustment. Figure 6.3 illustrates that a set of 13 keypoints is attributed the same anatomical significance in both datasets. Consequently, during the subsequent experiments we use the depicted keypoints for consistency.

6.3.2 Result Analysis

After preparing the necessary data, this section involves evaluating the different input data for suitable gesture recognition. To do so, we train and evaluate the DD-Net using input from either pseudo ground truth, per-frame, or tracking keypoint positions. We start by examining the DHG dataset. Since there is no official test set, we follow the leave-one-out principle, using one subject for evaluation and the remaining for training. We repeat this methodology for each subject, ensuring that each subject is used once for evaluation. The accuracy, calculated as the number of correctly identified gestures divided by the total number of performed gestures, is averaged across all subjects for this dataset.

Figure 6.4 displays the network's accuracy for the respective used keypoints over the training epochs, distinguishing between 14 labeled (left) and 28 labeled (right)

Table 6.1: Listing of achieved gesture recognition accuracies on the DHG dataset for 14 and 28 labeled gestures using various keypoint positions as network input. The accuracies of the last five epochs were averaged to reduce variance. The best results are highlighted in bold.

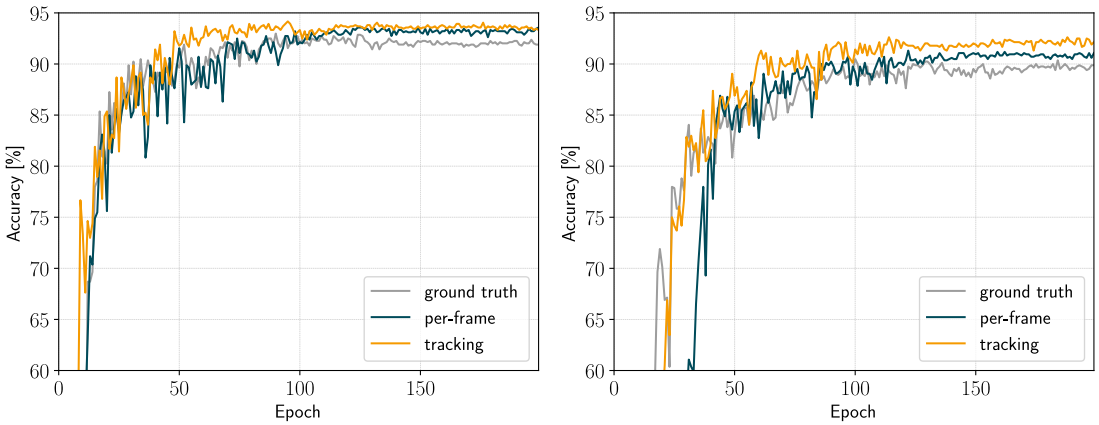
Input keypoints	14 labels [%]	28 labels [%]
pseudo ground truth	88.88	85.61
per-frame	**90.07**	**87.30**
tracking	89.95	87.26

gestures. One notable observation is that the accuracy, both for 14 and 28 labeled gestures, is lowest when utilizing the provided pseudo ground truth keypoint positions from the DHG dataset. This can be attributed to the use of the camera-integrated hand tracking method to determine the pseudo ground truth positions without manual annotation, resulting in imperfect positions. Despite the mentioned domain gaps that the per-frame and tracking methods must overcome, the estimated keypoints seem to be more suitable, leading to improved gesture recognition accuracy. Furthermore, the two figures show that the accuracy of the methods varies significantly at the beginning and still exhibits some variance after 199 epochs. To reduce the impact of this variance, we average the accuracy of the last five epochs to compute final results, as detailed in Table 6.1. The keypoints obtained through the per-frame method achieve 90.07% accuracy for 14 labeled gestures and 87.3% for 28 labeled gestures, showcasing the highest accuracies. However, these results differ only marginally from the accuracies obtained using tracking keypoint positions as input. In contrast, the pseudo ground truth keypoint positions result in accuracies over 1% lower.

We conduct a similar evaluation on the SHREC dataset. However, due to the structure of the dataset, the leave-one-out procedure can be omitted since gestures are explicitly designated as training or test splits. An examination of the accuracy for the respective keypoints across training epochs in Fig. 6.5 quickly reveals that the variance of accuracy is significantly higher than in the DHG dataset. This can be attributed to the fact that we evaluated only one test set instead of averaging over multiple conducted tests. The accuracy with pseudo ground truth keypoints remains consistently the lowest, still influenced from inaccuracies in the generation of positions. Another notable observation is that for 14 labeled gestures (left), the accuracy with tracking keypoints from epoch 50 to 100 is notably higher than with per-frame and pseudo ground truth keypoints. Subsequently, the accuracy with per-frame keypoints approaches the accuracy of the tracking keypoints. Particularly on the 28 labeled gestures, there is a small gap between tracking and per-frame keypoints with the accuracy of tracking keypoints being slightly higher. A look at Table 6.2 confirms this impression. Although the accuracy with tracking keypoints is only marginally higher

Table 6.2: Listing of achieved gesture recognition accuracies on the SHREC dataset for 14 and 28 labeled gestures using various keypoint positions as network input. The accuracies of the last five epochs were averaged to reduce variance. The best results are highlighted in bold.

Input keypoints	14 labels [%]	28 labels [%]
pseudo ground truth	92.04	89.66
per-frame	93.38	90.81
tracking	**93.43**	**92.14**



(a) 14 labels. **(b)** 28 labels.

Fig. 6.5: The accuracy of the DD-Net on the SHREC dataset is depicted for each training epoch using pseudo ground truth, per-frame, or tracking keypoint positions. The evaluation of the 14 labeled gestures is shown on the left, while the 28 labeled gestures is displayed on the right.

than with per-frame keypoints for 14 labeled gestures, there is an improvement of 1.33% in 28 labeled gestures when using tracking keypoints.

To further investigate the differences in gesture recognition in this specific scenario, we tallied the individual recognized 28 labeled gestures using the tracking and per-frame keypoints, subtracted them from each other, and visualized the results in a bar chart (Fig. 6.6). Orange bars indicate negative values, meaning that utilizing the per-frame keypoints results in more correctly identified gestures of a particular type. Conversely, dark green bars represent positive values, indicating more detected gestures by using the tracking-based keypoints. Unfortunately, a clear trend cannot be discerned from the illustration. However, it can be noted that the rotation gesture was consistently recognized more accurately using the per-frame keypoints. On the other hand, the tracking keypoints led to improved recognition of the tap, pinch, and shake gestures.

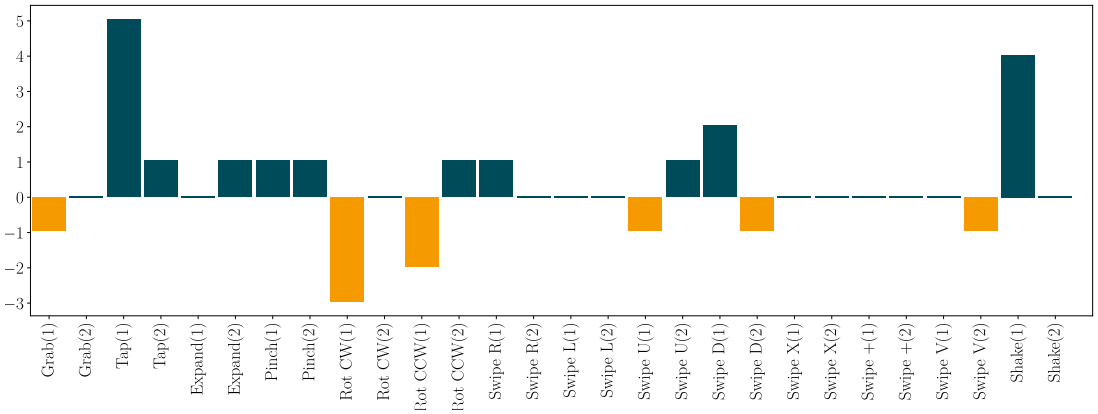


Fig. 6.6: The chart illustrates the differences in gesture recognition using per-frame and tracking keypoints on the SHREC dataset with 28 labeled gestures. The number of correctly detected gestures using the two variants was subtracted from each other and presented as bars. The ■ bar indicates more gestures of a certain type detected using per-frame keypoints, while conversely, ■ represent more detected gestures by the tracking keypoints.

From these results, we conclude that both methods to estimate hand keypoints are able to detect the global hand shape and thus yield a suitable gesture recognition for a small variety of different gestures. However, increasing the number of different gestures, utilizing tracking to improve the keypoint positions helps to better distinguish between gestures and thus enhances gesture recognition. It is worth mentioning that per-frame keypoints still achieve good quality for an increased amount of gestures. Depending on available computational resources, per-frame keypoints could therefore also be a suitable choice.

