POLITECNICO DI TORINO

Master's Degree in Data Science and Engineering

Master's Degree Thesis

Egocentric Hand Pose Estimation during Surgical Tool Usage: A Graph Convolution and Transformer-Based Approach

Supervisors

Prof. Giuseppe Bruno AVERTA

PhD. Sophokles KTISTAKIS

Candidate

Andrea PELLEGRINO

July 2025

Summary

Hand pose estimation is the task of determining the precise positions and orientations of a person's hand joints in an image or video. In the context of surgical activities, this involves accurately tracking the hand's movements and its interaction with surgical tools, which is crucial for enhancing mixed reality applications in surgery. Hand pose estimation in surgical activities presents unique challenges that are more complex than those encountered in general hand pose estimation tasks. The difficulty arises from several factors, including the scarcity of annotated surgical data, frequent occlusions caused by surgical tools or the surgeon's body, and visual obstructions such as blood and tissue. These factors significantly complicate the accurate recognition and tracking of hand poses during surgery, making it a critical area of research for enhancing the support provided by mixed reality systems in the operating room.

The goal of this thesis is to develop a 3D hand pose estimation model tailored for medical settings, specifically addressing the challenges mentioned earlier. To achieve this, the thesis utilizes the POV-Surgery dataset, a synthetic dataset created specifically for egocentric hand and tool pose estimation in surgical activities. This dataset is crucial as it offers a wide range of challenging scenarios that closely replicate real surgical environments, capturing the full range of complexities and issues involved in hand and object tracking, including those previously mentioned.

This model integrates the features of Transformers and Graph Convolutional Neural Networks (GCNNs). GCNNs are particularly effective in this context because they treat the hand structure as a graph, where the joints and their connections are represented as nodes and edges, respectively. Transformers, with their attention mechanisms, address the limitations of GCNNs by managing long-range dependencies, which GCNNs alone cannot effectively handle. This integration enhances the model's ability to accurately capture spatial relationships between hand joints, significantly improving pose estimation accuracy. Additionally, several extensions have been explored: incorporating information from previous video frames, reconfiguring the initial graph structure of the hand with intermediate connections, and integrating data about the level of blood present on the hand joints.

The culmination of this research is the development of OHRSA-Net (One Hand Reconstruction during Surgical Activities), a fast, robust and efficient model designed to accurately predict hand poses during surgical tool usage. OHRSA-Net has been rigorously tested on the POV-Surgery dataset, demonstrating its ability to perform under the challenging conditions typical of surgical environments, although the model's ability to predict object meshes remains limited due to the dataset's annotations.

In conclusion, this thesis not only advances the field of hand-pose estimation in surgical activities but also provides a framework for future research aimed at improving surgical tool interaction recognition and the overall support of mixed reality systems in the operating room.

Table of Contents

Chapter 1

Introduction

1.1 Computer Vision for Medical Applications

Computer vision is a branch of artificial intelligence focused on enabling machines to perceive, analyze, and comprehend visual information, such as images and videos. Its primary objective is to empower computers to interpret visual data. Computer vision techniques are employed for several tasks, such as:

- **Image classification**: it involves assigning a label or category to an entire image based on its content. The model analyzes the visual features of the image and classifies it into predefined categories.
- **Object detection**: it is a technique that involves identifying and locating specific objects within an image or video. The model not only recognizes the presence of objects but also determines their precise positions, often by drawing bounding boxes around them.
- **Image Segmentation**: it is the process of partitioning an image into distinct regions, with each region corresponding to different objects or areas of interest. This technique operates at the pixel level, classifying each pixel as belonging to a specific region.
- **Keypoints Extraction**: this technique involves identifying and locating specific points of interest within an image, such as the corners of an object, facial landmarks, or joint positions in a human body. These keypoints are used to understand the shape, pose, or structure of the objects.

Applying these computer vision techniques in the medical domain can significantly advance the capabilities of healthcare professionals. They enable more accurate identification of medical issues, detailed mapping of abnormalities, and

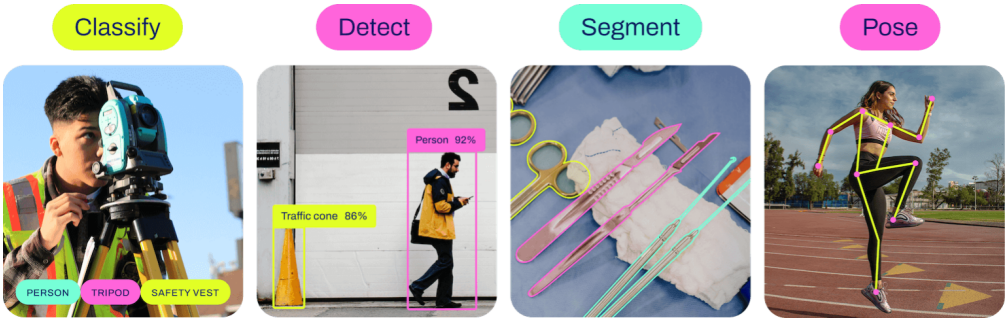


Figure 1.1: Computer vision tasks

in-depth analysis of body structures, thereby supporting more informed and effective treatment strategies. In medical imaging, object detection can help identify where an anomaly, such as a tumor, is located within an image. For example, in a scan of a body part, object detection can highlight the area where a tumor might be, allowing doctors to focus on that specific region for further analysis. Once an anomaly is detected, segmentation can be used to determine its precise shape and boundaries. For instance, in the case of a detected tumor, segmentation will outline the exact shape of the tumor, providing detailed information on its size and form, which is crucial for treatment planning. Keypoint extraction is useful for analyzing detailed anatomical features or tracking changes over time. For example, in orthopedic imaging, keypoint extraction can be applied to track joint positions and angles in a series of X-rays or MRIs, enabling precise assessment of joint alignment or the progression of degenerative diseases.

1.2 Hand pose estimation

Hand pose estimation is a computer vision task that involves predicting the positions and orientations of a hand's joints, either in 2D or 3D space. This technique is essential for understanding hand movements, gestures, and interactions, making it valuable for various applications such as virtual reality and human-computer interaction. Hand pose estimation can be particularly challenging due to the complexity of hand movements and the occlusions that can occur when hands overlap with objects or themselves. In a medical setting, additional challenges must be considered, including the scarcity of annotated surgical data and visual obstructions such as blood and tissue.

In this thesis, hand pose estimation is employed to predict both 2D and 3D hand poses, as well as to generate a 3D mesh of the hand. The process begins with identifying the location of the hand within an image, followed by extracting keypoints that represent the hand's joints in 2D. From these 2D keypoints, the

model then estimates the 3D pose of the hand and constructs a 3D mesh. This mesh is a detailed, three-dimensional representation of the hand's surface, which captures the precise shape and structure of the hand.

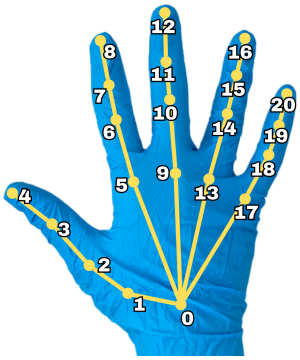


Figure 1.2: Numbered right-hand keypoints

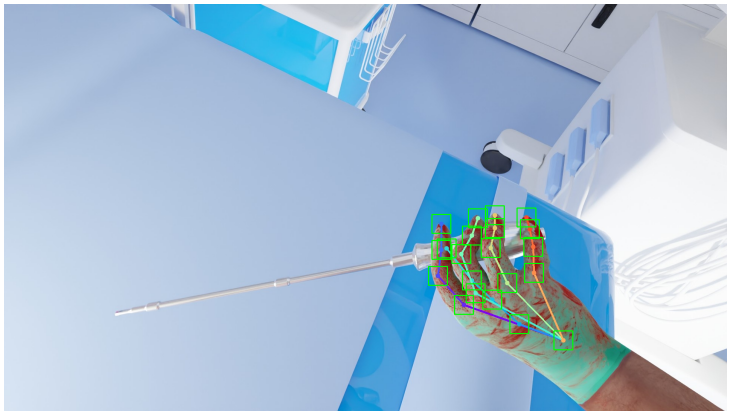


Figure 1.3: 2D projection of the 3D pose to be predicted

1.3 Objective and contributions

This thesis is motivated by the need to develop a reliable hand pose estimation model tailored specifically for surgical activities. Given the unique challenges presented by the surgical environment, there is a clear demand for a specialized approach to hand pose estimation. To address this, an existing model called THOR-Net [1] was selected as a foundation and was further developed and adapted to meet the specific requirements of surgical applications.

The adaptation of THOR-Net model for the surgical domain aims to achieve several key objectives to enhance its effectiveness in this challenging environment:

1. **Accuracy in Hand Pose Estimation**: A crucial goal is to create an accurate model that, despite structural changes, achieves results comparable to or better than the original THOR-Net model. The objective is to maintain or surpass the precision of hand pose estimation while incorporating modifications to handle the specific demands of surgical settings.

2. **Handling Occlusions**: The adapted model is designed to effectively manage various types of occlusions that can occur in surgical settings, such as obstructions caused by surgical instruments, other hands, or visual obstructions like blood stains. The goal is to maintain reliable pose estimation even when parts

of the hand are obscured. Special attention is given to mitigating the impact of blood stains, which can further complicate visual analysis.

3. **Faster Inference**: Achieving real-time performance is crucial for obtaining immediate feedback on hand positioning. Improvements focus on reducing the processing time at each stage of input handling.

The contributions of this thesis include several extensions and improvements to the original THOR-Net model, enhancing its capability for hand pose estimation in the surgical environments. Key contributions are as follows:

1. **Extensions to THOR-Net**:
    1.1. **Multi-Frame Integration**: To address the challenge of hand occlusion in single frames, the model incorporates information from multiple frames. By leveraging data from preceding frames where the hand may be more visible, the model improves the accuracy of hand pose prediction in occluded situations.
    1.2. **Hands Connectivity**: Enhancements to the internal representation of hand structure are introduced by incorporating additional intermediate connections between joints. A more interconnected graph structure facilitates the sharing of information between connected joints, which can be particularly beneficial when a joint is occluded.
2. **YOLOv8 Pose for 2D Keypoint Detection**: The 2D keypoint detection in THOR-Net is enhanced by substituting Keypoint RCNN with YOLOv8 Pose, resulting in OHRSA-Net. This upgrade to a more efficient and faster model significantly boosts detection performance through the application of state-of-the-art computer vision models.
3. **Bloodiness Feature**: The model incorporates a novel feature that quantifies the amount of blood present at each keypoint. This information is integrated with the 2D keypoints to enhance the accuracy of 3D pose and 3D mesh predictions, addressing the challenge of visual obstructions caused by blood and improving the robustness of hand pose estimation.

Chapter 2

Background

2.1 Keypoints extraction

2.1.1 Keypoint RCNN

Keypoint R-CNN is a keypoint detection model built upon the architecture of Mask R-CNN, which itself is an extension of Faster R-CNN. To fully understand Keypoint R-CNN, it's essential to first comprehend the underlying ideas and enhancements introduced by these two preceding models.

Fatser R-CNN

Faster R-CNN [2] is a model that addresses the task of object detection in images. Released in 2015, this model represents a significant advancement in the field by overcoming some of the major limitations of its predecessors, particularly in terms of efficiency. It builts upon the success of earlier models like R-CNN [3] and Fast R-CNN [4], but introduced a novel approach to generating region proposals. The key innovation was the introduction of the Region Proposal Network (RPN), which allowed the model to generate high-quality region proposals quickly and efficiently, all within a single, end-to-end trainable network. Prior to Faster R-CNN, object detection was primarily handled by two-stage pipelines that separated region proposal generation from object classification. The architecture of Faster R-CNN consists of three main components. First, a convolutional backbone network extracts feature maps from the input image. This backbone is typically a deep convolutional network like VGG16 or ResNet, which provides feature maps. Second, the Region Proposal Network operates on these feature maps to generate candidate object proposals. The RPN uses sliding windows over the feature map to predict objectness scores and bounding box coordinates for potential regions of interest. Finally, the proposed regions are refined in the second stage of the network, where

they are classified and their bounding boxes are adjusted.

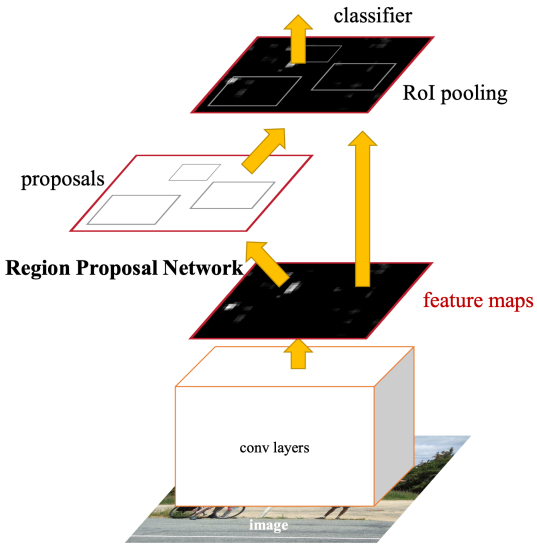


Figure 2.1: Fatser R-CNN architecture

Mask R-CNN and Keypoint R-CNN

The main improvement Mask R-CNN brings over its predecessor, Faster R-CNN, is the addition of a branch for predicting segmentation masks on each Region of Interest (RoI), in parallel with the existing branch for classification and bounding box regression. This allows the model to not only detect objects but also to precisely delineate their boundaries at the pixel level.

Similarly, Keypoint R-CNN further extends Mask R-CNN by incorporating an additional branch dedicated to keypoint detection.

2.1.2 YOLOv8 Pose

YOLO Pose builds on the original YOLO, so understanding YOLO is crucial to understand its enhancements.

YOLO

YOLO [5] (You Only Look Once) presents a fundamentally different approach compared to Faster R-CNN, prioritizing speed and efficiency, making it particularly suitable for real-time applications. While Faster R-CNN employs a two-stage process that first generates region proposals and then classifies them, YOLO adopts a one-stage detection approach. The key idea behind YOLO is to treat object

detection as a single regression problem, directly predicting bounding boxes and class probabilities from the entire image in one evaluation. This direct approach contrasts with the region proposal mechanism in Faster R-CNN and allows YOLO to perform detection much faster, as it does not require the intermediate step of generating proposals. The architecture of YOLO is designed to be both simple and efficient. YOLO divides the input image into a grid, with each grid cell responsible for predicting a set number of bounding boxes and their associated confidence scores, as well as class probabilities. The entire image is processed in a single forward pass through the network, which directly outputs the final detection results. The network backbone uses convolutional layers to extract features, followed by fully connected layers that predict the bounding boxes and class probabilities. This end-to-end approach eliminates the need for complex pipelines and region proposal networks, simplifying the detection process.

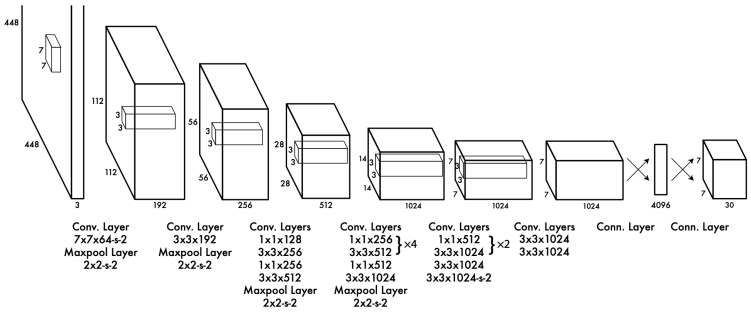


Figure 2.2: YOLO architecture

YOLO Pose

Since its initial release, YOLO has undergone continuous upgrades to enhance its accuracy, efficiency, and versatility across various computer vision tasks. It has been adapted for applications beyond basic object detection, including keypoint detection, leading to the development of the YOLO Pose model.
For this work, it has been used YOLOv8 Pose, the eighth version of YOLO.

2.2 Graph Convolutional Networks and Graph convolutions

To fully understand the THOR-Net and OHRSA-Net models, it's crucial to first comprehend graph convolution, a key operation derived from Graph Convolutional Networks and utilized within the GraFormer module present in both models.