6.4 Discussion and Conclusion

The aim of this chapter was to evaluate whether estimated hand keypoint positions are of sufficient quality to perform accurate gesture recognition. To achieve this, we trained and evaluated an existing skeleton-based gesture classification network on the public datasets DHG [De Smedt et al., 2016] and SHREC [De Smedt et al., 2017]. For this purpose, we utilized both the pseudo ground truth keypoint positions provided by the datasets and the methods presented in Chapter 3 and Chapter 5 trained on the NYU [Tompson et al., 2014] dataset, to obtain our own keypoint estimations. These estimations were then used for training DD-Net [Yang et al., 2020] as gesture recognition network. In the experiments, we were able to demonstrate that the previously developed methods for estimating keypoints provide results that allow gesture classification to be performed with very high accuracy. The results showed that both the per-frame

keypoints from Chapter 3 and the tracking keypoints from Chapter 5 achieve higher accuracy in gesture recognition than using pseudo ground truth keypoints, despite the need to overcome a domain gap. This is due to the use of camera-internal hand tracking within the datasets to determine the pseudo ground truth positions of the keypoints. Consequently, the generation of ground truth keypoints is already subject to errors. For coarse gestures, both per-frame and tracking keypoints yield similar performance results. However, for finer gestures, the temporal information introduced by the tracking algorithm can further improve the accuracy of gesture recognition beyond what the per-frame keypoints offer.

In conclusion, a lightweight network was used for estimating gestures, demonstrating that our presented methods for hand pose estimation can be utilized for high accuracy gesture recognition without significant computational overhead using the presented methodologies.

Chapter 7

Summary and Conclusion

Within the last four methodological chapters of this thesis, various novel deep learning-based approaches for hand analysis were introduced and comprehensively evaluated. To address challenges associated with hand-related tasks effectively, the potential of 3D data was explored, leveraging geometric deep learning methodologies in the proposed approaches. The central focus of these methods was particularly on extracting crucial hand information to enable intuitive and touchless human-machine interaction. The presented methods cover all necessary, interdependent aspects for recognizing hand gestures based on 3D point clouds.

This concluding part of the thesis aims to provide an overview of the achieved scientific contributions, as well as to discuss and contextualize the gathered insights. This chapter begins with a summary of the main contributions of the methodological chapters in Section 7.1. Subsequently, the research findings and the results of the introduced methods are discussed and placed in the context of the formulated objectives (Section 7.2). The final part discusses promising future research directions and identifies open problems and limitations of the proposed methods.

7.1 Contributions

Exploiting 3D information of point clouds in a graph CNN for hand pose estimation. With the emergence of stereo cameras and time-of-flight sensors and their deployment into the commercial sector, depth data and the extractable 3D point clouds have become a promising modality due to their richness in spatial information. In Chapter 3 of this thesis, the potential of point clouds for estimating hand poses using graph neural networks was investigated (Fig. 7.1). To achieve a possible reduction in the complexity of the network, the keypoint positions to be estimated were designed as weighted sums of the considered input point cloud. This enables a network to learn one-dimensional weights rather than three-dimensional positions or directions. The proposed hierarchical graph CNN is well-suited for analyzing both global and local spatial information within the point cloud, deriving suitable weights to determine the keypoint positions. This low-complexity network already achieves nearly state-of-the-art accuracy. However, the estimation of keypoint positions is limited to the convex hull

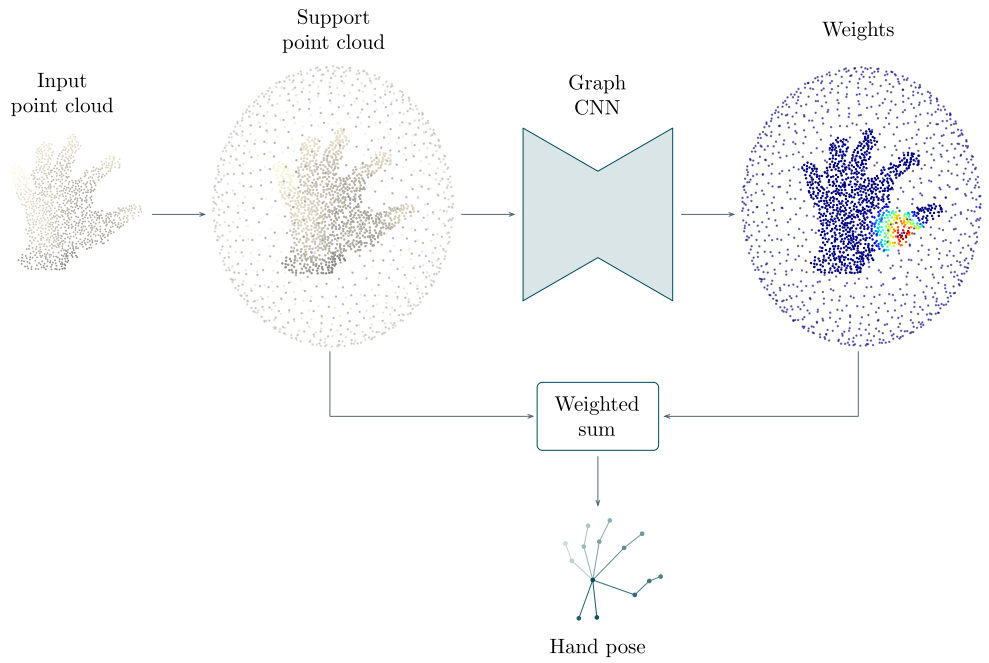


Fig. 7.1: Schematic overview of the method presented in Chapter 3 for estimating a hand pose on a 3D point cloud.

of the point cloud, particularly encountering issues with occluded hand parts. The definition of a support point cloud around the hand point cloud extends the convex hull and simultaneously provides a structured search space for keypoint positions. Through meaningful information exchange between the hand and support point clouds, a significant improvement in accuracy was achieved. The resulting network achieves state-of-the-art accuracy while significantly reducing the required network parameters. The conducted experiments confirmed that the exchange of information between the support and hand point clouds greatly enhances the accuracy of hand pose estimation. Furthermore, they also indicate that optimizing the support point cloud holds further potential for accuracy improvements.

In summary, this chapter demonstrated that 3D point clouds carry crucial spatial information and that a hierarchical graph CNN is a suitable approach for leveraging this information for hand pose estimation. The definition of a support point cloud around the cloud of interest simplifies dealing with occluded parts of the hand by structurally expanding the search space for potential keypoint positions.

Integration of motion information into hand pose estimation. In the first chapter, only static information in form of a single frame was utilized for hand analysis. To adequately address the issue of occluded hand parts, it is advantageous to consider the trajectory of hand movement to interpolate the information missing in the static data.

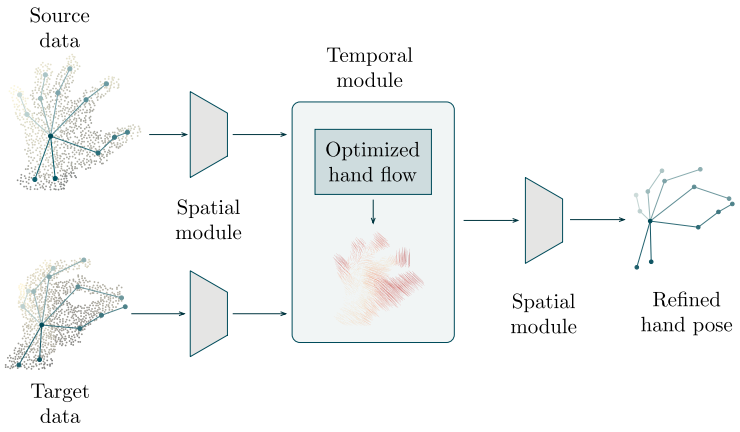


Fig. 7.2: Schematic overview of the method presented in Chapter 5 that leverages temporal information of a sequence of 3D point clouds for an improved hand pose estimation.

Therefore, in Chapters 4 and 5, we explored methods to analyze motion information in the form of scene flow and integrated the information into hand pose estimation.

Existing evaluations primarily rely on data with objects undergoing rigid transformations, meaning it is unknown whether these methods are suitable for human motion analysis that consists also of deformation of the objects. Thus, in Chapter 4, a new benchmark for scene flow techniques was developed. It was essential to specifically prepare existing human datasets for the task of scene flow estimation, enabling an evaluation on data that includes objects that are affected by realistic anatomical deformations. In the experiments, we demonstrated how scene flow methods behave in the individual aspects of movement: translation, rotation and deformation. Through the use of new metrics we were able to highlight important strengths and weaknesses of scene flow methods for estimating human movement. Furthermore, the real-time capability of scene flow methods regarding potential deployment in real applications was emphasized through a comprehensive runtime analysis.

Chapter 5 introduced a novel method that incorporated the insights gathered from motion analysis to enhance static hand pose estimation by leveraging an entire sequence (Fig. 7.2). Initial experiments showcased that optimizing scene flow techniques to specifically estimate the movement of the hand skeleton yielded certain improvements in accuracy, yet static hand pose estimation remained superior. Further experiments demonstrated that a geometric deep learning network is able to learn weights to choose the more accurate keypoints between statically estimated hand poses and those estimated through skeleton movement. This resulted in initial improvements compared to a purely statically generated hand pose. The network was complemented with a spatial module that validated generated hand poses for anatomical plausibility. In experiments, we demonstrated that the combination of motion information and anatomical plausibility significantly improved static hand poses. Another advantage

of the method is its flexibility in allowing a wide range of techniques for hand pose estimation and scene flow estimation. This flexibility may enable our method to benefit from future improvements in these respective domains. The demonstrated generalization capability enables a seamless interchange of methods for static hand pose estimation, ensuring that the trained network consistently achieves improvement of the static poses. Furthermore, the experiments analytically showed that the negative impact of noisy keypoint positions can be noticeably reduced by employing the novel network.

Overall, Chapter 4 initially revealed the strengths and weaknesses of existing scene flow estimation techniques for human hand movements. These valuable insights were then incorporated into a new methodology in Chapter 5, that, as to the best of our knowledge first approach, utilizes scene flow to significantly improve static hand pose estimation by employing sequences of point clouds.

Demonstrating the feasibility of predicted hand poses for gesture recognition. While the previous chapters focused on determining hand poses based on point clouds or sequences of point clouds, it remains unclear how suitable the resulting hand poses are for communication applications. Therefore, Chapter 6 centers around human-machine interaction through hand gestures. Specifically, this chapter investigates whether the methods developed in Chapters 3 and 5 for estimating hand poses are also suitable for recognizing hand gestures of users (Fig. 7.3). The use of common public datasets in all experiments ensured comparability with the state-of-the-art. An existing lightweight network was trained and evaluated using the estimated keypoint positions from Chapters 3 and 5 as input, aiming to analyze the potential of these methods in gesture recognition. With a frame rate of up to 3500 fps on a single GPU, the approach stands out particularly for real-world applications, where limited resources

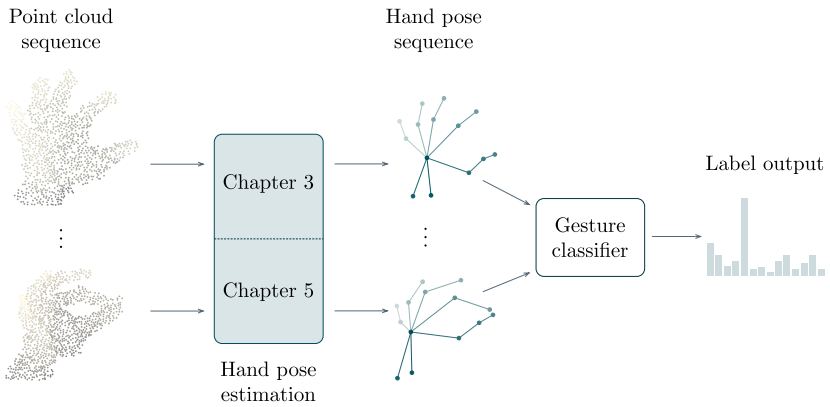


Fig. 7.3: Schematic overview of the method in Chapter 6 that recognizes gestures based on hand poses that are generated using the presented methods from Chapters 3 and 5.

can be allocated elsewhere and are only marginally needed for gesture estimation. The experiments demonstrated that the estimated hand poses are extremely well-suited for gesture recognition. Moreover, they achieve a more accurate gesture classification than the annotated keypoint positions included in the datasets, thus establishing a new state-of-the-art level of accuracy. Furthermore, it was numerically shown that the recognition of specific poses particularly benefits from the higher accuracy of pose estimation through the integration of motion information.

In summary, the final methodological chapter showed that the novel methods presented in this thesis for estimating hand poses are extremely well-suited for achieving precise gesture recognition with minimal computational effort.

7.2 Research Findings

In this section, the previously outlined contributions of this thesis will be correlated with the defined objectives formulated in the introduction Chapter 1. Three different aspects are considered in particular: the exploration of various tasks on point clouds using geometric deep learning methods, addressing and resolving of hand-related issues, and the overarching aim of this thesis, which is to investigate possibilities for non-contact human-machine interaction.

Tasks on point clouds for hand analysis. Within this thesis, various geometric deep learning-based methods on point clouds have been developed, covering diverse areas in hand analysis and beyond, thus enabling a wide range of applications.

In the first methodological chapter, a precise estimation of hand poses using 3D point clouds was achieved. A hand pose comprises a defined number of points corresponding to anatomical hand structures, and knowledge of their positions is a crucial component of hand analysis. The superiority of the presented graph CNN-based method is particularly evident on the challenging NYU dataset [Tompson et al., 2014], where an average keypoint error of 9.30 mm is achieved without post-processing. This means an error reduction of comparable state-of-the-art methods by an average of 20%. Besides its excellent accuracy, a reduction in computational complexity was achieved by reformulating the problem to find 1D weights for determining the 3D keypoint positions. Beyond focusing on the hand as a specific target, it can be assumed that the application of the method in other areas also yields accurate estimations, such as in human body pose estimation or the localization of 3D landmarks in organs.

Chapter 4 utilizes information from two point clouds captured at different time points to determine object movement with scene flow. A novel benchmark on data that contain objects that are affected by real anatomical deformations revealed significant insights. Optimal transport methods, in combination with translation-invariant features, proved to be excellent in analyzing the rotational and translational components in

object movement. With an average flow vector error of 3.07 mm, the FLOT approach [Puy et al., 2020] outperformed other state-of-the-art methods even on real hand data. While scene flow methods have potential beyond motion analysis, such as in object registration like organs, their application in finding correspondences revealed that while the examined methods could correctly assign nearly 80% of correspondences on synthetic data, they struggled to surpass 50% correctness on real data.

Chapter 5 of this thesis combines the two aforementioned tasks, utilizing information from a sequence of point clouds as input. A module analyzing the temporal information of the sequence determines the scene flow between two point clouds and uses it to transfer the hand pose from the earlier time point to the current time point based on the estimated motion. A novel approach to optimize the scene flow specifically for the movement of the hand skeleton improved the accuracy of motion-estimated poses by nearly 3% relative. Coupled with another module that proves and corrects existing hand poses for anatomical plausibility, a novel network was introduced to track hand poses for a given sequence of hand point clouds. With a 9% lower average error in the resulting hand poses, this method outperformed a comparable state-of-the-art method for pose tracking.

In summary, the methods developed in the first three methodological chapters address a wide spectrum of problems to be solved on point clouds. Individual point clouds are considered for estimating keypoint positions or localizing landmarks, two point clouds are analyzed to determine object movement or register two objects. When a whole sequence of point clouds is available, optimized movement for a skeleton or improved accuracy of hand poses through tracking can be achieved. All presented methods in the field of geometric deep learning yield promising results and consistently meet the requirement of real-time capability, allowing for their potential deployment in real-world applications.

Addressing hand-specific problems. The human hand is an anatomically complex organ with a high interindividual variability, resulting in numerous degrees of freedom in a confined space which makes simulations challenging. Additionally, parts of the hand are often occluded in projection images, providing only partial information for analysis. Addressing and solving these specific challenges was a primary focus of this thesis.

With the advent of stereo cameras and time-of-flight sensors in the commercial sector, generating 3D point clouds of objects has become more accessible than ever. In Chapter 3, we demonstrated that 3D point clouds are rich in spatial information and local structures, enabling a better analysis of hand complexity. A graph CNN proved to be a suitable geometric deep learning technique for extracting crucial spatial features from these local structures. The presented hierarchical approach, utilizing different point cloud resolutions, enables the incorporation of these local features into a global context, facilitating a robust estimation of hand poses. An essential component of the novel

methodology is the definition of a support point cloud around the hand point cloud. This structurally expands the search space for keypoint positions, significantly simplifying the estimation of occluded keypoints. Experiments demonstrated the significance of this methodology by reducing the error by 20% from 11.64 mm to 9.30 mm. Additionally, the potential of the support point cloud has not been fully exploited, as evidenced by achieving a reduction in error to 8.83 mm when optimally placed around the entire hand.

Even if the focus is on analyzing hand movement, the anatomical complexity poses a significant challenge. Object movement essentially consists of translation, rotation, and irregular deformation. The performed analysis in Chapter 4 showcased that the deformation component in body movement typically does not exceed 68%, while the examined hand dataset exhibited an average deformation component of over 80%. The novel introduced benchmark revealed that the optimal flow-based approach FLOT [Puy et al., 2020] is most capable of estimating deformation-based movement. While all examined methods showed similar small errors below 1 mm for small deformations under 2 mm, FLOT particularly stood out with the smallest error for larger deformations above 2 mm. Another insight from the experiments was how these methods dealt with occluded parts of the object. The investigated PointPWC network [Wu et al., 2020] emerged as the most precise method when an entire scan of the object was available. However, it faced problems in cases of occlusion, making the other methods more suitable specifically for estimating hand movements. The new benchmark demonstrated that some of the investigated methods hold great potential for determining hand movement. Subsequently, it was crucial to leverage these significant insights in the field of scene flow estimation for a more effective prediction of hand poses.