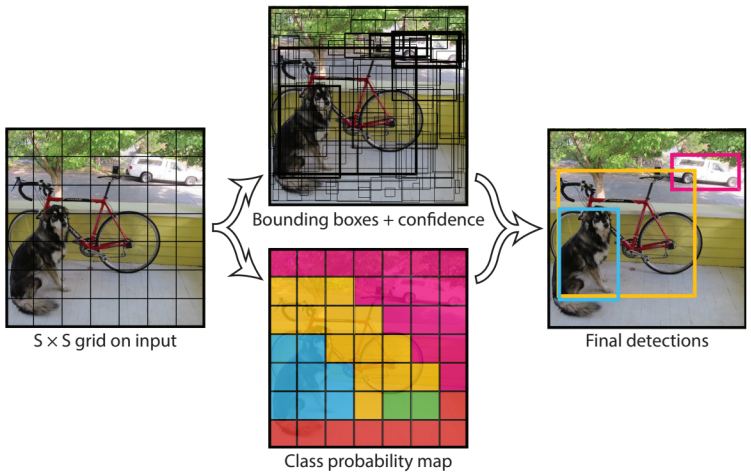


Figure 2.3: YOLO detection process

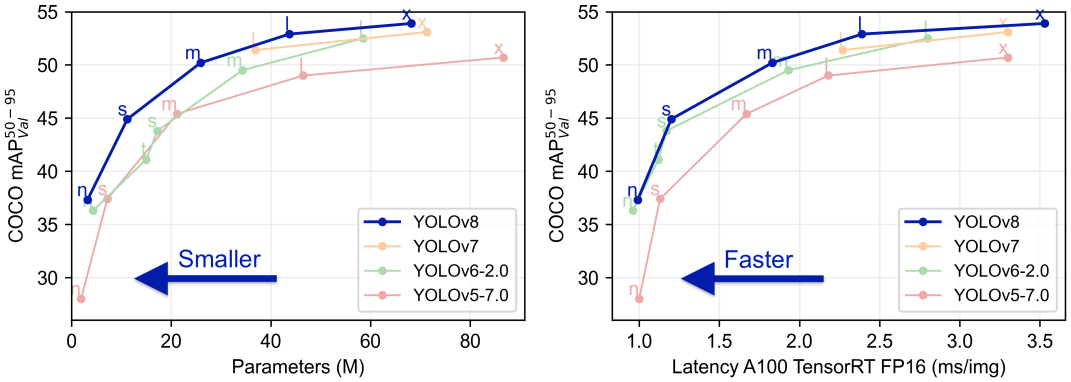


Figure 2.4: Performances of different YOLO versions

2.2.1 Graph Convolutional Networks

Graph Convolutional Networks (GCNs or GCNNs) represent a class of neural networks designed to operate on graph-structured data. Unlike traditional convolutional neural networks, which are optimized for grid-like data such as images, GCNNs are tailored for data represented as graphs, where relationships between data points are more complex.

In this context, each node is associated with a set of features, which provide specific information about the entity it represents. The connections between these nodes, known as edges, define the relationships or interactions between them, forming the structure of the graph that GCNNs operate on.

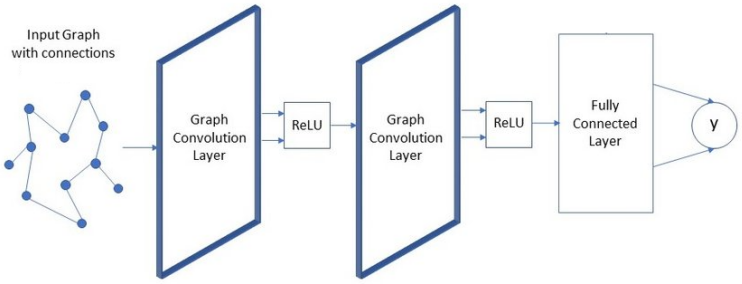


Figure 2.5: Model of the Graph Convolution Neural Network Model

2.2.2 Graph Convolutions

Graph convolution is the primary operation underlying graph neural networks. To understand what a graph convolution consists of, let's consider a graph G with a set of n vertices $\{v_1, \dots v_n\}$, a set of edges connecting these vertices and an associated feature vector x_i for each vertex. The goal of graph convolution is to update these feature vectors by aggregating information from neighboring vertices, allowing each vertex to learn from its local graph structure.

The graph convolution operation for a vertex v_i consists of two main steps:

1. Neighborhood Aggregation: for each vertex v_i, the features from its neighboring vertices and its own features are aggregated. The set $\mathcal{N}(v_i)$ denotes the set of neighbors of v_i, including v_i itself. While various aggregation functions can be employed in different GCNs architectures, for this explanation, we'll consider a simple averaging approach.

   Considering $\mathbf{x_i\prime}$ as the updated feature vector of v_i, the aggregation operation can be written as:

$$\mathbf{x_i\prime} = \frac{1}{|\mathcal{N}(v_i)|} \sum_{j \in \mathcal{N}(v_i)} \mathbf{x}_j$$

2. Feature Transformation: after aggregation, the resulting vector $\mathbf{x}_i\prime$ is transformed using a learnable weight matrix W . This step is analogous to applying a linear layer in traditional neural networks. The transformation is followed by a non-linear activation function (e.g., ReLU) to introduce non-linearity. The operation can be expressed as:

$$\mathbf{x}_i^{(l+1)} = \sigma\left(W^{(l)}\mathbf{x}_i\prime\right)$$

where $\mathbf{x}_i^{(l+1)}$ is the updated feature vector for vertex v_i at the next layer, $W^{(l)}$ is the weight matrix for layer l, and σ is the activation function.

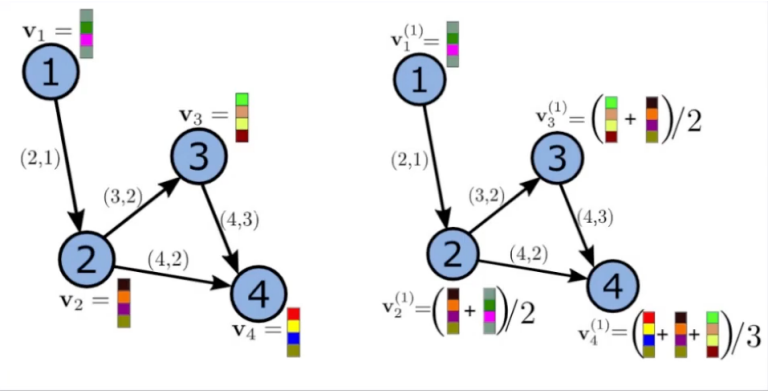


Figure 2.6: Graph Convolution Operation

Generalizing to Matrix Operations

The operations described so far for a single node in a graph convolutional network can be extended to understand how they operate across the entire graph. When dealing with multiple nodes, these operations naturally generalize to matrix operations.

The core function of a single graph convolutional layer, when viewed as matrix operations, is given by:

$$f(X, A) := \sigma(D^{-1/2}(A + I)D^{-1/2}XW)$$

Here, X represents the matrix of node features, A is the adjacency matrix, D is the degree matrix, I is the identity matrix, W is the weight matrix for the convolutional layer, and σ is the activation function.

The core components and operations of a graph convolutional layer can be broken down into the following key steps:

1. Adjacency Matrix Augmentation: The term $A + I$ modifies the adjacency matrix to include self-loops, ensuring that each node's own features are also considered in the aggregation process.

2. Normalization: The normalization step involves the degree matrix D, which scales the adjacency matrix to account for varying node degrees. Specifically, $D^{-1/2}(A + I)D^{-1/2}$ normalizes the adjacency matrix, balancing the influence of neighboring nodes by accounting for their degrees.

3. Feature Aggregation: The product $D^{-1/2}(A + I)D^{-1/2} \cdot X$ performs the aggregation of features from neighboring nodes, effectively pooling information across the graph.

4. Feature Transformation: The subsequent multiplication by the weight matrix W applies the learned transformation to these aggregated features.

5. Activation Function: Finally, applying the activation function σ introduces non-linearity, enabling the network to capture complex relationships within the graph.

2.3 Transformers

Transformers is the core architecture behind GraFormer, which enhances the traditional transformer design by incorporating the graph convolution operations previously described.

2.3.1 An Overview of Transformers

Transformers have become fundamental to modern deep learning, revolutionizing nearly every field, including natural language processing, computer vision, multimodality, audio/speech processing, and signal processing [6]. Since their introduction, transformers have dramatically reshaped the landscape of artificial intelligence, advancing many of today's most influential models, including GPT, BART, DALL-E, CLIP and other state-of-the-art systems.

At a high level, the transformer architecture is designed around a novel approach to handling sequential data. Unlike traditional models that rely on recurrent layers, transformers leverage a mechanism known as self-attention to process input sequences in parallel, allowing for greater scalability and efficiency. This architecture consists primarily of encoder and decoder layers, each built from a series of attention mechanisms and feed-forward neural networks. By moving away from the sequential nature of recurrent layers, transformers can capture long-range dependencies in data more effectively, making them particularly powerful in tasks that require understanding complex relationships within the data. This versatility and power have made transformers the backbone of modern AI systems, enabling a new era of models that can generate text, interpret images, and even create art with unprecedented sophistication.

2.3.2 The Attention Mechanism

Attention is the fundamental mechanism at the heart of transformer models, allowing them to dynamically focus on relevant parts of the input when producing each

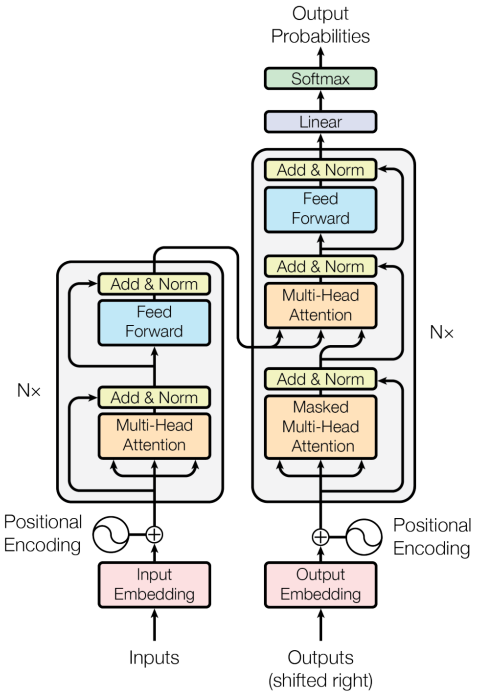


Figure 2.7: Transformer architecture.

element of the output. To understand how it works, let's consider its application in the field of natural language processing, particularly in computing how relevant one word is to another. In the transformer model, self-attention allows each word (or token) in a sentence to weigh the importance of other words when encoding its representation. This mechanism captures dependencies between words, regardless of their position in the sentence.

Computing Attention Scores

Each word in the input sequence is initially embedded as a vector into a continuous space, capturing its semantic meaning. These vectors, denoted as $\mathbf{x}_1$, $\mathbf{x}_2$, ..., $\mathbf{x}_n$ for a sequence of length n, form the basis for further processing.

The attention mechanism utilizes three key elements derived from these input embeddings:

- Queries (Q): These are vectors derived from the input embedding, used to query other words' importance.
- Keys (K): Also derived from the input embeddings, these vectors represent the words being "queried."

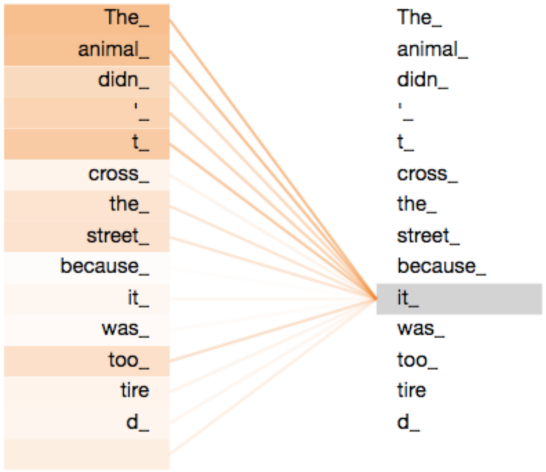


Figure 2.8: Visualization of Self-Attention Scores for the Word "It" in the Example Sentence

- Values (V): These are the actual values derived from input embeddings that are combined to form the output after attention scores are computed.

To compute the attention scores, these steps are followed, starting with the queries, keys, and values:

1. The attention score between two words is computed by taking the dot product of the Query vector of one word with the Key vector of another. For example, when calculating the attention score between words represented by $\mathbf{x}_1$ and $\mathbf{x}_2$, the dot product $\mathbf{q}_1 \cdot \mathbf{k}_2$ determines how much the word associated with $\mathbf{x}_2$ should influence the representation of the word associated with $\mathbf{x}_1$.

2. This score is then divided by the square root of the dimension of the key vectors (denoted as $\sqrt{d_k}$), a scaling factor that stabilizes the gradients during training.

3. The softmax function is applied to these scaled scores to convert them into probabilities that sum to 1, reflecting how much focus each word should receive in the context of the current word.

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax}\left(\frac{\mathbf{Q} \cdot \mathbf{K}^\top}{\sqrt{d_k}}\right) \cdot \mathbf{V}$$

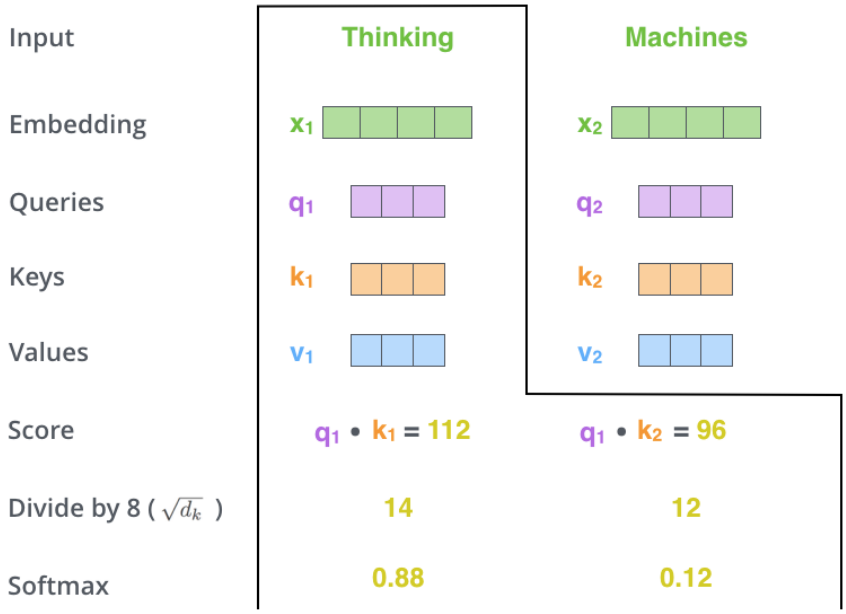


Figure 2.9: Computing Attention Scores for the word "Thinking"

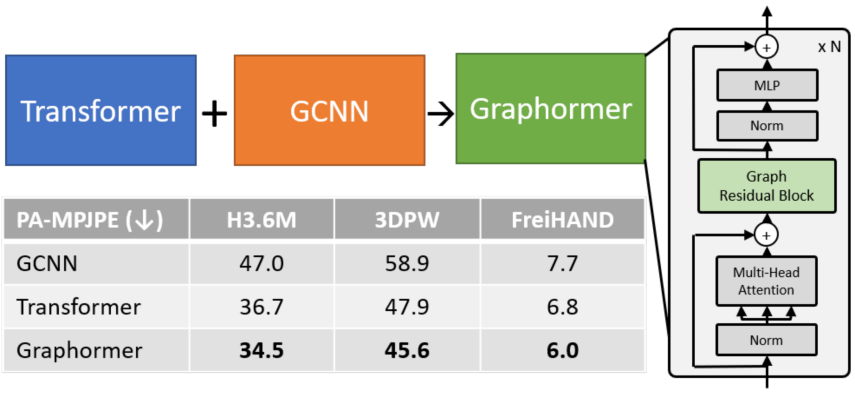
2.4 From Graph-Based Models to GraFormer

One of the primary stages in THOR-Net is the transformation of 2D keypoint predictions into their 3D representations, which is handled by GraFormer, a graph convolution-based transformer that enables the construction of a 3D pose from 2D representations of graph-structured objects

2.4.1 The Idea Behind Integrating Graph Convolutions into Transformer Models

Traditional Graph Convolutional Networks (GCNs) have been widely used for pose estimation due to their ability to handle graph-structured data. However, GCNs are primarily effective at sharing information across first-order neighbors, meaning nodes that are directly connected. As nodes become more distant in the graph, it becomes increasingly difficult to share information effectively, which limits the model's ability to capture relationships between these distant nodes. This limitation becomes especially problematic in the task of hand-pose estimation in a surgical environment, where various types of occlusions can occur more frequently. In such settings, the interactions between distant joints are crucial for accurate 3D reconstruction, making it essential to prioritize and effectively manage these long-range dependencies.

To overcome this challenge, modern deep learning advancements have introduced transformers [7] [8] [9], known for their self-attention mechanism. This mechanism allows the model to effectively consider relationships between elements that are far apart in the input sequence, making it well-suited for managing long-range dependencies. The integration of attention mechanisms with graph convolutional networks has led to the development of models like GraFormer [9] and similar models.



PA-MPJPE (↓)	H3.6M	3DPW	FreiHAND
GCNN	47.0	58.9	7.7
Transformer	36.7	47.9	6.8
Graphormer	**34.5**	**45.6**	**6.0**

Figure 2.10: Performance comparison of GCNN, Transformer, and Graphormer models across mesh reconstrucion datasets [7]

2.4.2 GraFromer

GraFormer is an architecture that combines the strengths of GCNs with the global interaction capabilities of transformers. The core idea behind GraFormer is to leverage the self-attention mechanism of transformers, which can manage long-range dependencies, while preserving the graph structure inherent in the 2D joint data through graph convolutions. By integrating these two powerful techniques, GraFormer is able to more effectively capture both local and global relationships among 2D joints, leading to superior performance in reconstructing accurate 3D poses. GraFormer takes as input a set of N distinct 2D joint coordinates, structured as a graph. The output of the GraFormer model is a set of 3D joint coordinates, represented with dimensions of (N, 3).