In Chapter 5, the hand-specific challenges were further addressed and solutions improved by utilizing information from an entire sequence of point clouds. The first introduced module uses PCA to map an existing hand pose into a lower-dimensional pose space, avoiding unnatural hand poses. A learned confidence measure enables the selection of better keypoint positions from incoming and anatomically constrained poses. Incorporating this module into the network reduced the error of the statically generated hand poses from Chapter 3 by 1.1%. This result also indicates that the poses from Chapter 3 generally exhibit plausible anatomy. A second module employs a scene flow network specifically optimized for the hand skeleton movement, thus enabling to transfer the pose from a previous time point to the current one using the estimated hand movement. Similar to the spatial module, a learned confidence measure ensures the optimal choice of keypoints from the existing and motion-based poses. Integrating this module into the entire network compensated for missing information due to occlusion, achieving a reduction in error of up to 7.6%. This experiment also demonstrated that occlusion of crucial parts is a significant challenge in frame-wise hand pose estimation and integrating motion information in the form of scene flow represents a suitable optimization strategy.

To sum up, this thesis introduced a variety of novel strategies to effectively address and resolve hand-specific problems. The high anatomical complexity of the hand can be effectively modelled and its challenges for machine learning algorithms mitigated by exploiting three-dimensional data. However, the occlusion of essential hand parts poses a substantial challenge, particularly for individual frame-based methods. Leveraging temporal information proves to be a meaningful strategy to better tackle this problem.

Exploiting hand information for human-machine interaction. The overarching goal of this thesis was to leverage extracted hand information for touchless human-machine interaction. Chapter 6 focussed on this objective and demonstrated the potential of the obtained hand poses from Chapters 3 and 5 for gesture recognition. In potential interactions, hand gestures can be used by a user to trigger specific events at an interface, enabling them to control a machine. A lightweight network was trained and evaluated on public datasets using hand poses generated by the novel methods. An accuracy of over 93% in recognizing 14 different gestures showcased the potential of the proposed methods in gesture recognition. Additionally, our estimated hand poses could correctly assign up to 2.5% more gestures compared to using annotated hand poses from the datasets. A closer examination of the achieved results also revealed that recognizing specific gestures such as tap, swipe, and shake benefits from the integration of temporal information in hand pose estimation. More gestures were correctly assigned when relying on the hand poses generated by the method from Chapter 5.

Especially in interactions within a virtual environment, the ability to interact with objects is crucial. Knowledge of the positions of effectors is essential for precise manipulation of virtual objects. The approach presented in Chapter 3 for estimating hand poses provides robust results for fingertips with an average error of 11.3 mm on the NYU [Tompson et al., 2014] dataset, thus enables centimetre precision in controlling the objects and making it a suitable choice for such applications.

Another important aspect for real-world applications is the runtime of the developed methods to ensure real-time capability. Through the achieved reduction in the complexity of the hand pose estimation method in Chapter 3, a runtime of over 30 fps was achieved without further optimization, ensuring real-time capability. Gesture recognition can also be added to an application seamlessly. Since a particularly fast lightweight network was used for this purpose, a runtime of up to 3500 fps on a GPU can be achieved, requiring only minimal computational resources.

In summary, within this thesis novel methods for extracting hand information were developed, which are highly suitable for gesture recognition, providing valuable information for touchless human-machine interaction applications. Moreover, the developed methods are real-time capable, enabling integration into an application with low effort.

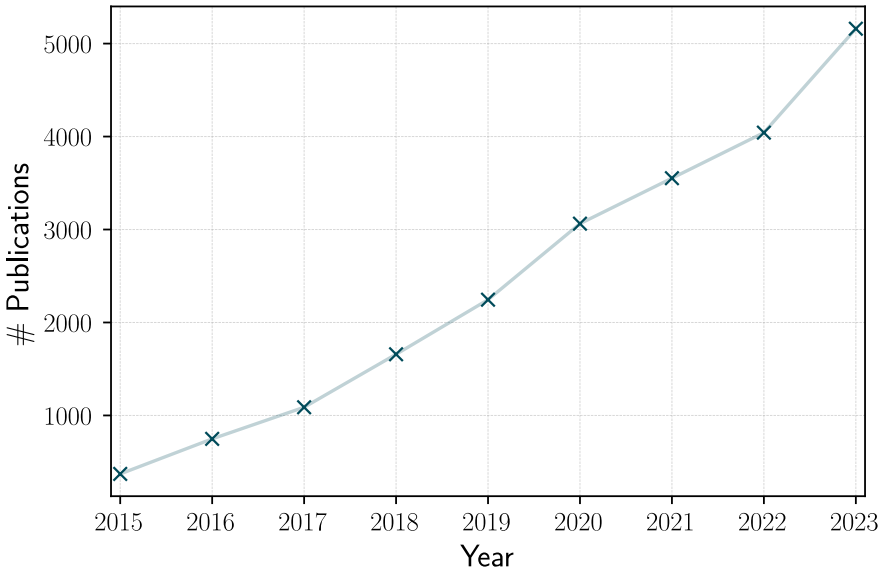


Fig. 7.4: Research trend in the field of hand analysis using three-dimensional data in recent years. The data was gathered using the Dimensions software [Digital Science, 2018] with the following search query: *hand AND pose AND estimation AND 3d*. The following source was considered: *arXiv*, as well as the fields of research: *'Computer Vision and Multimedia Computation', 'Machine Learning', 'Graphics, Augmented Reality and Games', 'Artificial Intelligence', 'Information and Computing Sciences'*.

7.3 Recent Developments and Outlook

The primary focus of this thesis was on leveraging spatial information in 3D point clouds for hand analysis using geometric deep learning methods. Figure 7.4 illustrates the number of publications in the field of hand pose estimation with 3D data over recent years, highlighting the growing research activity in this area and underscoring the significance of this scientific field. In the final section of this thesis, the current research direction will be briefly outlined and open questions in this area will be discussed.

Current research directions. In hand pose estimation, the processing of depth information continues to play a significant role. Depth map information is often used instead of a point cloud that offer a grid structure, allowing for different processing methods. However, they limit data processing to data of a single sensor and do not readily allow for the fusion of data from multiple sensors. The most common underlying strategy involves Transformer [Vaswani et al., 2017] networks that employ attention mechanisms to focus the network on critical areas of the hand. For instance, Rezaei et al. [2023] extract features from the depth map and process them through three different branches. The *UV Branch* handles the 2D localization of the respective joints, the *Attention Branch* analyzes essential areas for determining the depth of the joints

and the *Depth Branch* generates a depth feature map, enabling the determination of joint depth through attention-based processing.

Similarly, Kanis et al. [2023] initially use a CNN to extract features from the depth map. These features are then processed in a Transformer architecture to determine the hand pose. The authors first generate an estimation for the entire image, subsequently crop the area around the pose and finally determine the precise pose. This approach initially addresses the problem of hand localization and then conducts hand pose estimation, thus solving two different tasks at the same time.

The use of gestures in human-machine interfaces is currently multifaceted. In the virtual reality domain, recent studies indicate that a controller-oriented solution offers users higher precision, making it the most favored choice for object selection and manipulation in subjective perception [Rantamaa et al., 2023]. This preference is attributed to higher inaccuracies associated with hand tracking variants, enabling the assumption that ongoing research in more robust methods may alter this perception in future studies. Furthermore, there are approaches aiming to combine hand information with gaze direction to achieve a quicker and more precise mode of interaction [Zhao et al., 2023]. Outside the domain of virtual reality, gestures also find applications in the 2D imaging field within various applications. For instance, on the iPhone, gestures can be utilized to trigger visual reaction effects during a video call [Apple Inc, 2024].

Limitations and Outlook Although the novel methods we presented in the geometric deep learning domain have generated various beneficial outcomes and successfully addressed different tasks on point clouds, they also come with certain limitations and open questions that need to be investigated in the future. These will be briefly discussed in this final section of the thesis.

Each method presented in this thesis demonstrated excellent results in an experimental environment, showcasing their contributions to the respective tasks. However, the data and metrics used in these environments might not directly reflect the conditions in a real-world application, necessitating further investigations for real-world validation. The methods presented in Chapters 3 and 5 showcased their accuracy for 3D hand pose estimation on public, comparatively small datasets. Since the results in supervised learning methods generally depend on both the quality and quantity of the underlying datasets, it is desirable for a real-world application to expand the data or adapt training for certain specific scenarios. This leads to a reduction of the amount of unseen data while running the application and ensures optimal results. Although capturing three-dimensional hand data is trivial due to the high density of available depth sensors, manually generating annotations required for supervised training is a time-consuming task. Therefore, it might be desirable to explore other options.

One possibility is attaching small magnetic sensors to the hand to create a tracking system [Yuan et al., 2017]. After calibration, the keypoint positions can be automatically calculated. Cables and sensors should be as small as possible to blend into the typical

noise level of depth calculation of the 3D cameras and remain invisible. Another option is to train methods unsupervised, avoiding the use of ground truth keypoint positions for training. Wan et al. [2019] render a hand model created from spheres and align it with the depth map of the incoming frame to measure the accuracy of pose estimation. In this context, a hybrid approach is also imaginable, utilizing existing datasets with annotated poses for initial training and subsequently optimizing the method through unsupervised training on additional data without ground truth positions. Reinforcement learning can also be applied as a strategy to avoid immediate dependence on labeled data. However, existing methods for hand pose estimation that leverage this strategy often still use ground truth positions in their reward system [Chen et al., 2023; Yang et al., 2022b].

The method in Chapter 6 achieved excellent accuracy in gesture recognition when provided with our generated hand pose estimations. However, it is desirable to evaluate the experimental setup in the context of a potential real-world application. If a system is intended to verify a single user input based on a gesture, the conducted evaluation reflects this scenario and may potentially be expanded to include further gestures. For continuous human-machine communication, it is generally necessary for the machine to distinguish specific gestures from ordinary hand movements. For such applications, it is beneficial to perform training and evaluation on datasets containing sequences of regular hand movements from subjects, labeled as *no gesture*.

Chapter 5 demonstrated that incorporating temporal information into hand pose estimation significantly improves accuracy, yet the potential of this strategy has not been fully explored. In the presented evaluation, only past frames were considered to enhance the current time point. Inclusion of future frames has the potential to further enhance hand pose estimation. In a possible application, this would mean that a pose is output with a delay as more frames need to be processed. However, depending on the application scenario, this delay may be acceptable.
The method presented already achieves excellent results through temporal analysis of predicted hand poses, making it largely independent of the pose estimation methodology. If we allow ourselves to focus on a specific method as backbone for spatial feature extraction, an open research question is to investigate the potential of the temporal analysis of these features for hand pose estimation.

All the methods presented in this thesis have demonstrated favorable outcomes by processing point clouds extracted from the data of a single depth sensor. One significant advantage of these methods is the use of fundamental geometric deep learning techniques and their associated independence from a grid structure, making the presence of a depth map unnecessary at any given time. Assuming that the respective point clouds from multiple sensors can be fused into a global cloud, this enables the processing of information from multiple points of view without further adjustment of the approaches. Therefore, an additional open research question involves exploring the potential of a multi-sensor setup for the presented tasks on hand analysis.

References

[Alldieck et al., 2017] Alldieck, T., Kassubeck, M., Wandt, B., Rosenhahn, B., and Magnor, M. "Optical Flow-Based 3D Human Motion Estimation from Monocular Video". In: *Pattern Recognition*. Springer International Publishing, 2017, pp. 347–360.

[Ao et al., 2021] Ao, S., Hu, Q., Yang, B., Markham, A., and Guo, Y. *SpinNet: Learning a General Surface Descriptor for 3D Point Cloud Registration*. 2021.

[Apple Inc, 2024] Apple Inc, *iPhone User Guide*. `https://support.apple.com/en-bw/guide/iphone/iphaa0b5671d/ios`. Accessed: 2024-01-04. 2024.

[Arko et al., 2022] Arko, A. R., Little, J. J., and Yi, K. M. *Bootstrapping Human Optical Flow and Pose*. 2022. arXiv: `2210.15121 [cs.CV]`.

[Barsoum, 2016] Barsoum, E. "Articulated Hand Pose Estimation Review". *CoRR* abs/1604.06195, 2016.

[Battrawy et al., 2019] Battrawy, R., Schuster, R., Wasenmüller, O., Rao, Q., and Stricker, D. "LiDAR-Flow: Dense Scene Flow Estimation from Sparse LiDAR and Stereo Images". *CoRR* abs/1910.14453, 2019. arXiv: `1910.14453`.

[Behl et al., 2018] Behl, A., Paschalidou, D., Donné, S., and Geiger, A. "PointFlowNet: Learning Representations for 3D Scene Flow Estimation from Point Clouds". *CoRR* abs/1806.02170, 2018. arXiv: `1806.02170`.

[Besl et al., 1992] Besl, P. and McKay, N. D. "A method for registration of 3-D shapes". *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14 (2), 1992, pp. 239–256.

[Bhatnagar et al., 2020] Bhatnagar, B. L., Sminchisescu, C., Theobalt, C., and Pons-Moll, G. "LoopReg: Self-supervised Learning of Implicit Surface Correspondences, Pose and Shape for 3D Human Mesh Registration". In: *Neural Information Processing Systems (NeurIPS)*. 2020.

[Bigalke et al., 2021] Bigalke, A. and Heinrich, M. P. "Fusing Posture and Position Representations for Point Cloud-Based Hand Gesture Recognition". In: *2021 International Conference on 3D Vision (3DV)*. 2021, pp. 617–626.

[Bogo et al., 2017] Bogo, F., Romero, J., Pons-Moll, G., and Black, M. J. "Dynamic FAUST: Registering Human Bodies in Motion". In: *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[Bowman, 2002] Bowman, D. "Chapter 15 – Principles for the Design of Performance-oriented Interaction Techniques", 2002.

[Bronstein et al., 2016] Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P. "Geometric deep learning: going beyond Euclidean data". *CoRR*, 2016. eprint: `1611.08097`.

[Bronstein et al., 2021] Bronstein, M. M., Bruna, J., Cohen, T., and Velickovic, P. "Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges". *CoRR* abs/2104.13478, 2021. arXiv: `2104.13478`.

[Buchmann et al., 2004] Buchmann, V., Violich, S., Billinghurst, M., and Cockburn, A. "FingARtips: Gesture Based Direct Manipulation in Augmented Reality". In: *Proceedings of the 2nd International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia*. GRAPHITE '04. Association for Computing Machinery, 2004, pp. 212–221.

[Buckingham, 2021] Buckingham, G. "Hand tracking for immersive virtual reality: opportunities and challenges". *Frontiers in Virtual Reality* 2, 2021, p. 728461.

[Chatzis et al., 2020] Chatzis, T., Stergioulas, A., Konstantinidis, D., Dimitropoulos, K., and Daras, P. "A comprehensive study on deep learning-based 3D hand pose estimation methods". *Applied Sciences* 10 (19), 2020.

[Chen et al., 2018a] Chen, X., Wang, G., Guo, H., and Zhang, C. "Pose Guided Structured Region Ensemble Network for Cascaded Hand Pose Estimation". *Neurocomputing*, 2018.

[Chen et al., 2018b] Chen, X., Wang, G., Zhang, C., Kim, T.-K., and Ji, X. "SHPR-Net: Deep Semantic Hand Pose Regression From Point Clouds". *IEEE Access* 6, 2018, pp. 43425–43439.

[Chen et al., 2019a] Chen, M., Tang, Y., Zou, X., Huang, K., Li, L., and He, Y. "High-accuracy multi-camera reconstruction enhanced by adaptive point cloud correction algorithm". *Optics and Lasers in Engineering* 122, 2019, pp. 170–183.

[Chen et al., 2019b] Chen, Y., Zhao, L., Peng, X., Yuan, J., and Metaxas, D. N. "Construct Dynamic Graphs for Hand Gesture Recognition via Spatial-Temporal Attention". In: *BMVC*. 2019.

[Cheng et al., 2020] Cheng, K., Zhang, Y., He, X., Chen, W., Cheng, J., and Lu, H. "Skeleton-Based Action Recognition With Shift Graph Convolutional Network". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 180–189.

[Chen et al., 2022] Chen, J., Yan, M., Zhang, J., Xu, Y., Li, X., Weng, Y., Yi, L., Song, S., and Wang, H. "Tracking and Reconstructing Hand Object Interactions from Point Cloud Sequences in the Wild". *arXiv preprint arXiv:2209.12009*, 2022.

[Chen et al., 2023] Chen, M., Shuang, F., Li, S., and Liu, X. "ASCS-Reinforcement Learning: A Cascaded Framework for Accurate 3D Hand Pose Estimation". In: *Proceedings of the 2023 ACM International Conference on Multimedia Retrieval*. ICMR '23. Association for Computing Machinery, 2023, pp. 335–342.

[Chizat et al., 2016] Chizat, L., Peyré, G., Schmitzer, B., and Vialard, F.-X. *Scaling Algorithms for Unbalanced Transport Problems*. 2016.

[Cho et al., 2014] Cho, K., Merrienboer, B., Gülçehre, Ç., Bougares, F., Schwenk, H., and Bengio, Y. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". *CoRR* abs/1406.1078, 2014. arXiv: 1406.1078.

[Clough et al., 2020] Clough, S. and Duff, M. C. "The Role of Gesture in Communication and Cognition: Implications for Understanding and Treating Neurogenic Communication Disorders". *Frontiers in Human Neuroscience* 14, 2020.

[Corso et al., 2020] Corso, G., Cavalleri, L., Beaini, D., Liò, P., and Velickovic, P. "Principal neighbourhood aggregation for graph nets". *CoRR* abs/2004.05718, 2020. eprint: 2004.05718.

[Cuturi, 2013] Cuturi, M. "Sinkhorn Distances: Lightspeed Computation of Optimal Transport". In: *Advances in Neural Information Processing Systems*. Vol. 26. Curran Associates, Inc., 2013.

[Dai et al., 2019] Dai, Z., Yang, Z., Yang, Y., Carbonell, J. G., Le, Q. V., and Salakhutdinov, R. "Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context". *CoRR* abs/1901.02860, 2019. arXiv: 1901.02860.

[De Smedt et al., 2016] De Smedt, Q., Wannous, H., and Vandeborre, J.-P. "Skeleton-Based Dynamic Hand Gesture Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2016, pp. 1206–1214.

[De Smedt et al., 2017] De Smedt, Q., Wannous, H., Vandeborre, J.-P., Guerry, J., Le Saux, B., and Filliat, D. "SHREC'17 Track: 3D Hand Gesture Recognition Using a Depth and Skeletal Dataset". In: *3DOR - 10th Eurographics Workshop on 3D Object Retrieval*. 2017, pp. 1–6.