At the heart of GraFormer are two key modules: GraAttention and the ChebG-Conv block.

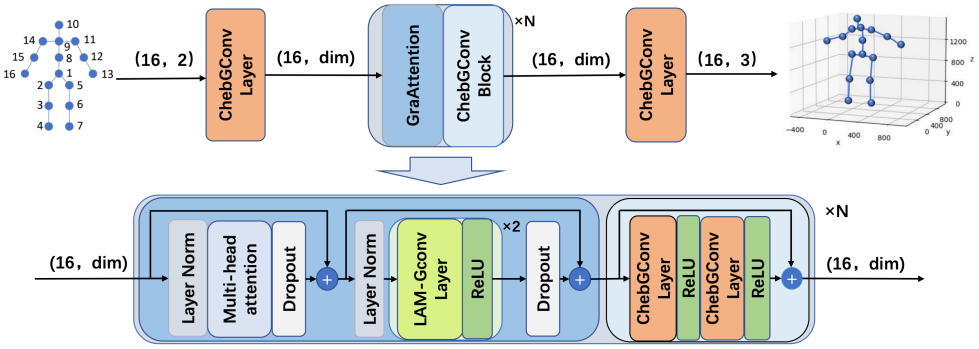


Figure 2.11: GraFormer acrhitecture

GraAttention Module

The GraAttention module is designed to capture both the local and global relationships among 2D joints.

- Multi-Head Attention with Graph Structure: At the core of the GraAttention module is the multi-head self-attention mechanism, inherited from standard transformer architectures. However, unlike standard transformers that treat the input sequence as independent tokens, GraAttention incorporates the inherent graph structure of the joints into the attention mechanism. This is achieved by integrating a learnable adjacency matrix within the attention computation, referred to as LAM-Gconv (Learnable Adjacency Matrix Graph Convolution). This matrix allows the attention mechanism to consider both the similarity of the joints (as in a standard transformer) and their connectivity in the graph, ensuring that the spatial relationships between joints are not lost during the attention process.

- Graph Residual Block: Following the multi-head attention, the GraAttention module includes a residual connection that integrates additional graph convolution operations. This is critical for maintaining the integrity of the graph structure while allowing for deep feature extraction. The residual connection also helps in stabilizing the training process by mitigating issues such as gradient vanishing or exploding, which are common in deep networks. The graph residual block typically involves applying a ChebGConv layer (described below) to further refine the features extracted by the attention mechanism, followed by activation functions like ReLU and normalization layers to enhance learning stability and feature representation.

- Dropout and Layer Normalization: Regularization techniques such as dropout and layer normalization are also employed within the GraAttention module to

prevent overfitting and ensure that the features learned by the model are robust and generalizable. These layers help in balancing the model's complexity and its ability to generalize well to unseen data.

ChebGConv Block

The ChebGConv block is designed to enhance the processing of graph-structured data by extending its ability to capture information from nodes that are several steps away within the graph. This is achieved through Chebyshev polynomials, which allow the model to approximate the graph convolution operation over a larger area, expanding the receptive field beyond immediate neighbors.

This broader reach enables the ChebGConv block to understand and process more complex relationships between joints, crucial for accurate 3D pose reconstruction. By efficiently propagating information across the graph, Chebyshev graph convolution allows the model to capture long-range dependencies, improving 3D pose estimation accuracy. The block introduces non-linearity through ReLU activation functions, enabling the model to learn more intricate patterns, while normalization techniques ensure stable training and performance.

2.5 THOR-Net

Now that all the core building blocks have been presented, it is possible to introduce THOR-Net [1], the foundational model of this thesis.

THOR-Net is a deep learning model designed to reconstruct realistic 3D shapes and poses of two hands interacting with an object from a single RGB image. At the heart of THOR-Net are two critical components: the Keypoint R-CNN and the GraFormer. The Keypoint R-CNN serves as the initial stage of the model, extracting 2D keypoints (or heatmaps) from the input image. These 2D keypoints are then modeled as graphs and passed into the GraFormer network. The GraFormer transforms these 2D keypoints into accurate 3D pose and shape representations. In addition to pose estimation, THOR-Net includes a mesh extraction process, where a coarse-to-fine GraFormer network gradually refines the shape, producing detailed 3D meshes of the hands and the interacting object. By combining the precise feature extraction of Keypoint R-CNN with the robust 2D-to-3D transformation capabilities of the GraFormer, THOR-Net can effectively addresses the challenges of hand-object reconstruction.

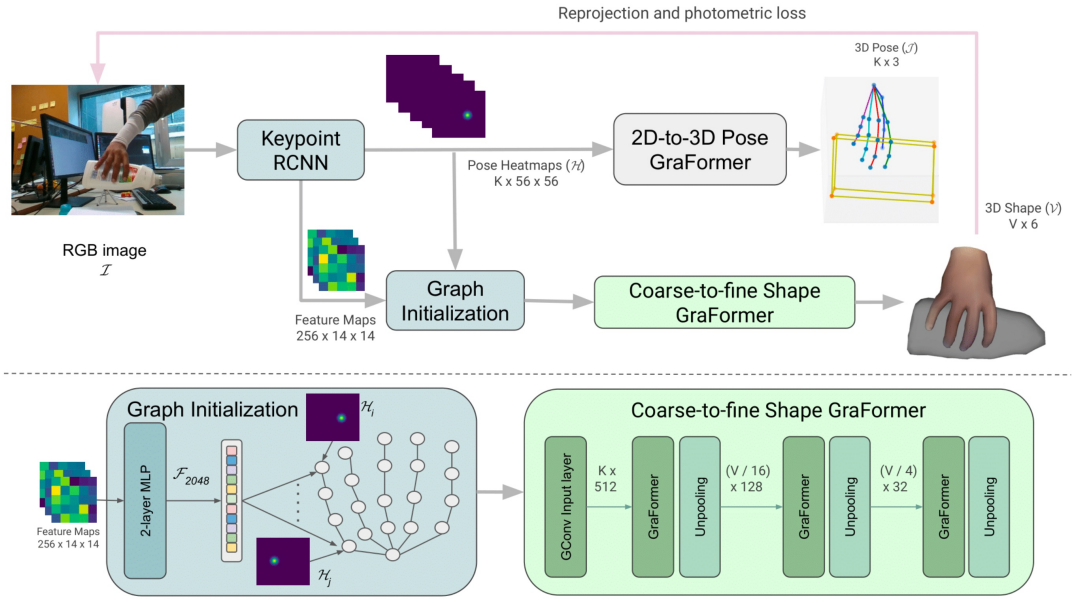


Figure 2.12: THOR-Net acrhitecture

2.5.1 THOR-Net architecture and prediction pipeline

Input and Keypoint Extraction (Keypoint R-CNN)

The process begins with an RGB image, which is fed into the Keypoint R-CNN. This component is responsible for extracting essential 2D features from the image, which could be N distinct 2D coordinates or N heatmaps, one for each keypoint. The Keypoint R-CNN identifies specific points of interest in the hands and objects, creating pose heatmaps (or 2D coordinates) for these keypoints. The feature maps generated at this stage are $256 \times 14 \times 14$ in size, capturing spatial information from the image, which will then be used for graph initialization and, subsequently, 3D mesh reconstruction.

Graph Initialization

The next step involves initializing the graph structure for the hands and object, where the 2D information extracted by the Keypoint R-CNN is organized into a graph format The $256 \times 14 \times 14$ feature maps generated by the Keypoint R-CNN are first passed through a 2-layer Multi-Layer Perceptron, which extracts a condensed 2048-dimensional feature vector for each keypoint. These extracted features, along with the 2D coordinates or heatmaps, are then used to create the initial graph. In this graph, each keypoint and its associated features are treated as nodes, while the connections between them (edges) represent the relationships between different joints in the hands or between the hands and the object.

2D-to-3D Pose Estimation (Pose GraFormer)

The previously extracted keypoint heatmaps are fed into the Pose GraFormer. The Pose GraFormer, as described earlier, is a specialized network that integrates graph convolutional layers and transformer-based attention mechanisms. Its primary role is to convert the 2D pose information from the heatmaps into 3D coordinates. The output of this stage is a set of 3D joint coordinates, which represent the pose of the hands and the object in 3D space.

Coarse-to-Fine Shape Reconstruction (Shape GraFormer)

Parallel to the 3D pose estimation, THOR-Net also performs 3D shape reconstruction using the Coarse-to-Fine Shape GraFormer, a slightly modified version of the vanilla GraFormer. This network takes the initialized graph and keypoint heatmaps and progressively refines them to produce a detailed 3D mesh of the hands and the object. The process begins with a coarse representation, which is gradually refined through several stages, each increasing the resolution and detail of the mesh. Unpooling layers are used between the stages to increase the number of vertices,

allowing the network to capture finer details of the hand shapes and the interacting object.

Reprojection and Photometric Loss

To ensure that the reconstructed 3D poses and shapes align accurately with the original image, THOR-Net employs a reprojection step, where the 3D outputs are projected back into 2D space. This projection is compared with the original input image, and the differences are used to compute a photometric loss, which is an additional loss computed alongside the pose and mesh losses. This loss function plays a crucial role in refining the realism of the reconstructed shapes by directly penalizing discrepancies between the predicted and actual textures of the mesh vertices, ensuring that the reconstructed hands and objects not only have accurate shapes but also realistic textures.

Photometric Loss The process of computing the photometric loss begins with the projection of the 3D hand shape, represented as a mesh, onto the 2D image plane using the camera's intrinsic parameters. This projection maps the 3D vertices of the hand mesh onto corresponding 2D coordinates in the image. For each vertex in the projected 3D mesh, the corresponding pixel in the 2D image is identified, and the RGB color values of these pixels are retrieved. These values represent the true colors of the hand texture as captured by the camera.

Simultaneously, THOR-Net predicts RGB color values for the projected vertices based on its learned texture model. These predicted RGB values are compared with the actual RGB values extracted from the image. The photometric loss is then calculated as the Mean Squared Error (MSE) between the actual RGB values from the target image and the predicted RGB values generated by the model. Mathematically, this is expressed as:

$$L_{photo} = \text{MSE}(I[\text{proj}(V_{gt})], V_{pred,\text{rgb}})$$

Where $I[\text{proj}(V_{gt})]$ refers to the RGB values of the pixels in the target image corresponding to the projected vertices of the ground truth 3D shape, and $V_{pred,\text{rgb}}$ refers to the RGB values predicted by the model for those vertices.

The primary purpose of the photometric loss is to refine the texture of the 3D hand mesh by ensuring that the model's predicted RGB values closely match the actual RGB values observed in the image.

2.6 POV-Surgery Dataset

Dataset Overview

The POV-Surgery dataset [10] is a synthetic, egocentric video dataset designed to advance research in hand and tool pose estimation during surgical activities. Its primary purpose is to aid in developing applications like robot-assisted surgery, surgical navigation systems, and skill assessment tools. The dataset focuses on capturing the surgeon's perspective, providing highly detailed, temporally consistent sequences that simulate real-world surgical environments.

The dataset consists of 53 video sequences, each capturing interactions between a surgeon's hands and surgical tools. These sequences are recorded from an egocentric perspective, simulating the surgeon's point of view during real surgeries. The dataset contains a total of 88,329 frames and provides high-resolution RGB-D video streams, which can be used for model training. To support accurate hand and tool pose estimation, the POV-Surgery dataset includes a comprehensive set of annotations, which are as follows:

- RGB images for visual representation
- Depth information to capture spatial relationships
- Segmentation masks outlining the boundaries of hands and objects
- 3D mesh data for both hands and objects
- 2D annotations specifying hand joint keypoints
- 3D annotations for hand joint keypoints and object bounding boxes

Objects in the Scene

The POV-Surgery dataset includes interactions between a surgeon's hands and three common orthopedic tools: the diskplacer, scalpel, and friem. Each tool is associated with specific hand movements, making the dataset valuable for training models on different types of tool manipulations:

- Scalpel: Used primarily for cutting, requiring precise side-to-side motions.
- Diskplacer: Typically used with a screwing motion, suitable for tasks involving rotation and adjustments.
- Friem: Resembling an awl, this tool demands downward punching motions, adding variability in hand dynamics.

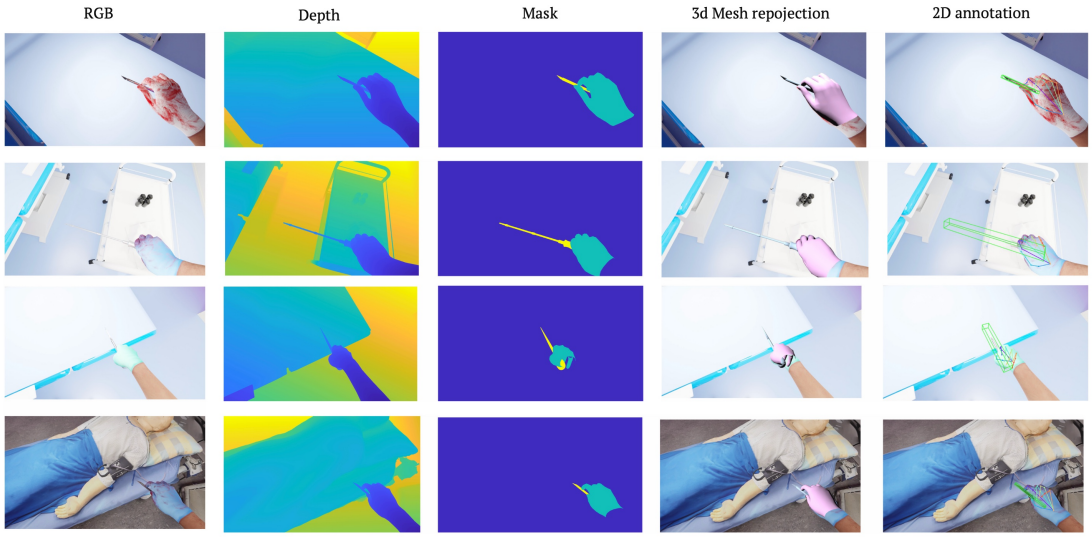


Figure 2.13: Dataset samples for sequences and annotation

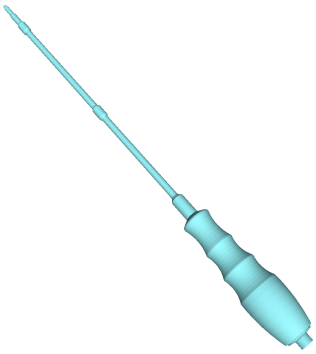


Figure 2.14: Diskplacer

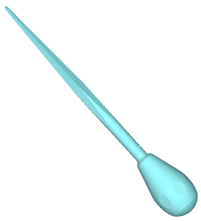


Figure 2.15: Friem

Figure 2.16: Scalpel

As can be seen in Figure 2.17, the distribution of the three tools—scalpel, diskplacer, and friem—is evenly spread across the training and testing sets, ensuring that each object has a similar representation in both splits. This equal distribution is important because it allows models trained on the dataset to encounter a consistent range of hand-object interactions during both training and testing. As a result, it helps to prevent bias toward any specific tool and ensures that model performance can be evaluated fairly across all object types, leading to more reliable and generalizable outcomes. In addition to scenes where tools are present, the dataset also includes frames with no object interactions (hand-only), allowing for the capture of diverse hand manipulations, even in the absence of tools. This

variety makes the dataset versatile for training models on different hand motions, whether or not tools are involved.

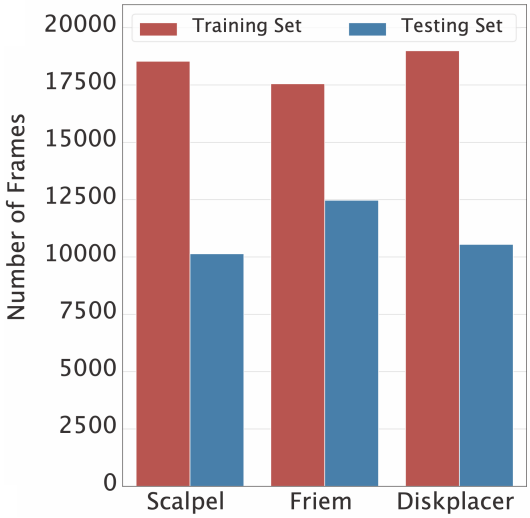


Figure 2.17: Number of frames for each surgical instrument in training and testing sets.

Gloves, Blood and Scene variability

Another key feature of the POV-Surgery dataset is its inclusion of various surgical glove colors to simulate real-world variability. The dataset contains gloves in blue, green, and white, each exhibiting different bloodstain patterns or no blood at all. This variability is crucial for creating realistic surgical scenarios, as in real surgeries, gloves are often obscured by fluids or blood, making hand pose estimation more challenging. Notably, gloves with visible bloodstains are more frequent in the testing set than in the training set, ensuring that the test set better simulates real-world conditions. This strategic distribution helps assess how well models generalize across diverse surgical environments. The combination of different glove colors, blood patterns, and interactions with surgical tools provides a comprehensive range of scenarios for training and testing robust pose estimation models.