[Deng et al., 2022] Deng, B., Yao, Y., Dyke, R. M., and Zhang, J. "A Survey of Non-Rigid 3D Registration". *Computer Graphics Forum* 41 (2), 2022, pp. 559–589. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14502.

[Devlin et al., 2019] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, 2019, pp. 4171–4186.

[Digital Science, 2018] Digital Science, *Dimensions [Software]*. available from `https://app.dimensions.ai`. Accessed: 2024-01-02, under licence agreement. 2018.

[Ding et al., 2022] Ding, L., Dong, S., Xu, T., Xu, X., Wang, J., and Li, J. "FH-Net: A Fast Hierarchical Network for Scene Flow Estimation on Real-World Point Clouds". In: *Computer Vision – ECCV 2022*. Springer Nature Switzerland, 2022, pp. 213–229.

[Du et al., 2019] Du, K., Lin, X., Sun, Y., and Ma, X. "Crossinfonet: Multi-task information sharing based hand pose estimation". *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 2019-June, 2019, pp. 9888–9897.

[Fan et al., 2019] Fan, H. and Yang, Y. *PointRNN: Point Recurrent Neural Network for Moving Point Cloud Processing*. 2019.

[Feng et al., 2021] Feng, W., Zhang, J., Cai, H., Xu, H., Hou, J., and Bao, H. "Recurrent Multi-view Alignment Network for Unsupervised Surface Registration". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[Ge et al., 2018a] Ge, L., Cai, Y., Weng, J., and Yuan, J. "Hand PointNet: 3D Hand Pose Estimation Using Point Sets". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[Ge et al., 2018b] Ge, L., Ren, Z., and Yuan, J. "Point-to-point regression pointnet for 3D hand pose estimation". In: *Computer Vision – ECCV 2018*. Springer International Publishing, 2018, pp. 489–505.

[Gojcic et al., 2021] Gojcic, Z., Litany, O., Wieser, A., Guibas, L. J., and Birdal, T. "Weakly Supervised Learning of Rigid 3D Scene Flow". *CoRR* abs/2102.08945, 2021. arXiv: `2102.08945`.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016.

[Gu et al., 2019] Gu, X., Wang, Y., Wu, C., Lee, Y. J., and Wang, P. "HPLFlowNet: Hierarchical Permutohedral Lattice FlowNet for Scene Flow Estimation on Large-scale Point Clouds", 2019.

[Gu et al., 2022] Gu, X., Tang, C., Yuan, W., Dai, Z., Zhu, S., and Tan, P. "RCP: Recurrent Closest Point for Point Cloud". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 8216–8226.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778.

[Hein et al., 2021] Hein, J., Seibold, M., Bogo, F., Farshad, M., Pollefeys, M., Fürnstahl, P., and Navab, N. "Towards markerless surgical tool and hand pose estimation". *Int J Comput Assist Radiol Surg* 16, 2021, pp. 799–808.

[Hermes et al., 2022] Hermes, N., Hansen, L., Bigalke, A., and Heinrich, M. P. "Support Point Sets for Improving Contactless Interaction in Geometric Learning for Hand Pose Estimation". In: *Bildverarbeitung für die Medizin 2022*. Springer Fachmedien Wiesbaden, 2022, pp. 89–94.

[Hermes et al., 2023] Hermes, N., Bigalke, A., and Heinrich, M. P. "Point cloud-based scene flow estimation on realistically deformable objects: A benchmark of deep learning-based methods". *Journal of Visual Communication and Image Representation* 95, 2023, p. 103893.

[Hermes et al., in press] Hermes, N., Bigalke, A., and Heinrich, M. P. "Incorporating Temporal Information into 3D Hand Pose Estimation using Scene Flow". In: *VISAPP - 19th International Conference on Computer Vision Theory and Applications*. in press.

[Horn et al., 1981] Horn, B. K. and Schunck, B. G. "Determining optical flow". *Artificial Intelligence* 17 (1), 1981, pp. 185–203.

[Hou et al., 2019] Hou, J., Wang, G., Chen, X., Xue, J.-H., Zhu, R., and Yang, H. "Spatial-Temporal Attention Res-TCN for Skeleton-Based Dynamic Hand Gesture Recognition". In: *Computer Vision – ECCV 2018 Workshops*. Springer International Publishing, 2019, pp. 273–286.

[Hu et al., 2021] Hu, T., Lin, G., Han, Z., and Zwicker, M. "Learning to Generate Dense Point Clouds with Textures on Multiple Categories". In: *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE Computer Society, 2021, pp. 2169–2178.

[Hur et al., 2020] Hur, J. and Roth, S. "Self-Supervised Monocular Scene Flow Estimation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[Hur et al., 2021] Hur, J. and Roth, S. "Self-Supervised Multi-Frame Monocular Scene Flow". In: *CVPR*. 2021.

[Ioffe et al., 2015] Ioffe, S. and Szegedy, C. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*. ICML'15. JMLR.org, 2015, pp. 448–456.

[Jin et al., 2022] Jin, Z., Lei, Y., Akhtar, N., Li, H., and Hayat, M. "Deformation and Correspondence Aware Unsupervised Synthetic-to-Real Scene Flow Estimation for Point Clouds". In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 7223–7233.

[Kanis et al., 2023] Kanis, J., Gruber, I., Krňoul, Z., Boháček, M., Straka, J., and Hrúz, M. "MuTr: Multi-Stage Transformer for Hand Pose Estimation from Full-Scene Depth Image". *Sensors* 23 (12), 2023.

[Kingma et al., 2014] Kingma, D. P. and Ba, J. *Adam: A Method for Stochastic Optimization*. 2014.

[Kittenplon et al., 2020] Kittenplon, Y., Eldar, Y. C., and Raviv, D. "FlowStep3D: Model Unrolling for Self-Supervised Scene Flow Estimation". *ArXiv* abs/2011.10147, 2020.

[Lenman et al., 2002] Lenman, S., Bretzner, L., and Thuresson, B. "Using Marking Menus to Develop Command Sets for Computer Vision Based Hand Gesture Interfaces". In: *Proceedings of the Second Nordic Conference on Human-Computer Interaction*. NordiCHI '02. Association for Computing Machinery, 2002, pp. 239–242.

[Lenz et al., 2011] Lenz, P., Ziegler, J., Geiger, A., and Roser, M. "Sparse scene flow segmentation for moving object detection in urban environments". In: *2011 IEEE Intelligent Vehicles Symposium (IV)*. 2011, pp. 926–932.

[Lhommet et al., 2014] Lhommet, M. and Marsella, S. "Expressing Emotion Through Posture". In: 2014, pp. 273–285.

[Li et al., 2019a] Li, S. and Lee, D. "Point-to-pose voting based hand pose estimation using residual permutation equivariant layer". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 11927–11936.

[Li et al., 2019b] Li, Y., Huang, J., Tian, F., Wang, H.-A., and Dai, G.-Z. "Gesture interaction in virtual reality". *Virtual Reality & Intelligent Hardware* 1 (1), 2019, pp. 84–112.

[Li et al., 2019c] Li, Y., He, Z., Ye, X., He, Z., and Han, K. "Spatial temporal graph convolutional networks for skeleton-based dynamic hand gesture recognition". *EURASIP Journal on Image and Video Processing* 2019, 2019.

[Li et al., 2019d] Li, Y., Tofighi, M., Monga, V., and Eldar, Y. C. *An Algorithm Unrolling Approach to Deep Image Deblurring*. 2019.

[Li et al., 2021a] Li, R., Lin, G., He, T., Liu, F., and Shen, C. "HCRF-Flow: Scene Flow from Point Clouds with Continuous High-order CRFs and Position-aware Flow Embedding". In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2021, pp. 364–373.

[Li et al., 2021b] Li, Y., Takehara, H., Taketomi, T., Zheng, B., and Nießner, M. *4dcomplete: Non-rigid motion estimation beyond the observable surface*. 2021.

[Li et al., 2022] Li, Y. and Harada, T. "Lepard: Learning partial point cloud matching in rigid and deformable scenes." *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[Lin et al., 2021] Lin, G., Tang, Y., Zou, X., and Wang, C. "Three-dimensional reconstruction of guava fruits and branches using instance segmentation and geometry analysis". *Computers and Electronics in Agriculture* 184, 2021, p. 106107.

[Liu et al., 2019a] Liu, X., Qi, C. R., and Guibas, L. J. "FlowNet3D: Learning Scene Flow in 3D Point Clouds". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[Liu et al., 2019b] Liu, X., Qi, C. R., and Guibas, L. J. "FlowNet3D: Learning Scene Flow in 3D Point Clouds". *CVPR*, 2019.

[Liu et al., 2019c] Liu, X., Yan, M., and Bohg, J. "MeteorNet: Deep Learning on Dynamic 3D Point Cloud Sequences". In: *ICCV*. 2019.

[Liu et al., 2020] Liu, J., Liu, Y., Wang, Y., Prinet, V., Xiang, S., and Pan, C. "Decoupled Representation Learning for Skeleton-Based Gesture Recognition". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 5750–5759.

[Liu et al., 2021] Liu, X., Yu, S.-y., Flierman, N. A., Loyola, S., Kamermans, M., Hoogland, T. M., and De Zeeuw, C. I. "OptiFlex: Multi-Frame Animal Pose Estimation Combining Deep Learning With Optical Flow". *Frontiers in Cellular Neuroscience* 15, 2021.

[Liu et al., 2023] Liu, J., Wang, X., Wang, C., Gao, Y., and Liu, M. "Temporal Decoupling Graph Convolutional Network for Skeleton-based Gesture Recognition". *IEEE Transactions on Multimedia* PP, 2023, pp. 1–13.

[Loper et al., 2015] Loper, M., Mahmood, N., Romero, J., Pons-Moll, G., and Black, M. J. "SMPL: A Skinned Multi-Person Linear Model". *ACM Trans. Graphics (Proc. SIGGRAPH Asia)* 34 (6), 2015, 248:1–248:16.

[Ma et al., 2011] Ma, Y., Mao, Z.-H., Jia, W., Li, C., Yang, J., and Sun, M. "Magnetic Hand Tracking for Human-Computer Interface". *IEEE Transactions on Magnetics* 47 (5), 2011, pp. 970–973.

[Malik et al., 2019] Malik, J., Elhayek, A., and Stricker, D. "WHSP-Net: A Weakly-Supervised Approach for 3D Hand Shape and Pose Recovery from a Single Depth Image". *Sensors* 19 (17), 2019.

[Maretic et al., 2019] Maretic, H. P., El Gheche, M., Chierchia, G., and Frossard, P. "GOT: An Optimal Transport Framework for Graph Comparison", 2019.

[Mayer et al., 2016] Mayer, N., Ilg, E., Häusser, P., Fischer, P., Cremers, D., Dosovitskiy, A., and Brox, T. "A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 4040–4048.

[Menze et al., 2015a] Menze, M. and Geiger, A. "Object scene flow for autonomous vehicles". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 3061–3070.

[Menze et al., 2015b] Menze, M., Heipke, C., and Geiger, A. "Joint 3D Estimation of Vehicles and Scene Flow". In: *ISPRS Workshop on Image Sequence Analysis (ISA)*. 2015.

[Menze et al., 2018] Menze, M., Heipke, C., and Geiger, A. "Object Scene Flow". *ISPRS Journal of Photogrammetry and Remote Sensing (JPRS)*, 2018.

[Min et al., 2020] Min, Y., Zhang, Y., Chai, X., and Chen, X. "An Efficient PointLSTM for Point Clouds Based Gesture Recognition". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[Mittal et al., 2020] Mittal, H., Okorn, B., and Held, D. "Just Go With the Flow: Self-Supervised Scene Flow Estimation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[Molchanov et al., 2015] Molchanov, P., Gupta, S., Kim, K., and Kautz, J. "Hand gesture recognition with 3D convolutional neural networks". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2015, pp. 1–7.

[Monga et al., 2021] Monga, V., Li, Y., and Eldar, Y. C. "Algorithm Unrolling: Interpretable, Efficient Deep Learning for Signal and Image Processing". *IEEE Signal Processing Magazine* 38 (2), 2021, pp. 18–44.

[Moon et al., 2018] Moon, G., Chang, J., and Lee, K. M. "V2V-PoseNet: Voxel-to-Voxel Prediction Network for Accurate 3D Hand and Human Pose Estimation from a Single Depth Map". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[Movella Inc, 2023] Movella Inc, *Xsens motion capture*. `https://www.movella.com/products/motion-capture`. Accessed: 2023-10-09. 2023.

[Myronenko et al., 2010] Myronenko, A. and Song, X. "Point Set Registration: Coherent Point Drift". *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32 (12), 2010, pp. 2262–2275.

[Nielsen et al., 2004] Nielsen, M., Störring, M., Moeslund, T. B., and Granum, E. "A Procedure for Developing Intuitive and Ergonomic Gesture Interfaces for HCI". In: *Gesture-Based Communication in Human-Computer Interaction*. Springer Berlin Heidelberg, 2004, pp. 409–420.

[Oberweger et al., 2015] Oberweger, M., Wohlhart, P., and Lepetit, V. "Hands deep in deep learning for hand pose estimation". *CoRR* abs/1502.06807 (July), 2015. eprint: `1502.06807`.

[Oberweger et al., 2017] Oberweger, M. and Lepetit, V. “DeepPrior++: Improving Fast and Accurate 3D Hand Pose Estimation”. *CoRR* abs/1708.08325, 2017. arXiv: 1708.08325.

[Ohn-Bar et al., 2014] Ohn-Bar, E. and Trivedi, M. M. “Hand gesture recognition in real time for automotive interfaces: A multimodal vision-based approach and evaluations”. *IEEE transactions on intelligent transportation systems* 15 (6), 2014, pp. 2368–2377.

[Oudah et al., 2020] Oudah, M., Al-Naji, A., and Chahl, J. “Hand Gesture Recognition Based on Computer Vision: A Review of Techniques”. *Journal of Imaging* 6 (8), 2020.

[Owoyemi et al., 2018] Owoyemi, J. and Hashimoto, K. “Spatiotemporal Learning of Dynamic Gestures from 3D Point Cloud Data”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 5929–5934.

[Pfister et al., 2015] Pfister, T., Charles, J., and Zisserman, A. “Flowing ConvNets for Human Pose Estimation in Videos”. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2015.

[Pickering et al., 2007] Pickering, C. A., Burnham, K. J., and Richardson, M. J. “A Research Study of Hand Gesture Recognition Technologies and Applications for Human Vehicle Interaction”. In: *2007 3rd Institution of Engineering and Technology Conference on Automotive Electronics*. 2007, pp. 1–15.

[Poier et al., 2018] Poier, G., Opitz, M., Schinagl, D., and Bischof, H. “MURAUER: Mapping Unlabeled Real Data for Label AUstERity”. *CoRR* abs/1811.09497, 2018. arXiv: 1811.09497.

[Poiesi et al., (early access) 2022] Poiesi, F. and Boscaini, D. “Learning general and distinctive 3D local deep descriptors for point cloud registration”. In: *IEEE Trans. on Pattern Analysis and Machine Intelligence*. (early access) 2022.

[Puy et al., 2020] Puy, G., Boulch, A., and Marlet, R. “FLOT: Scene Flow on Point Clouds Guided by Optimal Transport”. In: *European Conference on Computer Vision*. 2020.

[Qi et al., 2016] Qi, C. R., Su, H., Mo, K., and Guibas, L. J. “PointNet: Deep learning on point sets for 3D classification and segmentation”. *CoRR* abs/1612.00593, 2016. eprint: 1612.00593.

[Qi et al., 2017] Qi, C. R., Yi, L., Su, H., and Guibas, L. J. “PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space”. *CoRR* abs/1706.02413, 2017. arXiv: 1706.02413.

[Quek, 1996] Quek, F. “Unencumbered gestural interaction”. *IEEE MultiMedia* 3 (4), 1996, pp. 36–47.

[Rantamaa et al., 2023] Rantamaa, H.-R., Kangas, J., Kumar, S. K., Mehtonen, H., Järnstedt, J., and Raisamo, R. "Comparison of a VR Stylus with a Controller, Hand Tracking, and a Mouse for Object Manipulation and Medical Marking Tasks in Virtual Reality". *Applied Sciences* 13 (4), 2023.

[Reifinger et al., 2007] Reifinger, S., Wallhoff, F., Ablassmeier, M., Poitschke, T., and Rigoll, G. "Static and dynamic hand-gesture recognition for augmented reality applications". In: *Human-Computer Interaction. HCI Intelligent Multimodal Interaction Environments: 12th International Conference, HCI International 2007, Beijing, China, July 22-27, 2007, Proceedings, Part III 12*. 2007, pp. 728–737.

[Rezaei et al., 2023] Rezaei, M., Rastgoo, R., and Athitsos, V. "TriHorn-Net: A model for accurate depth-based 3D hand pose estimation". *Expert Systems with Applications*, 2023, p. 119922.

[Sagayam et al., 2016] Sagayam, K. M. and Hemanth, D. J. "Hand posture and gesture recognition techniques for virtual reality applications: a survey". *Virtual Reality* 21, 2016, pp. 91–107.

[Sarode et al., 2019] Sarode, V., Li, X., Goforth, H., Aoki, Y., Srivatsan, R. A., Lucey, S., and Choset, H. *PCRNet: Point Cloud Registration Network using PointNet Encoding*. 2019.

[Schwarz et al., 2012] Schwarz, L. A., Mkhitaryan, A., Mateus, D., and Navab, N. "Human skeleton tracking from depth data using geodesic distances and optical flow". *Image and Vision Computing* 30 (3), 2012. Best of Automatic Face and Gesture Recognition 2011, pp. 217–226.

[Shen et al., 2021] Shen, Z., Feydy, J., Liu, P., Curiale, A. H., Estepar, R. S. J., Estepar, R. S. J., and Niethammer, M. *Accurate Point Cloud Registration with Robust Optimal Transport*. 2021.