Additionally, the dataset includes environmental textures that simulate surgical rooms with varying backgrounds and lighting conditions. In the test set, scenes are more complex, featuring elements such as a surgical bed, patient, and hospital furniture. These added details, visible in the last row of Figure 2.13, are intended to assess how well models generalize to new, unseen conditions, providing a more rigorous evaluation of their ability to handle real-world surgical scenarios.

Chapter 3

Related works

In the domain of hand pose estimation, especially in medical settings, there remains a significant gap in models that are specifically designed to handle the unique challenges posed by surgical environments. As of now no dedicated architectures exist that explicitly target hand pose estimation in these settings. This is critical because surgery introduces additional challenges such as occlusions from surgical tools, variability in lighting and visibility (due to elements like blood or the surgeon's body), and the need for high accuracy in dynamic environments.

Some of the model architectures presented in this chapter were originally designed for generic hand pose estimation tasks using more generalized datasets. While they were not specifically developed for medical scenarios, they have been adapted and tested on POV-Surgery dataset [10], which simulates the unique challenges of surgical environments. These models are included and explained here because they have demonstrated the best performance on the POV-Surgery dataset, making them highly relevant to the context of hand pose estimation in surgical activities.

The first two models discussed are Semi-Hand-Object [11] and HandOccNet [12], both of which have been evaluated on the POV-Surgery dataset, as outlined in the POV-Surgery paper [10]. These models show promising results in handling occlusions, particularly those caused by interactions between hands and objects.

Following this, H2ONet [13] will be presented as a further evolution of HandOccNet. This model enhances the capabilities of its predecessors by incorporating more advanced features, particularly in handling interactions between hands and objects in cluttered, occluded environments.

3.1 Semi-Hand-Object

The Semi-Hand-Object model is designed to address the problem of estimating 3D hand and object poses from monocular RGB images, a task that becomes

significantly more complex during interactions due to frequent occlusions. One of the key issues with such tasks is the lack of sufficient 3D annotations, which limits the performance and generalization of supervised models. To overcome these challenges, Semi-Hand-Object employs a semi-supervised learning framework that leverages large-scale unlabeled video data. By utilizing spatial-temporal consistency and generating pseudo-labels for self-training, the model improves both hand and object pose estimation accuracy, even in scenarios with occlusions and limited 3D ground-truth data.

The core idea behind Semi-Hand-Object is the joint estimation of hand and object poses, where the interactions between the hand and object are leveraged as contextual information to improve the accuracy of pose estimation. By employing a Transformer-based contextual reasoning module, the model can effectively model the relationships between the hand and the object during interaction, using these relations to improve pose estimation.

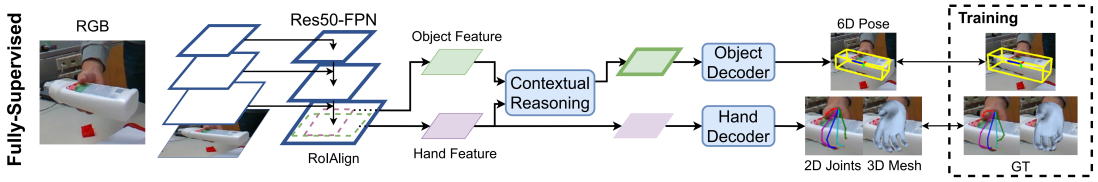


Figure 3.1: Semi-Hand-Object acrhitecture

3.1.1 Semi-Hand-Object architecture

The Semi-Hand-Object model is composed of several key components, including a shared encoder, a contextual reasoning module, and two separate decoders for hand and object pose estimation.

Shared Encoder

The backbone of the model is a ResNet-50-based Feature Pyramid Network, which extracts multi-scale feature maps from the input RGB image. These features provide information for both hand and object pose estimation. To focus on the specific regions of the image that contain the hand and object, RoIAlign is applied, cropping the relevant regions to produce and features and object features. These features are then passed to the next module for contextual reasoning.

Contextual Reasoning

The Contextual Reasoning (CR) module plays a key role in handling occlusions and improving pose estimation. It leverages a Transformer-based attention mechanism to

model hand-object interactions during manipulation tasks, using these interactions as critical contextual information to enhance pose estimation accuracy for both entities.

The CR module processes features from the regions corresponding to the hand and object, with a focus on areas where they intersect. These intersections, often occluded during tasks, are treated as opportunities to infer additional information about the object from the hand's position, and vice versa.

The process starts by using the object's feature map as the query and the hand-object intersecting regions as the key and value. The attention mechanism updates the object's features based on the hand's, producing an enhanced object representation that is passed to the object decoder for pose estimation.

The Transformer architecture in the CR module has two components: an attention module that computes relationships between hand-object feature maps, and a feed-forward module that refines the enhanced features, preparing them for decoding. This ensures that the object features are improved before being passed to the decoders.

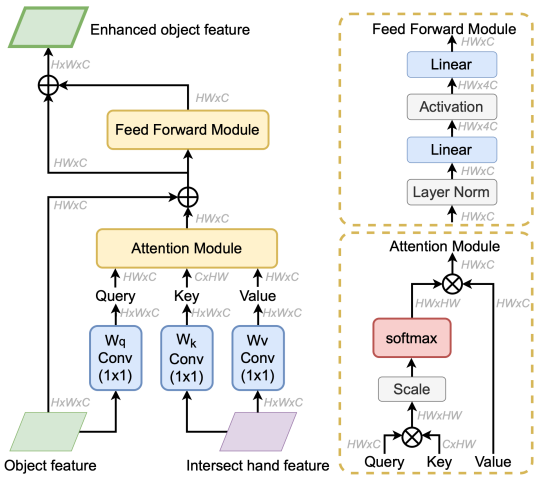


Figure 3.2: Contextual reasoning module

Hand and Object Decoders

After the contextual reasoning stage, the enhanced feature maps are sent to two decoders: one for the hand and one for the object.

The Hand Decoder consists of two main parts: one that predicts 2D joint positions by generating heatmaps, and another that estimates the 3D hand mesh using a model that captures hand shape and pose.

The Object Decoder is responsible for estimating the object pose. It uses

two streams: one to predict the 2D coordinates of control points on the object, and another to assess the confidence of each prediction, ensuring accurate pose estimation.

3.1.2 Semi-Supervised Learning Framework

In hand-object pose estimation, the lack of fully annotated 3D datasets poses challenges for effective model training. To overcome this, the Semi-Hand-Object model employs a semi-supervised learning framework that combines annotated data with pseudo-labels generated from unlabeled videos. By applying spatial-temporal consistency checks and self-training, the model improves its performance, making it more robust in handling occlusions and diverse, real-world environments.

As shown in Figure 3.3, the semi-supervised learning framework consists of the following stages:

1. **Initial Supervised Learning**: The process starts by training the model on an annotated dataset containing fully labeled 3D hand and object poses. This supervised training provides the foundation for the model to estimate hand and object poses during interaction tasks.

2. **Pseudo-Label Generation**: Once trained, the model is deployed on large-scale, unlabeled video datasets that capture hand-object interactions in uncontrolled environments, referred to as "videos in the wild." The model predicts 3D hand and object poses for each frame, generating pseudo-labels that serve as additional training data without the need for human annotation.

3. **Self-Training with Pseudo-Labels**: The model is then retrained using both the original annotated data and the pseudo-labels generated from the video dataset. To ensure the quality of these pseudo-labels, spatial-temporal consistency checks are applied: spatial consistency ensures that the predicted 3D hand and object poses align with their 2D projections in the image, while temporal consistency ensures smooth transitions in hand and object poses across consecutive video frames, filtering out noisy predictions.

4. **Testing and Evaluation**: After self-training, the model is tested to assess improvements. First, the model shows better performance, with improved accuracy in hand-object pose estimation, as seen in comparisons of predictions made before and after the process. Additionally, it demonstrates robust cross-domain generalization, performing well on out-of-domain datasets that were not part of the initial training. This generalization ability is crucial for real-world applications where the model may encounter new environments and objects.

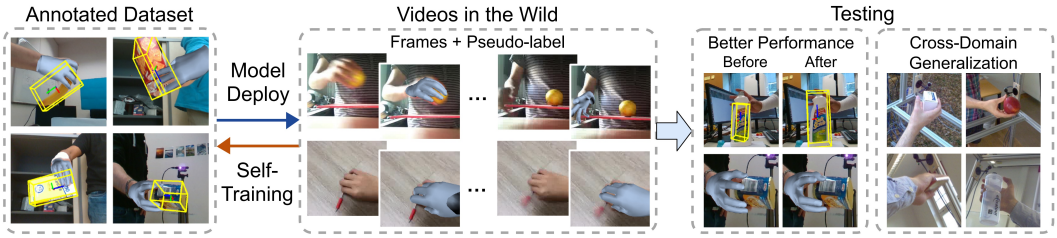


Figure 3.3: Semi-supervised learning framework

3.2 HandOccNet

HandOccNet [12] is a model designed for the 3D hand mesh estimation task in scenarios where significant parts of the hand are occluded by objects, a frequent issue in hand-object interaction tasks. Traditional hand pose estimation models tend to perform poorly under occlusions, as they rely heavily on visible hand regions. HandOccNet overcomes this limitation by not only focusing on visible parts of the hand but also learning to exploit occluded regions to improve overall pose estimation accuracy.

The core innovation of HandOccNet lies in its ability to integrate information from both visible and occluded hand regions using a feature injection mechanism. This mechanism is implemented through two key Transformer-based modules:

- Feature Injecting Transformer (FIT): This module injects hand feature information from visible regions into occluded areas, enriching the feature representation of the occluded regions.
- Self-Enhancing Transformer (SET): After the initial feature injection, the SET module further refines the injected features.

3.2.1 HandOccNet architecture

The HandOccNet architecture consists of four main components: the Backbone, the Feature Injecting Transformer (FIT), the Self-Enhancing Transformer (SET), and the Regressor.

Backbone

The Backbone is responsible for extracting essential features from the input hand image, which has a resolution of 512×512. It uses a ResNet50-based Feature Pyramid Network to generate a feature map with dimensions $32 \times 32 \times 256$. Additionally, a necessity map (M) is computed from the feature map to assess the

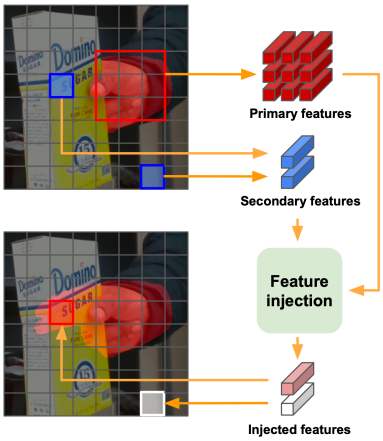


Figure 3.4: Feature injection process operated by FIT

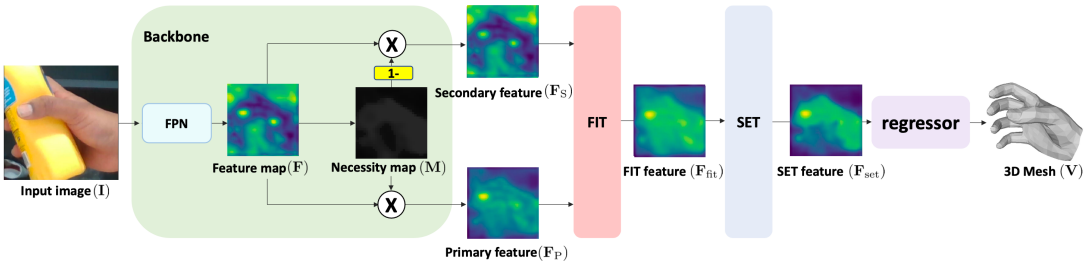


Figure 3.5: HandOccNet architecture

importance of different regions in the image. This map helps divide the feature map into two components:

1. Primary feature (F_P): Contains critical hand information from the visible regions.

2. Secondary feature (F_S): Represents occluded regions where hand details are hidden.

F_P is used as the primary input for hand mesh estimation, while F_S contains information about occluded areas that are not directly used for mesh prediction. However, in later steps, both F_P and F_S are employed as the query, key, and value for the FIT module.

Feature Injecting Transformer (FIT)

The Feature Injecting Transformer (FIT) injects hand feature information from visible regions into the occluded areas to enhance the feature representation of

those occluded regions. The FIT module operates using two attention mechanisms: softmax-based attention and sigmoid-based attention. Both mechanisms contribute to enriching the secondary feature map F_S by leveraging the primary feature map F_P.

The primary feature F_P and secondary feature F_S are processed through two attention mechanisms: the Softmax-Based Attention Module and the Sigmoid-Based Attention Module, as shown in Figure 3.6.

Softmax-Based Attention Module The primary feature F_P is processed into the query q_{soft}, while the secondary feature F_S is transformed into the key k_{soft}. A matrix-matrix multiplication is then performed between q_{soft} and k_{soft}, generating an attention map that captures the relationships between the visible and occluded regions. This attention map is passed through a softmax function to highlight the most relevant areas between these regions, producing the output C_{soft}.

Sigmoid-Based Attention Module Similarly, F_P and F_S are processed into the query q_{sig} and the key k_{sig}, respectively. An element-wise multiplication is performed between q_{sig} and k_{sig}, followed by a fully connected layer and the application of a sigmoid function to filter out irrelevant or noisy information, resulting in the output C_{sig}. This sigmoid-based module provides complementary filtering, preventing overly strong correlations between unrelated features and effectively reducing noise.

Fusion and Output After the softmax-based and sigmoid-based attention modules generate their respective outputs, C_{soft} and C_{sig}, these outputs are combined by summing the two results. The sum is then element-wise multiplied with the value vector v, which contains feature information from both the visible and occluded regions. This process produces the enhanced feature map R_{FIT}.

Finally, the result R_{FIT} passes through a series of fully connected layers, dropout, and normalization, producing the final output F_{FIT}, which contains enriched information from both the visible and occluded hand regions. This enhanced feature map is used in subsequent stages of the model.

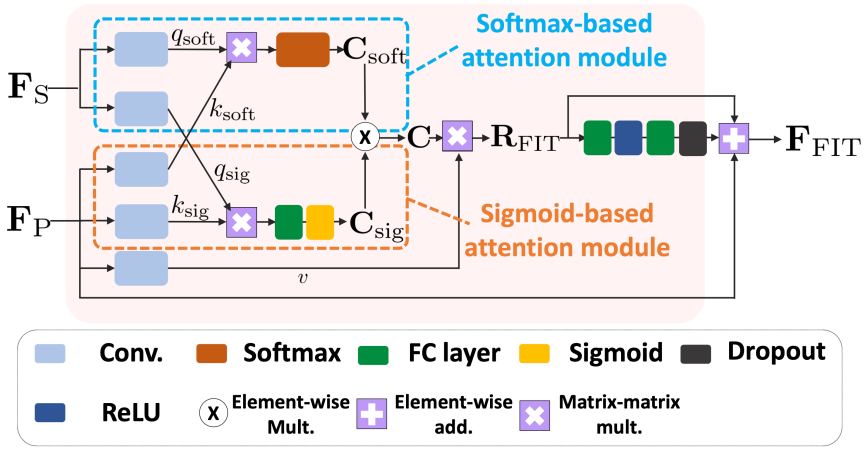


Figure 3.6: FIT module pipeline

Self-Enhancing Transformer (SET)

The Self-Enhancing Transformer (SET) module further refines the feature map generated by the FIT module F_{FIT}. Unlike FIT, which primarily focuses on injecting visible hand information into occluded areas, the SET module focuses on enhancing the entire feature map by considering long-range dependencies within the same feature space. This allows the model to capture relationships across various spatial locations, ensuring that both visible and occluded regions are represented in greater detail.

In the SET module:

1. The feature map F_{FIT} is processed into query q', key k', and value v' vectors.

2. A matrix-matrix multiplication between q' and k' generates an attention map that captures spatial relationships.

3. The attention map is normalized using a softmax function and then element-wise multiplied with the value vector v'.

4. The processed features are passed through fully connected layers, ReLU activations, and dropout layers.

The output is the refined feature map F_{SET}, which captures detailed relationships between visible and occluded regions.

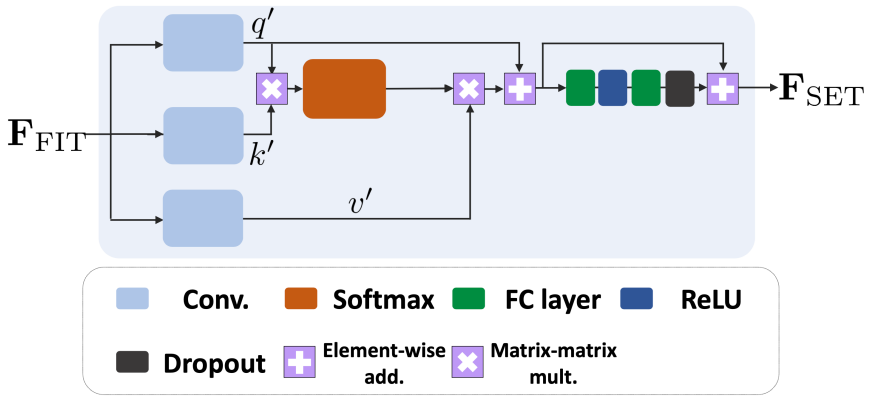


Figure 3.7: SET module pipeline

Regressor

The Regressor takes the refined feature map F_{SET} and predicts the 3D hand mesh by estimating the MANO model parameters (pose and shape). This is done in a two-step process:

1. First, a heatmap for the 2D hand joints is generated.
2. Then, a series of residual blocks concatenate the hand features and the heatmap to produce the final MANO pose and shape parameters.

Once the parameters are predicted, they are fed into the MANO layer, which produces the final 3D hand mesh, representing the full hand pose and structure.

3.3 H2ONet

H2ONet introduces several advanced concepts in hand pose estimation, including a two-branch architecture designed to efficiently leverage non-occluding information from multiple frames, along with novel finger-level and hand-level occlusion-aware feature fusion modules. However, a detailed description of this model will not be provided here for a few key reasons.

The reason for including H2ONet here is its noteworthy multi-frame integration, which serves as a key inspiration for extending THOR-Net in this thesis by incorporating multi-frame analysis to enhance performance in handling occlusions during surgical tool usage. Additionally, H2ONet has not been tested or mentioned in the POV-Surgery paper, which limits its direct relevance to the current focus. The paper also positions H2ONet as being inspired by and closely related to HandOccNet, a model already discussed in detail.

3.3.1 H2ONet Overview

H2ONet [13] introduces a framework for addressing 3D hand mesh reconstruction and orientation prediction, particularly in scenarios where hand-object interactions lead to significant occlusions. The model's architecture is structured around two main modules: Hand Mesh Reconstruction (HMR) and Hand Orientation Regression (HOR), both of which are designed to exploit multi-frame information for better handling of occlusions.

Hand Mesh Reconstruction (HMR)

The HMR module is tasked with generating a detailed 3D hand mesh, even in the presence of occlusions. To achieve this, HMR uses a finger-level occlusion-aware feature fusion mechanism, which integrates information from multiple consecutive frames to selectively utilize non-occluded data. Specifically, for each frame, HMR predicts occlusion probabilities at the finger level, allowing the network to identify which parts of the hand are occluded in the current frame and which are visible. By doing so, the model can aggregate non-occluded information from different frames to reconstruct the full 3D hand mesh more reliably. This approach helps mitigate the occlusion problem, as the model can pull relevant details from past frames where the hand may not be occluded, thus improving the accuracy of the hand shape reconstruction.

Hand Orientation Regression (HOR)

The HOR module focuses on predicting the global orientation of the hand, a task made challenging by occlusions, especially when the entire hand is blocked from view. To address this, HOR utilizes a hand-level occlusion-aware feature fusion mechanism, aggregating orientation information from multiple frames, particularly those where the hand is visible or less occluded. By collecting data from previous frames, the HOR module compensates for moments when the hand is fully or heavily occluded in the current frame. This approach allows HOR to overcome the "ill-posed" problem of estimating the hand's orientation when little to no visual information is available in a single frame.

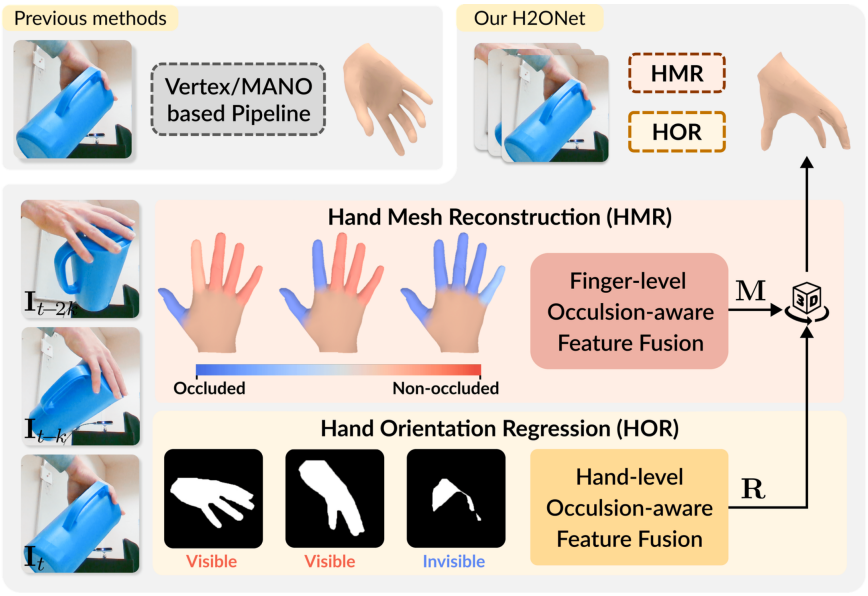


Figure 3.8: H2ONet architecture overview

3.3.2 Multi-frame integration

One of H2ONet's key innovations is its ability to exploit multi-frame integration, using temporal context to enhance both HMR and HOR. By analyzing multiple frames, the model can effectively gather information from visible regions in other frames to compensate for occluded parts. This temporal fusion not only improves hand mesh reconstruction but also ensures more robust and accurate orientation estimation, making the model more resilient to occlusions and noisy data.

The effectiveness of multi-frame integration, as shown in Figure 3.9 from the H2ONet paper, demonstrates significant improvements over single-frame methods. This inspired its inclusion as a key extension in the base THOR-Net model to better address the frequent and varied occlusions encountered during surgical activities. By leveraging multi-frame data, THOR-Net can more effectively manage occlusions caused by tools, instruments, and anatomy, improving hand pose estimation in complex surgical environments.

	Methods	J-PE ↓	J-AUC ↑	V-PE ↓	V-AUC ↑	F@5 ↑	F@15 ↑
SF	Pose2Mesh [13]	12.5	-	12.7	-	44.1	90.9
	I2L-MeshNet [39]	11.2	-	13.9	-	40.9	93.2
	ObMan [20]	11.1	-	11.0	77.8	46.0	93.0
	HO3D *et al.* [16]	10.7	78.8	10.6	79.0	50.6	94.2
	METRO [33]	10.4	-	11.1	-	48.4	94.6
	Liu *et al.* [35]	10.2	79.7	9.8	80.4	52.9	95.0
	I2UV-HandNet [7]	9.9	80.4	10.1	79.9	50.0	94.3
	Tse *et al.* [54]	-	-	10.9	-	48.5	94.3
	HandOccNet [45]	9.1	81.9	<u>9.0</u>	<u>81.9</u>	<u>56.1</u>	<u>96.2</u>
	MobRecon* [8]	9.2	-	9.4	-	53.8	95.7
	MobRecon [8]	9.4	81.3	9.5	81.0	53.3	95.5
	Our H2ONet†	<u>9.0</u>	<u>82.0</u>	<u>9.0</u>	<u>81.9</u>	55.4	96.0
MF	Hasson *et al.* [18]	11.4	77.3	11.4	77.3	42.8	93.2
	Hasson *et al.* [19]	-	-	14.7	-	39.0	88.0
	Liu *et al.* ‡ [35]	9.8	-	9.4	81.2	53.0	95.7
	Our H2ONet	**8.5**	**82.9**	**8.6**	**82.8**	**57.0**	**96.6**

Figure 3.9: Comparison of model performance showing improved hand pose estimation results when using multi-frame integration (MF)versus single-frame approaches (SF)

Chapter 4

Contributions and Methodology

In this chapter, the core contributions of this thesis will be presented, from the initial extensions made to the THOR-Net model to the development of OHRSA-Net (One Hand Reconstruction during Surgical Activities), a novel architecture specifically designed for surgical activities. The primary objective of these contributions is to tackle the persistent challenges posed by occlusions, which frequently occur in surgical environments, as discussed in previous chapters. The extensions made to THOR-Net are aimed not only at improving occlusion handling but also at enhancing the model's overall speed and efficiency.

4.1 Hand connectivity

4.1.1 Overview

One of the key extensions introduced in this thesis is the enhancement of the hand's internal representation within the model, referred to as Hand Connectivity.

In hand pose estimation, the hand can be modeled as a graph where its 21 joints can be represented as nodes, and the connections between them as edges. A visualization of the hand graph is provided in Figure 4.2.

In THOR-Net, defining hand connectivity is essential for transforming 2D keypoints into their corresponding 3D poses. This transformation is carried out by the GraFormer module, which relies on a graph structure modeled to represent the hand. The connections between hand joints in this graph dictate how information is exchanged, both incoming data from connected joints and outgoing data shared with other joints.

The core idea behind this extension is to add additional intermediate edges

between fingers. By expanding the connectivity between the fingers, the model can estimate the position of occluded joints using visible information from other joints in the same frame. For instance, if a finger is partially or completely occluded, the model can infer its position by referencing the positions of other, unobstructed fingers.

However, while adding more connections improves the model's ability to estimate occluded joints, it comes with trade-offs. Introducing too many connections can slow down the processing pipeline of the GraFormer module. The complexity of the graph increases as more connections are introduced, leading to higher computational demands during the transformation from 2D keypoints to 3D poses. Therefore, a balance must be struck between enhancing connectivity to handle occlusions and maintaining the efficiency of the model's inference process.

4.1.2 Implementation Details

Three additional types of hand connectivity have been implemented: SIMPLE, EXTENDED, and FULL hand connectivity. For each type, the corresponding adjacency matrix used as input for the graph in GraFormer will be provided, along with a visualization of the hand. The hand connectivity that models the natural structure of a real hand will be referred to as the BASE connectivity, as shown in Figures 4.1, 4.2.

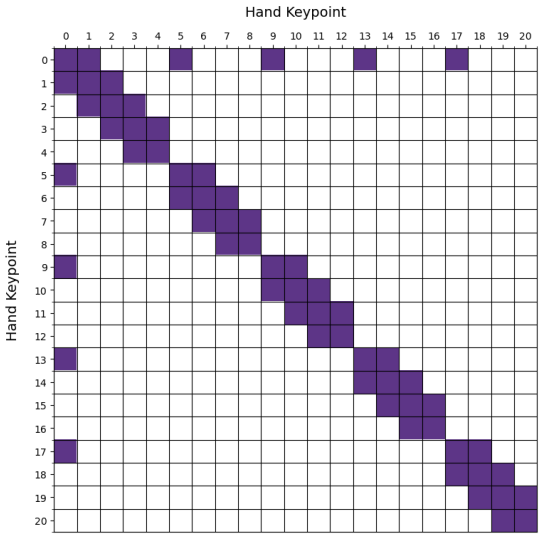


Figure 4.1: BASE adjacency matrix

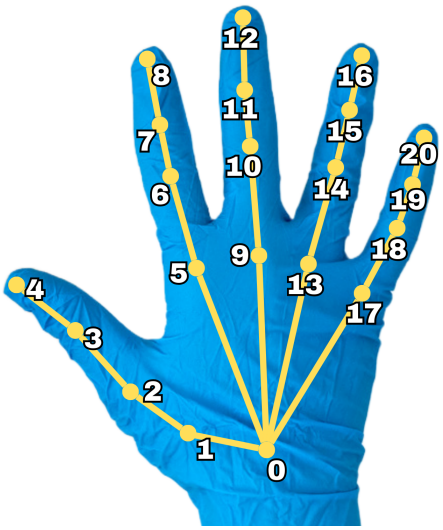


Figure 4.2: BASE hand connectivity

SIMPLE Hand Connectivity The SIMPLE hand connectivity is a slight upgrade of the BASE connectivity. In addition to the connections in the BASE model, this configuration introduces edges between joints located at the same height across each finger. These additional edges enable the sharing of information between corresponding joints on different fingers, allowing the model to better estimate joint positions by leveraging the information from adjacent fingers at similar positions.

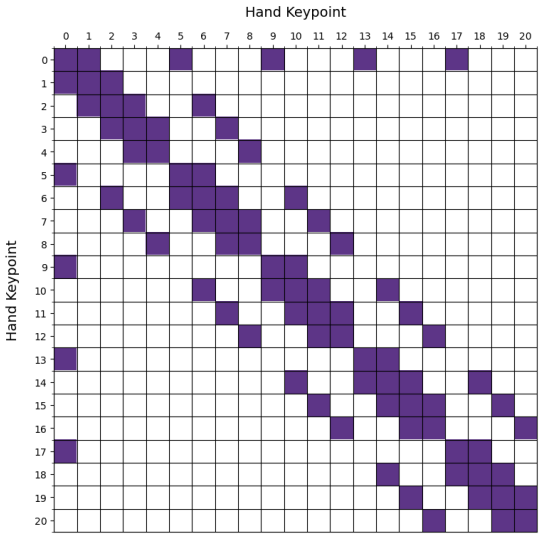


Figure 4.3: SIMPLE adjacency matrix

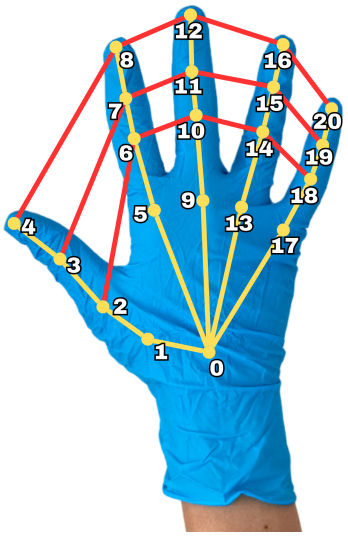


Figure 4.4: SIMPLE Hand Connectivity

EXTENDED Hand Connectivity The EXTENDED hand connectivity builds upon the BASE and SIMPLE configurations. In this structure, for each keypoint on a finger, additional connections are introduced between that joint and every corresponding joint on the neighboring fingers. In Figure 4.6, these additional connections are visualized for keypoint 6 only, but they apply to every keypoint across all fingers.

FULL Hand Connectivity Expanding on the BASE, SIMPLE, and EXTENDED configurations, the FULL hand connectivity introduces further interconnections between each keypoint on a finger and every other keypoint across all fingers. In Figure 4.8, these additional connections are visualized for keypoint 6 only, but they apply to every keypoint across all fingers. This connectivity structure

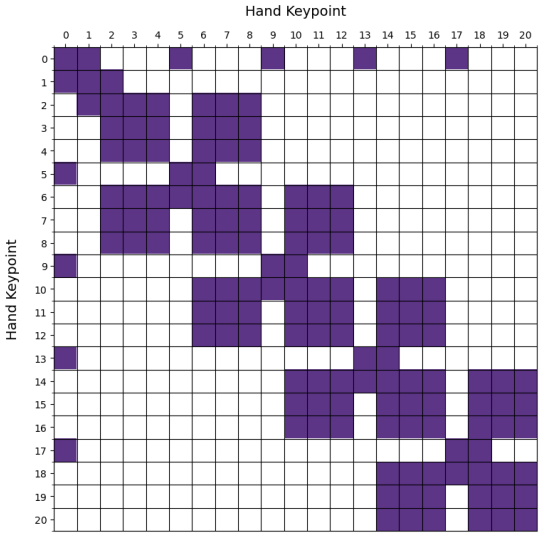


Figure 4.5: EXTENDED adjacency matrix

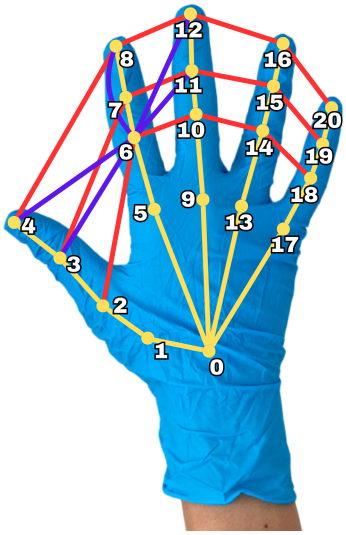


Figure 4.6: EXTENDED Hand Connectivity for keypoint 6

maximizes the information sharing between joints, allowing the model to have the most comprehensive view of the hand.