[Shi et al., 2020] Shi, L., Zhang, Y., Cheng, J., and Lu, H. "Decoupled Spatial-Temporal Attention Network for Skeleton-Based Action-Gesture Recognition". In: *ACCV*. 2020.

[Srivastava et al., 2014] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". *Journal of Machine Learning Research* 15 (56), 2014, pp. 1929–1958.

[Stable Diffusion AI, 2023] Stable Diffusion AI, *Dezgo*. `https://dezgo.com/txt2img`. Accessed: 2023-11-29. 2023.

[Stiefmeier et al., 2006] Stiefmeier, T., Ogris, G., Junker, H., Lukowicz, P., and Troster, G. "Combining Motion Sensors and Ultrasonic Hands Tracking for Continuous Activity Recognition in a Maintenance Scenario". In: *2006 10th IEEE International Symposium on Wearable Computers*. 2006, pp. 97–104.

[Sun et al., 2015] Sun, X., Wei, Y., Liang, S., Tang, X., and Sun, J. “Cascaded hand pose regression”. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 824–832.

[Sun et al., 2018] Sun, D., Yang, X., Liu, M.-Y., and Kautz, J. “PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume”. In: *CVPR*. 2018.

[Sun et al., 2020] Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., Vasudevan, V., Han, W., Ngiam, J., Zhao, H., Timofeev, A., Ettinger, S., Krivokon, M., Gao, A., Joshi, A., Zhang, Y., Shlens, J., Chen, Z., and Anguelov, D. “Scalability in Perception for Autonomous Driving: Waymo Open Dataset”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2020, pp. 2443–2451.

[Tang et al., 2014] Tang, D., Chang, H. J., Tejani, A., and Kim, T.-K. “Latent Regression Forest: Structured Estimation of 3D Articulated Hand Posture”. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 3786–3793.

[Teed et al., 2020] Teed, Z. and Deng, J. “RAFT: Recurrent All-Pairs Field Transforms for Optical Flow”. In: *Computer Vision – ECCV 2020*. Springer International Publishing, 2020, pp. 402–419.

[Tishchenko et al., 2020] Tishchenko, I., Lombardi, S., Oswald, M. R., and Pollefeys, M. *Self-Supervised Learning of Non-Rigid Residual Flow and Ego-Motion*. 2020.

[Titouan et al., 2019] Titouan, V., Courty, N., Tavenard, R., Laetitia, C., and Flamary, R. “Optimal Transport for structured data with application on graphs”. Proceedings of Machine Learning Research 97, 2019, pp. 6275–6284.

[Tokgoz et al., 2003] Tokgoz, A. and Sullivan, J. “A Hypermedia Representation of a Taxonomy of Usability Characteristics in Virtual Environments”. PhD thesis. 2003.

[Tompson et al., 2014] Tompson, J., Stein, M., Lecun, Y., and Perlin, K. “Real-Time Continuous Pose Recovery of Human Hands Using Convolutional Networks”. *ACM Transactions on Graphics* 33, 2014.

[Varol et al., 2017] Varol, G., Romero, J., Martin, X., Mahmood, N., Black, M. J., Laptev, I., and Schmid, C. “Learning from Synthetic Humans”. In: *CVPR*. 2017.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. “Attention is All you Need”. In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017.

[Vedula et al., 1999] Vedula, S., Baker, S., Rander, P., Collins, R., and Kanade, T. "Three-dimensional scene flow". In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. Vol. 2. 1999, 722–729 vol.2.

[Voigt-Antons et al., 2020] Voigt-Antons, J.-N., Kojic, T., Ali, D., and Möller, S. "Influence of Hand Tracking as a Way of Interaction in Virtual Reality on User Experience". In: *2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*. 2020, pp. 1–4.

[Wang et al., 2009] Wang, R. Y. and Popović, J. "Real-Time Hand-Tracking with a Color Glove". *ACM Trans. Graph.* 28 (3), 2009.

[Wan et al., 2017] Wan, C., Probst, T., Van Gool, L., and Yao, A. "Crossing nets: Combining gans and vaes with a shared latent space for hand pose estimation". In: *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*. Vol. 2017-January. IEEE, 2017, pp. 1196–1205. eprint: 1702.03431.

[Wan et al., 2018] Wan, C., Probst, T., Gool, L. V., and Yao, A. "Dense 3D Regression for Hand Pose Estimation". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 5147–5156.

[Wang et al., 2018] Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., and Solomon, J. M. "Dynamic graph CNN for learning on point clouds". *CoRR* abs/1801.07829, 2018. eprint: 1801.07829.

[Wan et al., 2019] Wan, C., Probst, T., Gool, L. V., and Yao, A. "Self-Supervised 3D Hand Pose Estimation Through Training by Fitting". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[Wang et al., 2020] Wang, Z., Li, S., Howard-Jenkins, H., Prisacariu, V., and Chen, M. "FlowNet3D++: Geometric Losses For Deep Scene Flow Estimation". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2020.

[Wang et al., 2021] Wang, H., Pang, J., Lodhi, M. A., Tian, Y., and Tian, D. "FESTA: Flow Estimation via Spatial-Temporal Attention for Scene Point Clouds". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 14173–14182.

[Wang et al., 2022] Wang, G., Hu, Y., Liu, Z., Zhou, Y., Tomizuka, M., Zhan, W., and Wang, H. "What Matters for 3D Scene Flow Network". In: *Computer Vision – ECCV 2022*. Springer Nature Switzerland, 2022, pp. 38–55.

[Wei et al., 2021] Wei, Y., Wang, Z., Rao, Y., Lu, J., and Zhou, J. "PV-RAFT: Point-Voxel Correlation Fields for Scene Flow Estimation of Point Clouds". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 6954–6963.

[Wu et al., 2019] Wu, W., Qi, Z., and Fuxin, L. “PointConv: Deep Convolutional Networks on 3D Point Clouds”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 9613–9622.

[Wu et al., 2020] Wu, W., Wang, Z. Y., Li, Z., Liu, W., and Fuxin, L. “PointPWC-Net: Cost Volume on Point Clouds for (Self-) Supervised Scene Flow Estimation”. In: *European Conference on Computer Vision*. 2020, pp. 88–107.

[Xiong et al., 2019] Xiong, F., Zhang, B., Xiao, Y., Cao, Z., Yu, T., Zhou Tianyi, J., and Yuan, J. “A2J: Anchor-to-Joint Regression Network for 3D Articulated Pose Estimation from a Single Depth Image”. In: *Proceedings of the IEEE Conference on International Conference on Computer Vision (ICCV)*. 2019.

[Xu et al., 2017] Xu, J., Ranftl, R., and Koltun, V. “Accurate Optical Flow via Direct Cost Volume Processing”. In: *CVPR*. 2017.

[Yang et al., 2020] Yang, F., Wu, Y., Sakti, S., and Nakamura, S. “Make Skeleton-Based Action Recognition Model Smaller, Faster and Better”. In: *Proceedings of the ACM Multimedia Asia*. Association for Computing Machinery, 2020.

[Yang et al., 2022a] Yang, H., Yan, D., Zhang, L., Sun, Y., Li, D., and Maybank, S. J. “Feedback Graph Convolutional Network for Skeleton-Based Action Recognition”. *IEEE Transactions on Image Processing* 31, 2022, pp. 164–175.

[Yang et al., 2022b] Yang, J., Bhalgat, Y., Chang, S., Porikli, F., and Kwak, N. “Dynamic Iterative Refinement for Efficient 3D Hand Pose Estimation”. In: *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2022, pp. 2703–2713.

[Yuan et al., 2017] Yuan, S., Ye, Q., Stenger, B., Jain, S., and Kim, T.-K. “BigHand2.2M Benchmark: Hand Pose Dataset and State of the Art Analysis”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[Zhang et al., 2020] Zhang, Z., Xie, S., Chen, M., and Zhu, H. “HandAugment: A Simple Data Augmentation Method for Depth-Based 3D Hand Pose Estimation”. *arXiv*, 2020, arXiv–2001.

[Zhao et al., 2021] Zhao, H., Jiang, L., Jia, J., Torr, P. H., and Koltun, V. “Point transformer”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 16259–16268.

[Zhao et al., 2023] Zhao, M., Pierce, A. M., Tan, R., Zhang, T., Wang, T., Jonker, T. R., Benko, H., and Gupta, A. “Gaze Speedup: Eye Gaze Assisted Gesture Typing in Virtual Reality”. In: *Proceedings of the 28th International Conference on Intelligent User Interfaces*. IUI ’23. Association for Computing Machinery, 2023, pp. 595–606.

[Zhou et al., 2018a] Zhou, Y. and Tuzel, O. "VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2018, pp. 4490–4499.

[Zhou et al., 2018b] Zhou, Y., Lu, J., Du, K., Lin, X., Sun, Y., and Ma, X. "HBE: Hand Branch Ensemble Network for Real-Time 3D Hand Pose Estimation". In: *Computer Vision – ECCV 2018*. Springer International Publishing, 2018, pp. 521–536.

[Zimmermann et al., 2017] Zimmermann, C. and Brox, T. "Learning to Estimate 3D Hand Pose from Single RGB Images". In: *IEEE International Conference on Computer Vision (ICCV)*. https://arxiv.org/abs/1705.01389. 2017.