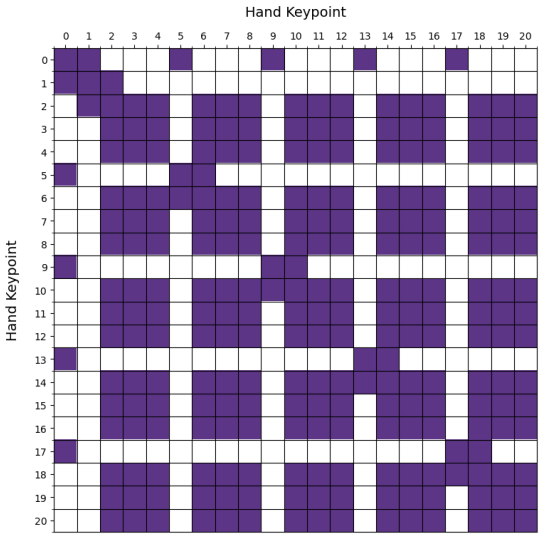


Figure 4.7: FULL adjacency matrix

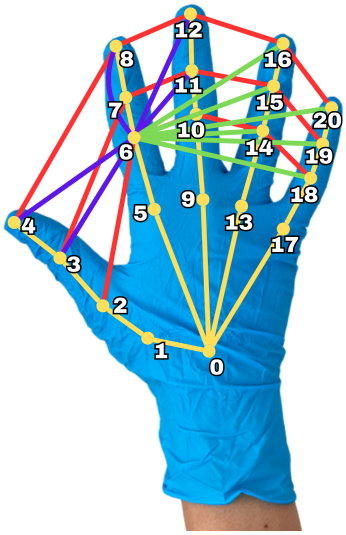


Figure 4.8: FULL Hand Connectivity for keypoint 6

The motivation behind the various hand connectivity designs presented in this thesis is rooted in the goal of enhancing information sharing between the joints, particularly across the fingers. Fingers have a wide range of motion and can occupy diverse positions, which makes them more prone to occlusions during activities, especially in scenarios involving hand-object interactions or complex movements. By increasing the connectivity between the joints in the fingers, the model can share more information between corresponding keypoints across fingers, improving its ability to infer the position of occluded joints.

Palms, on the other hand, are relatively static compared to fingers. The movement and possible positions of the palm are far more limited, as the palm serves primarily as a base for finger movement rather than being actively involved in most complex tasks. Due to this limited range of motion, even if part of the palm is occluded, its position can be more easily estimated without the need for additional connections. Furthermore, adding extra connections for palm keypoints would only increase the complexity of the graph, leading to higher computational costs without providing a significant benefit. The additional connections in the palm would not contribute substantially to resolving occlusions, since the palm's position is relatively fixed and predictable.

4.2 Multi-frame integration

4.2.1 Overview

In most hand pose estimation models, including THOR-Net, the prediction of hand pose and shape for a given frame is typically based solely on the data available in that specific frame. These models attempt to reconstruct the 3D pose without leveraging information from surrounding frames. This single-frame approach has limitations, particularly when dealing with occlusions, where parts of the hand are obstructed or not fully visible, making it difficult to accurately reconstruct the full 3D pose from limited visual information.

The multi-frame extension aims to address these limitations, especially the occlusion problem. When predicting hand pose in a single frame, the model may struggle if certain hand parts are occluded or obstructed. However, by incorporating information from preceding temporal frames, the model can leverage previously visible data to reconstruct occluded parts in the current frame. For instance, if a finger is hidden or partially occluded in the present frame, the model can refer to earlier frames where that finger was fully visible, allowing for more accurate pose estimation.

The idea of utilizing multiple frames to improve pose estimation was inspired by the H2ONet paper [13], as discussed in previous chapters. This approach, specifically highlighted in Figure 3.9, demonstrates the advantages of leveraging

temporal information for more accurate hand pose predictions. Motivated by these findings, a multi-frame strategy was incorporated into THOR-Net architecture to test its effectiveness in addressing the occlusion problem more robustly.

It is important to note that this multi-frame feature is only available in THOR-Net. OHRSA-Net does not support multi-frame integration due to limitations in its implementation. Additionally, while multi-frame approaches can significantly enhance pose estimation accuracy in various scenarios, they also introduce additional computational complexity and memory usage. A more detailed exploration of these limitations will be presented in a subsequent chapter of this work.

4.2.2 Implementation Details

The multi-frame integration feature is only available during the training phase. When enabled, the model can utilize information from preceding frames in the prediction pipeline by incorporating temporal context to improve accuracy. However, during inference, this feature is disabled because the model is designed to work at the image level, meaning there is no guarantee that a video or sequence of frames will be provided as input. In real-world scenarios, the model must be capable of accurately predicting hand pose from single, standalone images, without relying on multi-frame data.

Parameter Configuration for Multi-frame Processing

Before training, two parameters must be configured: N (number of frames) and S (stride).

Number of Frames N This parameter determines how many previous frames the model can access during training. While more frames provide a broader temporal context for making predictions, frames that are too far in the past may not be relevant to the current frame, limiting their usefulness. So, selecting an appropriate number of frames is critical to ensure the model captures meaningful temporal information without incorporating irrelevant data.

Stride S The stride controls the temporal distance between the frames used in the multi-frame integration. A larger stride means the selected frames are further apart in time, providing a broader view of the hand's movement over a longer sequence. Conversely, a smaller stride means the frames are closer together, giving the model more immediate past information, which can be useful for capturing fine-grained details in continuous movements.

For instance, if $N = 3$ and $S = 5$, and the current frame is frame 149, the model will extract features from the 3 previous frames, each spaced 5 frames apart. In this case, the model would use frames 144, 139, and 134 to provide temporal context.

Feature Extraction and Aggregation

In THOR-Net, the ResNet-50 feature extractor is used to process the images and extract relevant features from each frame. When implementing multi-frame integration, ResNet-50 is capable of receiving multiple frames as input, producing N output feature maps corresponding to the N input frames.

Once the feature maps are extracted, they are aggregated by averaging the values together. This aggregation allows the model to combine temporal information from the multiple frames into a unified representation.

In some cases, during the aggregation process, the feature map shapes from different frames may not match due to slight variations in the input data. To handle this, zero-padding is applied to the smaller feature maps to ensure that all feature maps are of the same size. Once padded, the feature maps can be successfully aggregated.

4.3 Bloodiness feature

4.3.1 Overview

The Bloodiness feature is a method introduced to help the model handle occlusions caused by the presence of blood on the glove during surgical activities. This feature provides information about the amount of blood present in specific areas of the hand, particularly around keypoints, to help the model make more accurate predictions when parts of the hand are obscured by blood.

The bloodiness feature is represented as a value between 0 and 1, where 0 indicates no blood and 1 signifies a significant amount of blood covering the keypoint. This value is passed along with the 2D keypoints during the pose to improve predictions of both the 3D pose and mesh of the hand.

The feature quantifies the amount of blood present around each keypoint by analyzing the pixel data and determining the proportion of pixels that can be classified as "red," which indicates the presence of blood.

4.3.2 Implementation Details

The steps for creating this feature for a specific predicted keypoint are outlined below:

1. **Defining a Region Around the Keypoint**: For each hand keypoint detected in the image, a square box with dimensions DIM × DIM pixels is defined, centered on the keypoint. This region is used to analyze the color information around the keypoint. In the current implementation, the box size is set to 50 × 50 pixels.

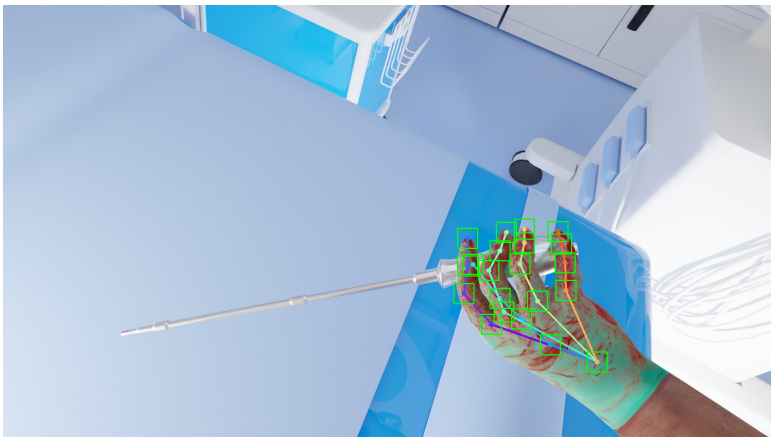


Figure 4.9: Detected bounding boxes around keypoints

2. **Red Color Detection**: Blood is typically associated with shades of red, so the next step involves determining which pixels in the region around each keypoint can be classified as red.

   The classification of a pixel as red is based on the principle that, in images, the color red is perceived when the red component in the RGB color space is higher than the green and blue components. This principle is the foundation for the implementation used to classify "red" pixels in the bloodiness feature.

   The detection of red pixels in the bloodiness feature relies on two parameters that determine how the color red is classified: Red Dominance and Red Minimum Value.

   - **Red Dominance** requires the red (R) component of a pixel's RGB value to be significantly higher than both the green (G) and blue (B) components by a specific ratio. For example, if this ratio is set to 1.5 and the values of G and B are 100 and 90 respectively, then R must be at least 150, for the pixel to be classified as red.
   - **Red Minimum Value** sets a threshold for the red component, requiring it to exceed a certain value to be classified as red. This prevents very low red component values, which might not indicate blood but rather colors closer to black, from being counted.

In the implementation of this feature, the values used are a Red Dominance of 1.15, meaning the red component must be at least 1.15 times greater than the green and blue components, and a Red Minimum Value of 100. Both criteria, dominance and minimum value, must be met for a pixel to be classified as red. These parameters have been calibrated to optimally detect complex patterns of blood colors in this dataset, allowing for robust detection across different lighting conditions and image qualities.

An example of blood detection on the hand is shown in Figure 4.11. It should be noted that irrelevant parts, such as the wrist, may also be classified as red. However, these areas are not relevant, as only the red pixels around the keypoints are used in further processing.

Figure 4.10: Original frame

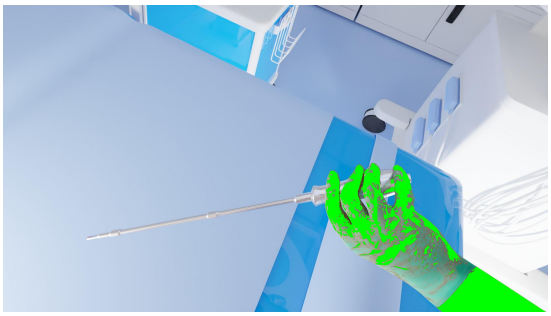


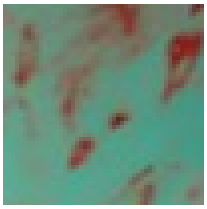
Figure 4.11: Pixels detected as blood highlighted in green

3. **Bloodiness Calculation**: Once the red pixels are identified within the dim × dim box around each keypoint, the bloodiness value is computed. This value is the ratio of red pixels to the total number of pixels in the box:

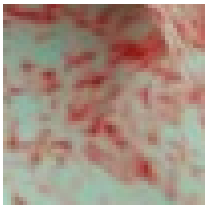
$$\text{bloodiness} = \frac{\text{Number of red pixels}}{\text{Total number of pixels in the box}}$$

   For each keypoint, this ratio gives a percentage that represents how much of the surrounding area is covered by blood. For example, if 50% of the pixels in the box are classified as red, the bloodiness value for that keypoint would be 0.5. An example of blodiness values computed for each bounding box around each keypoint are visible in figures 4.24, 4.34.

4. **Integration into the Model**: During the prediction process, the bloodiness feature is passed along with the 2D keypoints when the model is predicting the 3D pose and mesh of the hand.

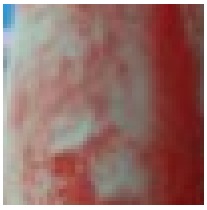


Keypoint 0: 10.80%



Keypoint 1: 51.88%

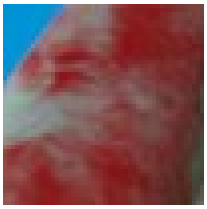
Keypoint 2: 71.92%



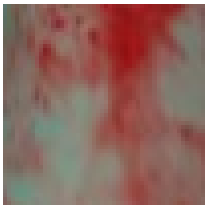
Keypoint 3: 71.48%

Keypoint 4: 89.68%

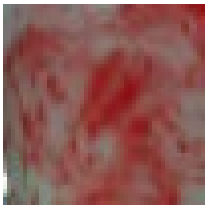
Keypoint 5: 92.12%



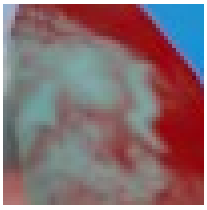
Keypoint 6: 74.80%



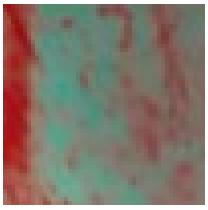
Keypoint 7: 68.84%



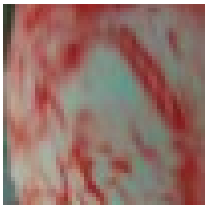
Keypoint 8: 91.16%



Keypoint 9: 47.56%



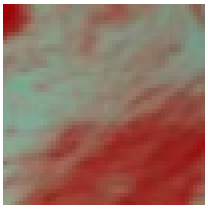
Keypoint 10: 39.72%



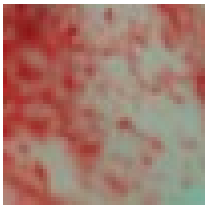
Keypoint 11: 66.88%

Bloodiness values as percentages for the first 12 keypoints in the previously shown frame.

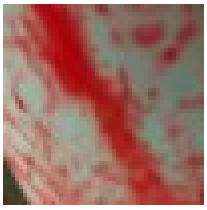
Keypoint 12: 85.76%



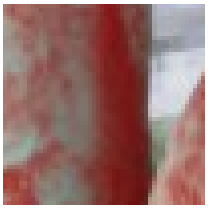
Keypoint 13: 53.68%



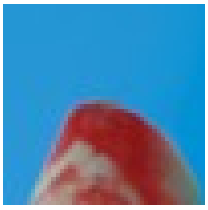
Keypoint 14: 67.68%



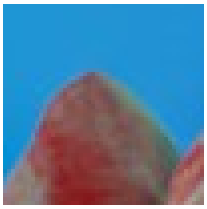
Keypoint 15: 69.12%



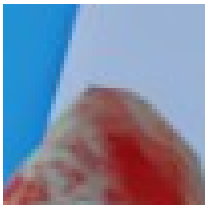
Keypoint 16: 70.76%



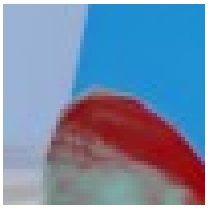
Keypoint 17: 24.92%



Keypoint 18: 33.84%



Keypoint 19: 30.60%



Keypoint 20: 22.64%

Bloodiness values as percentages for the remaining 9 keypoints in the previously shown frame.

4.4 OHRSA-Net: One Hand Reconstruction during Surgical Activities

From the THOR-Net paper, it is evident that some of its results are not the best among existing models in this domain. However, the GraFormer module, the key component of THOR-Net responsible for transforming 2D keypoints into 3D poses, has been shown to outperform other models in this specific task. This indicates that, to create a more advanced version of THOR-Net, the primary bottleneck lies in the keypoint extraction stage.

To address this, OHRSA-Net (One Hand Reconstruction during Surgical Activities) was developed with the aim of replacing the KeypointRCNN component with a more efficient and accurate keypoint detector: YOLOv8 Pose. Furthermore, OHRSA-Net is designed to incorporate the best-performing extensions tested on THOR-Net, making it a model specifically tailored for surgical activities.

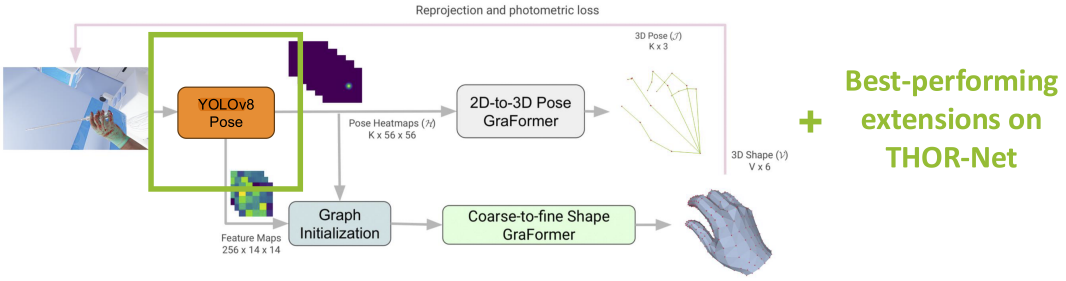


Figure 4.35: OHRSA-Net architecture

4.4.1 Transforming THOR-Net into OHRSA-Net

The decision to replace the KeypointRCNN component in OHRSA-Net with YOLOv8 Pose was driven by the continuous advancements in the YOLO model over the years, as well as an analysis of the inference speed of THOR-Net and KeypointRCNN, which will be explained further. The specific implementation from Ultralytics [14] is consistently updated with improvements, and as of now, there are more than 10 versions, each offering better performance and optimizations.

One key advantage of YOLOv8 Pose is its efficiency. There are five different scales of YOLOv8 Pose, ranging from small to extra-large. Even the extra-large version is smaller in terms of parameters compared to KeypointRCNN, making it more lightweight and easier to deploy in real-time applications, such as surgical hand pose estimation.

Designing OHRSA-Net

To replace KeypointRCNN with YOLO in the design of OHRSA-Net, several design choices were made to ensure a fair comparison with the original THOR-Net model. The first decision involved selecting the scale of YOLO to use. Given that the original THOR-Net model comprises approximately 197 million parameters and its KeypointRCNN module has around 59 million parameters, it was determined to opt for the small version of YOLO Pose (11 millions parameters) while extracting a larger number of features.

To achieve this, an important adjustment was made to the original two-layer perceptron (MLP) used to transform the extracted 2D keypoint features into input

features suitable for the GraFormer. This MLP was originally designed with an input dimension of 256 × 14 × 14, producing a 2048-feature vector as output, used as input for the GraFormer. This MLP was modified to ensure that OHRSA-Net retains a similar parameter count to THOR-Net. Given the substantial reduction in parameters from the 2D keypoint extractor, there was an opportunity to increase the size of the MLP. This adjustment aimed to capture a greater number of features than the original configuration of 256 × 14 × 14.

The final decision was to use input features with dimensions 256 × 17 × 17. The dimension of 17 × 17 was computed from this set of equations:

Given:

- $N_{MLP} = 256 \cdot 14 \cdot 14 \cdot 2048$, the original number of parameters of MLP
- $N_{MLP_{OHRSA-Net}} = 256 \cdot X^2 \cdot 2048$, the newly designed MLP
- $N_{YOLO} = 11{,}528{,}225$, the number of parameters of YOLOv8 Pose small
- $N_{keypointRCNN} = 59{,}137{,}258$, the number of parameters of KeypointRCNN

So solving the equation for X:

$$N_{MLP} + N_{KeypointRCNN} = N_{MLP_{OHRSA-Net}} + N_{YOLO}$$

$$256 \cdot 14 \cdot 14 \cdot 2048 + 59{,}137{,}258 = 256 \cdot X^2 \cdot 2048 + 11{,}528{,}225$$

$$\rightarrow X \approx 16.94 \sim 17$$

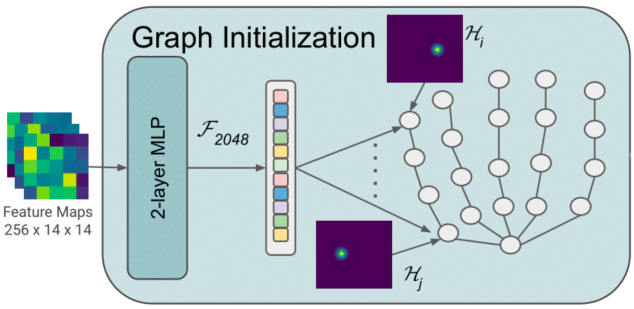


Figure 4.36: Original Two Layer MLP in THOR-Net

A Timing Analysis of KeypointRCNN and THOR-Net

One of the driving factors for replacing the KeypointRCNN component in THOR-Net was the inference time observed during two experiments focused on the keypoint extraction stage.

First Experiment In the first experiment, a direct comparison of the prediction speed between KeypointRCNN and YOLOv8 Pose was performed on a sample composed of 1000 POV Surgery frames. For both models, the average prediction time was calculated to assess their efficiency. The results clearly demonstrated a significant advantage in favor of YOLOv8 Pose. On average, YOLOv8 Pose required 61.4 ms per prediction, whereas KeypointRCNN took 140 ms per prediction, resulting in a reduction of approximately $\approx 56.14\%$. This considerable reduction in inference time highlighted the inefficiency of KeypointRCNN, especially in real-time applications like surgical hand pose estimation. These results made it evident that optimizing the keypoint extraction stage by using YOLOv8 Pose would lead to better real-time performance in OHRSA-Net.

Second Experiment The second experiment involved an analysis of the entire prediction pipeline of THOR-Net, from the extraction of 2D keypoints to the final 3D pose and mesh prediction. This experiment collected timestamps for the three main stages: 2D keypoint extraction, 3D pose prediction, and 3D mesh prediction, using the entire validation split of POV-Surgery ($\approx$ 9000 samples). Additionally, the processing times between these stages were recorded.

Below are the results of this analysis:

Stage	**Duration (ms)**
Image Preprocessing	3.3031
2D Keypoints Inference	37.5879
Pose Features Preprocessing	2.2427
3D Keypoints Inference	5.20815
Mesh Feature Preprocessing	0.1197
3D Mesh Inference	14.9182
Results Postprocessing	3.1528
Total	**66.5326**

Table 4.1: Duration of the different stages in THOR-Net's hand pose estimation process

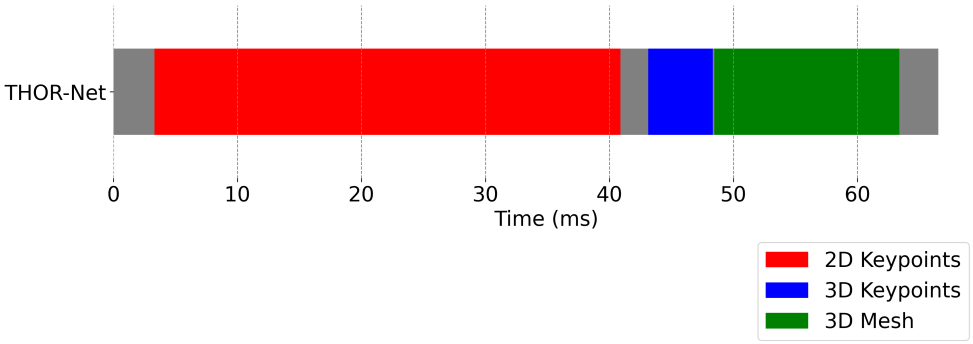


Figure 4.37: THOR-Net Timing Analysis

This experiment highlights that keypoint extraction is a bottleneck in THOR-Net's speed, as it is the longest stage among the three. As a positive outcome, this finding also led to the decision to replace the KeypointRCNN model with the faster YOLOv8 Pose.

Chapter 5

Experiments and Discussion

5.1 OHRSA-Net Timing analysis results

Using the same setup described in the previous chapter for THOR-Net, timestamps were collected for the three main stages across the full validation split of the POV-Surgery dataset.

The timing analysis results for each stage of the OHRSA-Net model, compared to THOR-Net, are presented below.

Stage	THOR-Net Duration (ms)	OHRSA-Net Duration (ms)
Image Preprocessing	3.3031	3.2795
2D Keypoints Inference	37.5879	13.5282
Pose Features Preprocessing	2.2427	0.0975
3D Keypoints Inference	5.2081	10.1792
Mesh Feature Preprocessing	0.1197	0.1743
3D Mesh Inference	14.9182	31.7364
Results Postprocessing	3.1528	0.1289
Total	**66.5326**	**58.1221**

Table 5.1: Duration of the different stages in THOR-Net and OHRSA-Net's hand pose estimation processes.

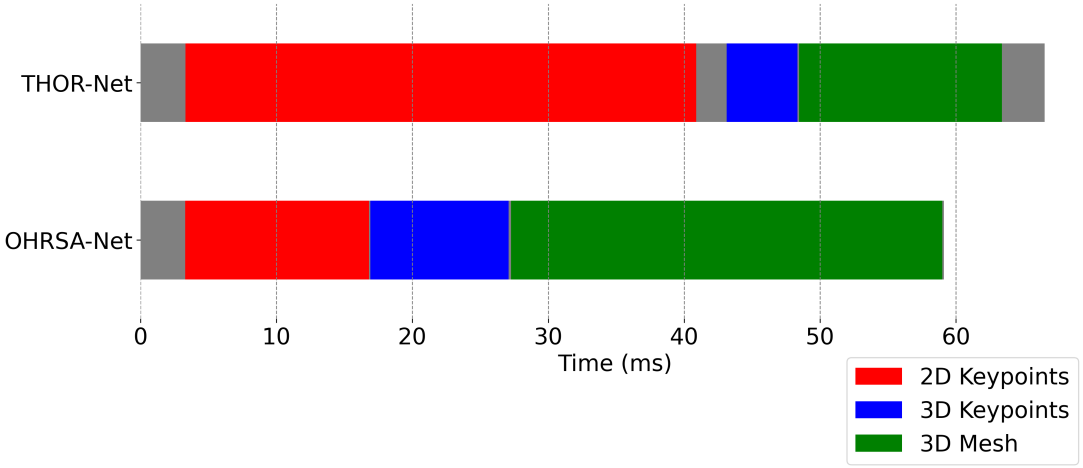


Figure 5.1: OHRSA-Net Timing Analysis Compared to THOR-Net

The most significant improvement in OHRSA-Net is observed in the 2D keypoints inference stage, where the processing time has been reduced from 37.59 ms in THOR-Net to 13.53 ms in OHRSA-Net, a reduction of $\approx 64\%$.

In terms inference time, OHRSA-Net demonstrates better efficiency, with a total processing time of 58.12 ms, compared to THOR-Net's 66.53 ms (a reduction of $\approx 12.65\%$).

Overall, OHRSA-Net demonstrates a significant reduction in the duration of every processing step, streamlining the pipeline and enhancing efficiency.

The 3D keypoints inference and 3D mesh inference stages in OHRSA-Net take longer than in THOR-Net, with OHRSA-Net requiring 10.18 ms for keypoints inference (compared to 5.21 ms in THOR-Net) and 31.74 ms for mesh inference (compared to 14.92 ms in THOR-Net). These increases are expected, as OHRSA-Net uses a richer set of features derived from the 2D keypoints prediction, which necessitates additional computations for enhanced accuracy in both 3D pose and mesh estimation. While these stages take more time, the trade-off is a more detailed and accurate output, which improves overall performance in subsequent steps.

Overall, despite the increased time in certain stages, OHRSA-Net performs faster than THOR-Net, particularly in preprocessing and postprocessing. The trade-off in increased time for mesh and pose prediction stages is outweighed by the improved accuracy and the overall faster processing time. The results suggest that OHRSA-Net provides a more efficient solution for hand pose estimation, particularly in scenarios where speed and accuracy are critical, such as in surgical applications.

5.2 Evaluation of Extensions and OHRSA-Net

5.2.1 Methodology

The experiments presented in this chapter were conducted using the test split of the dataset, while models were trained on the training split and validated during training on the validation split. The dataset, comprising 53 sequences in total, was divided into three subsets: 35 sequences (54.02%, 47,715 frames) were used for training, 5 sequences (8.98%, 7,932 frames) for validation, and 13 sequences (37.01%, 32,682 frames) for testing.

All experiments were conducted using a single NVIDIA GeForce RTX 4090 GPU with 24GB of memory, which provided the necessary computational resources to handle the high complexity of the models and the large number of frames in the dataset.

It is important to note that the reported results are not in standard units such as millimeters or pixels but are instead computed in camera space. Camera space refers to the coordinate system defined by the camera's perspective, where distances and positions are represented relative to the camera's viewpoint. This choice of representation was made after encountering significant difficulties during training with alternative unit systems. Models trained with other representations consistently failed to learn effectively. After extensive experimentation, the decision to use values in camera space proved to be the only approach that allowed the models to converge during training. A more detailed discussion of this choice and its implications is provided in the Conclusions chapter 6 .

Metrics used for Evaluation The evaluation metrics used in this study assess the 3D accuracy of hand pose and shape predictions. These metrics provide a comprehensive view of the model's performance across different aspects of hand reconstruction.

The following metrics are employed to assess the performance of the proposed models:

- **MPJPE (Mean Per Joint Position Error)**:

  MPJPE quantifies the average Euclidean distance between the predicted and ground truth 3D joint positions. It provides an overall measure of the joint-level accuracy of the model's predictions. Lower MPJPE values indicate higher precision in estimating the 3D joint positions.

- **PA-MPJPE (Procrustes-Aligned Mean Per Joint Position Error)**:

  PA-MPJPE is a variant of MPJPE where the predicted and ground truth 3D joints are aligned using Procrustes analysis. This alignment technique removes

the effects of global translation, rotation, and scale, allowing the metric to focus solely on the structural differences in pose estimation. Lower values signify better structural alignment and improved pose estimation accuracy.

- **PVE (Per Vertex Error)**:

  PVE calculates the mean Euclidean distance between the predicted and ground truth 3D vertices of the hand mesh. This metric assesses the accuracy of the reconstructed hand shape, going beyond joint positions to consider the entire mesh. Lower PVE values indicate a more accurate reconstruction of the hand geometry.

- **PA-PVE (Procrustes-Aligned Per Vertex Error)**:

  PA-PVE extends the concept of Procrustes alignment to the 3D hand mesh vertices. By aligning the predicted and ground truth meshes, PA-PVE evaluates the shape accuracy independent of position and orientation. Lower PA-PVE values indicate a better fit of the predicted mesh to the actual hand shape.

5.2.2 Hand Connectivity Evaluation

This extension has been evaluated on the THOR-Net model and compared against the baseline hand connectivity configuration of the right hand, referred to as the BASE configuration.

The results for the various metrics considered are presented below.

	MPJPE ↓	**PA-MPJPE ↓**	**PVE ↓**	**PA-PVE ↓**	**Average Error Change**
THOR-Net (BASE)	0.00009613	0.0001629	0.0004685	0.0003494	
THOR-Net + SIMPLE hand connectivity	0.00008284	0.00018233	0.00027836	0.00003487	**-24.14%**
THOR-Net + EXTENDED hand connectivity	0.00006482	0.00016931	0.00029649	0.00003472	**-25.85%**
THOR-Net + FULL hand connectivity	0.00006993	0.00015342	0.00028038	0.0003496	**-29.35%**

Table 5.2: Experimental results for THOR-Net with different hand connectivity configurations, including percentage change in average error.

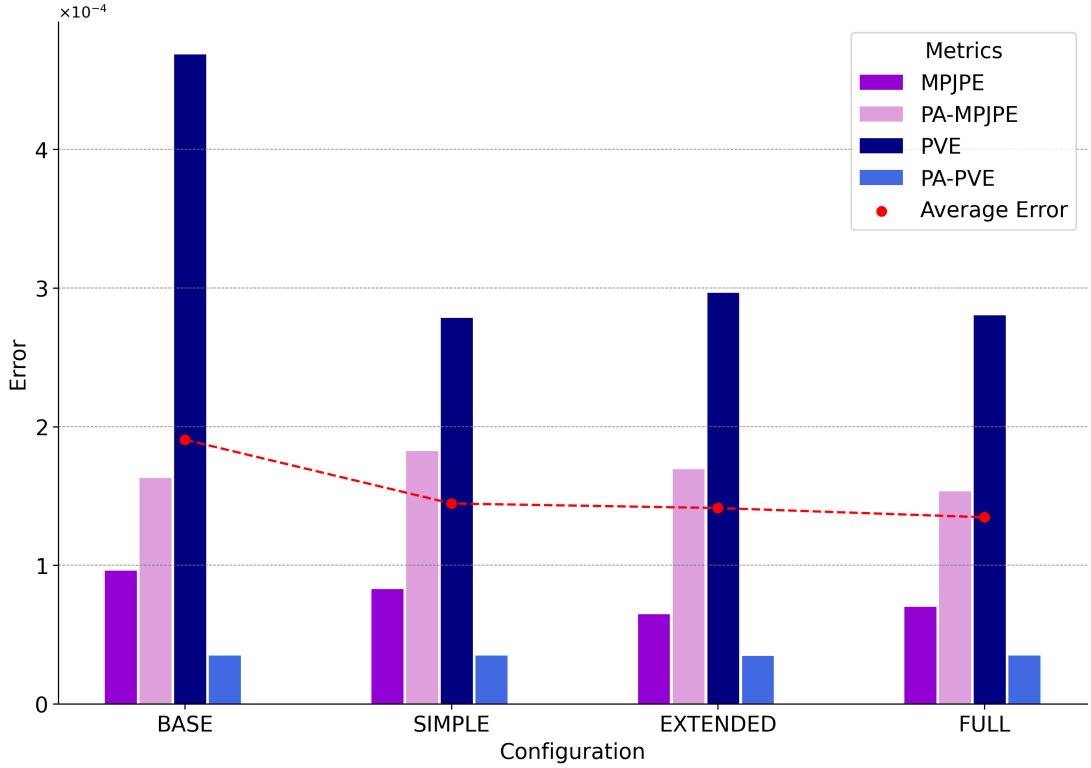


Figure 5.2: Hand Connectivity Results

Discussion The experimental configurations demonstrate a noticeable performance improvement with the introduction of custom hand connectivity strategies. Starting from the simplest configuration (BASE), we observe a significant reduction in error metrics when the SIMPLE hand connectivity scheme is applied. This suggests that even a modest increase in inter-joint connections positively influences the model's performance. Among all the metrics, PVE (Per Vertex Error) exhibits the most substantial improvement as more connections are added. This trend continues with more extensive connectivity configurations, such as EXTENDED and FULL, where the model shows further refinement in pose estimation. This is particularly evident in metrics like MPJPE and PA-MPJPE.

In further experiments on timing, despite the increased complexity of the graph structure due to more connections, the inference time remained largely unaffected. This indicates that the computational overhead introduced by additional connections is minimal and does not significantly impact performance. The absence of a noticeable slowdown can be attributed to the model's design, where the main computational constraints are imposed not by the graph connections.

In summary, the results indicate that strategically increasing the connectivity

of the hand can enhance pose and shape estimation without sacrificing efficiency, making it a valuable extension for more accurate applications.

5.2.3 Multi-Frame Integration Evaluation

	MPJPE ↓	**PA-MPJPE ↓**	**PVE ↓**	**PA-PVE ↓**	**Average Error Change**
THOR-Net (BASE)	0.00009613	0.0001629	0.0004685	0.0003494	
THOR-Net + Multi-frame $N = 2$, $S = 10$	0.00002733	0.00007274	0.00017958	0.0001726	**-61.06%**
THOR-Net + Multi-frame $N = 2$, $S = 30$	0.00003203	0.00005211	0.00021052	0.0001739	**-59.07%**

Table 5.3: Experimental Results for Multi-frame Integration with THOR-Net, including percentage change in average error.

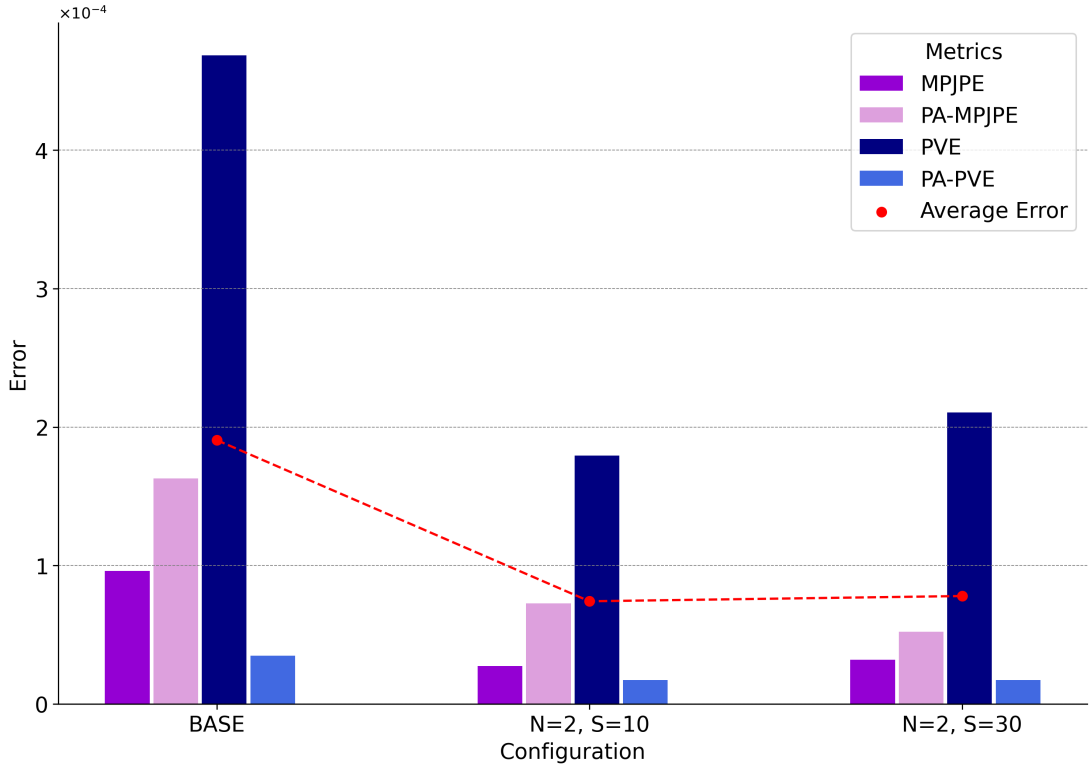


Figure 5.3: Multi-frame Integration Results

Discussion The multi-frame integration experiments on THOR-Net demonstrate a substantial enhancement in model performance, clearly surpassing the improvements achieved through the hand connectivity extensions. This significant drop is observed across all error metrics, indicating a uniform gain in both pose and shape reconstruction quality. The average error reduction of nearly 60% in both configurations underscores the effectiveness of leveraging multiple frames, suggesting a robust mitigation of occlusion issues inherent to the surgical setting. Unlike the hand connectivity adjustments, where specific metrics like PVE showed the most impact, the multi-frame approach results in a broader reduction in errors.

The improvement in accuracy can be attributed to the model's enhanced capacity to utilize temporal information from multiple frames, allowing it to infer hand poses more reliably even when parts of the hand are obstructed by surgical tools or other occluding elements. However, this approach introduces additional computational complexity. While integrating more frames improves performance, varying the stride parameter S does not appear to yield further gains, as it only adjusts the temporal sampling rate without increasing the number of frames processed concurrently.

A notable challenge faced during these experiments was the extended training time. Extracting features from multiple frames in parallel became the most computationally demanding part of the pipeline, leading to a training duration approximately 3-4 times longer than that of other experiments. Each epoch took about 1.5 to 2 hours, compared to the typical 30-minute duration in previous configurations. This increase in training time scales linearly with the number of frames due to the need for parallel feature extraction.

Additionally, hardware constraints limited the number of frames that could be processed. The increased memory and computational demands made it infeasible to train configurations with $N > 2$, as the available hardware could not accommodate the larger batch sizes required. Consequently, the experiments were restricted to $N = 2$. On the other hand, altering the stride parameter did not encounter these limitations, since it only modifies the sampling rate and does not increase the concurrent processing load.

	MPJPE ↓	PA-MPJPE ↓	PVE ↓	PA-PVE ↓	Average Error Change
THOR-Net (BASE)	0.00009613	0.0001629	0.0004685	0.0003494	
THOR-Net + Multi-frame $N = 2$, $S = 10$ + FULL hand connectivity	0.00002042	0.0000717	0.00013487	0.00001722	**-67.97%**

Table 5.4: Experimental Results for Multi-frame Integration with THOR-Net, including percentage change in average error.

5.2.4 Combining Hand Connectivity and Multi-frame Integration Extensions

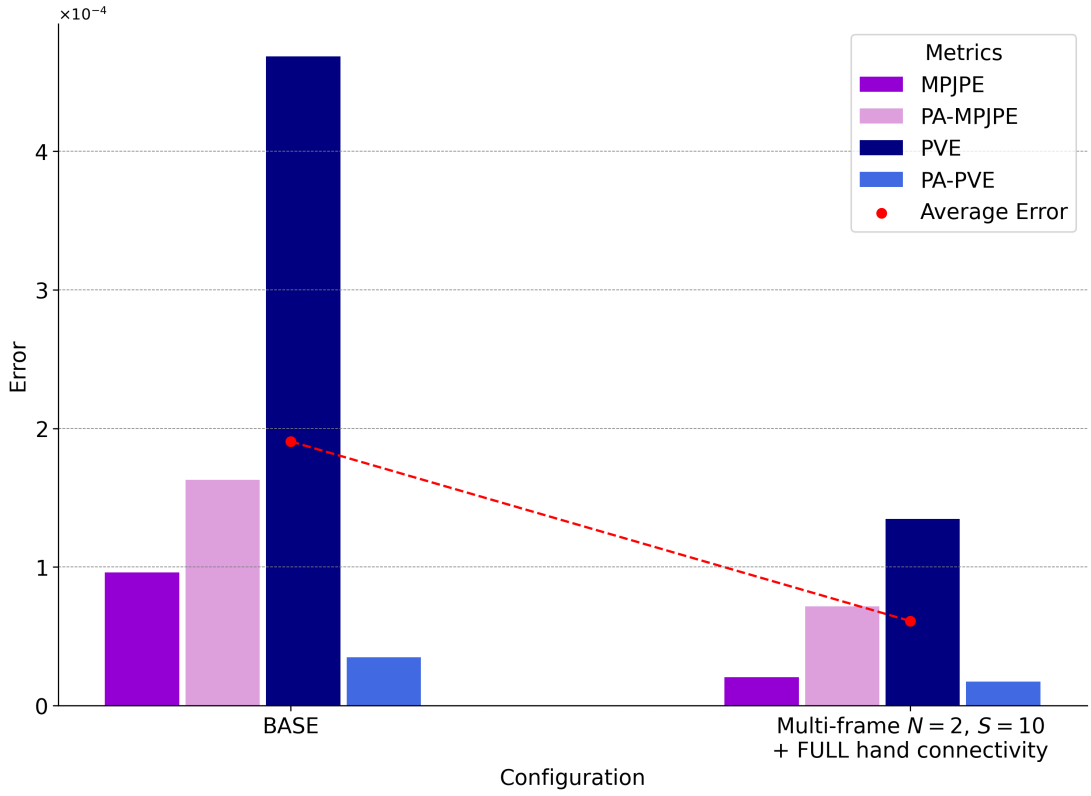


Figure 5.4: Results of Combining the Best Settings for Multi-frame Integration and Hand Connectivity Extensions

Discussion This combined experiment was conducted to evaluate whether the integration of both extensions shown previously could lead to an enhanced model performance compared to applying these extensions separately. While the goal

was primarily exploratory, rather than to form new conclusions, it serves as an informative baseline on the additive potential of these methods.

The combined extension showed a significant reduction in average error, achieving a -67.97% decrease compared to the original THOR-Net configuration. This is notably better than the results from the individual extensions, where the best-performing hand connectivity setup reduced the average error by -29.35%, and the multi-frame extension achieved a -61.06% reduction. The evidence points to an effective combination of the two strategies, significantly boosting the model's performance in hand pose estimation.

It is worth mentioning that both the limitations and strengths identified in each individual extension still apply.

5.2.5 Comparison of THOR-Net and OHRSA-Net

	MPJPE ↓	**PA-MPJPE ↓**	**PVE ↓**	**PA-PVE ↓**	**Average Error Change**
THOR-Net	0.00009613	0.0001629	0.0004685	0.0003494	
OHRSA-Net	0.00036798	0.00040626	0.00060424	0.00007018	**+90%**

Table 5.5: Experimental Results for Multi-frame Integration with THOR-Net, including percentage change in average error.

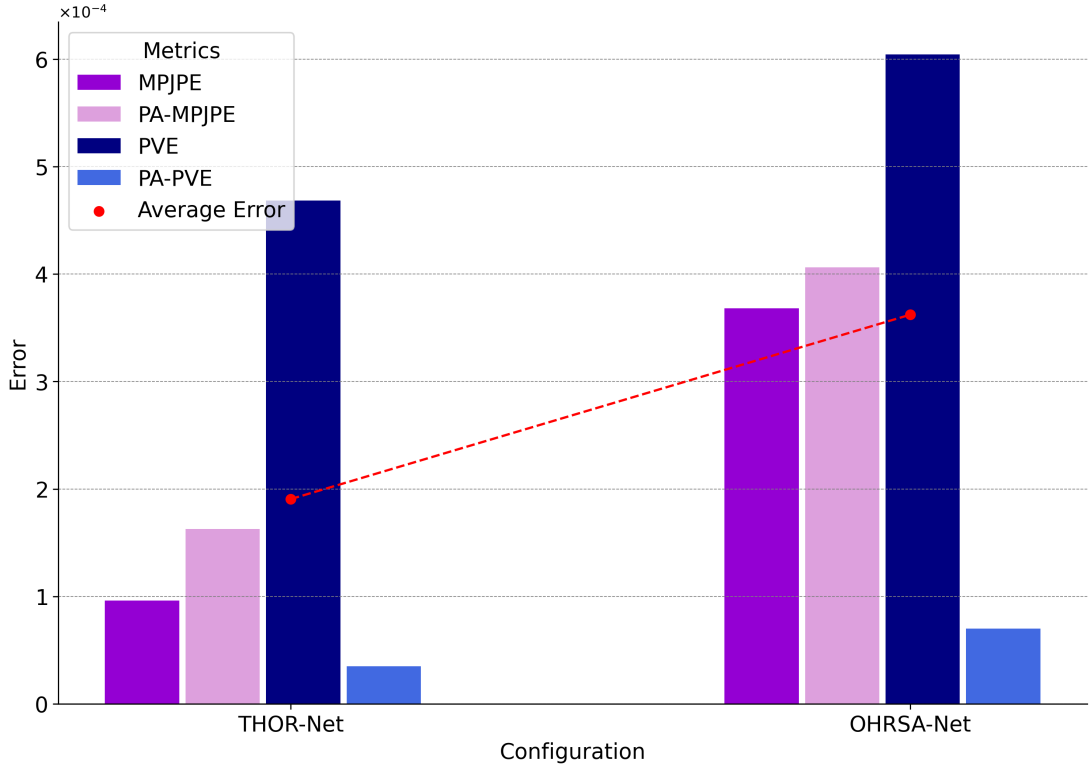


Figure 5.5: Comparison of THOR-Net and OHRSA-Net Results

Discussion The comparative analysis of THOR-Net and OHRSA-Net reveals that the performance of OHRSA-Net is consistently inferior to that of THOR-Net across all evaluation metrics. Since the primary architectural difference between these models lies in their keypoint detection module (KeypointRCNN in THOR-Net and YOLOv8 Pose in OHRSA-Net) the disparity in their performance can be directly attributed to the effectiveness of these keypoint extractors.

Despite the speed advantage, the lower accuracy of OHRSA-Net reflects the limitations of YOLO-based keypoint detection in handling intricate hand poses in the presence of occlusions. The simplification of the keypoint detection module in OHRSA-Net results in a model that is better suited for real-time applications but less capable of precise hand reconstruction.

The comparison between THOR-Net and OHRSA-Net highlights a fundamental trade-off in hand pose estimation tasks: while THOR-Net prioritizes accuracy and detailed reconstruction through its keypoint detection and feature extraction pipeline, it suffers from increased computational complexity and slower inference times. Instead, OHRSA-Net achieves faster and more efficient processing by utilizing YOLOv8 Pose, making it more practical for real-time scenarios, but at

the cost of reduced accuracy.

	MPJPE ↓	**PA-MPJPE ↓**	**PVE ↓**	**PA-PVE ↓**	**Average Error Change**
OHRSA-Net	0.00036798	0.00040626	0.00060424	0.00007018	
OHRSA-Net + Bloodiness	0.0003714	0.00037283	0.00093723	0.00007014	**+20.91%**

Table 5.6: Experimental Results of the Bloodiness Feature with OHRSA-Net

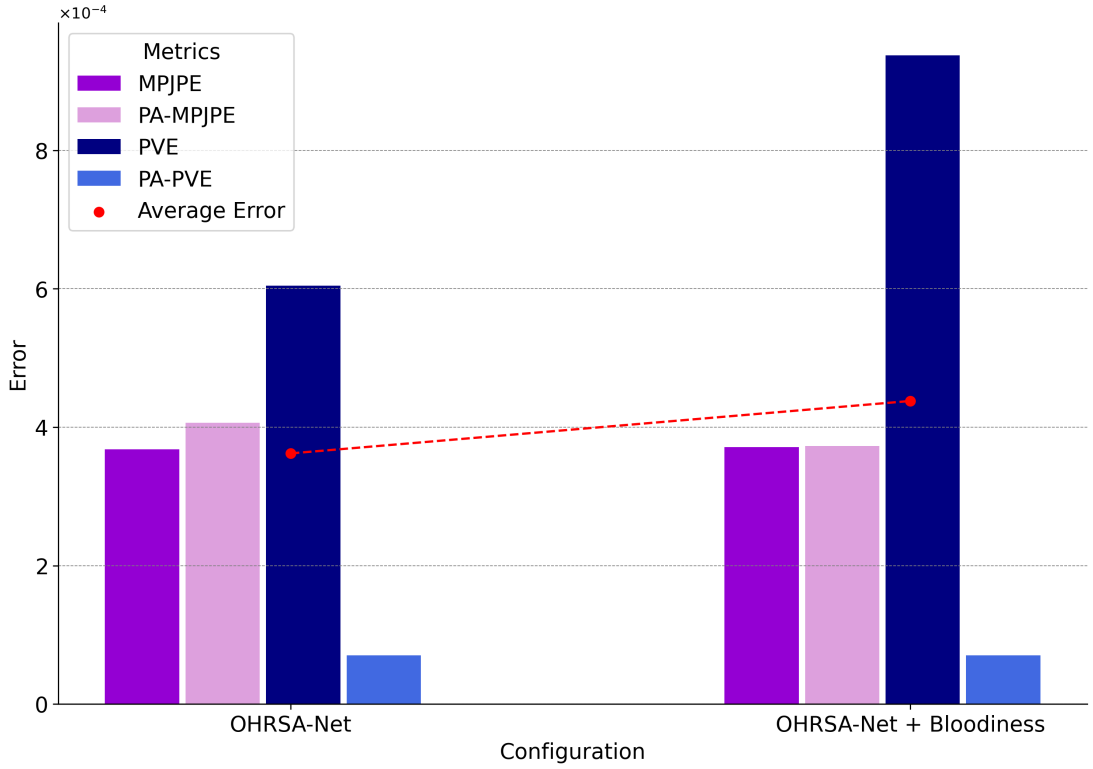


Figure 5.6: Results of the Bloodiness Feature on OHRSA-Net

Discussion The experimental results indicate that this extension did not lead to meaningful improvements. While the pose-related metrics remained unchanged, the shape-related metrics exhibited a decline in performance. These findings suggest that the inclusion of bloodiness as an additional feature for the GraFormer input did not enhance the reconstruction accuracy of the hand pose or shape.

This lack of improvement may be attributed to the way the bloodiness feature was integrated. If a keypoint is not accurately identified or is occluded, the bloodiness value associated with it might introduce noise rather than meaningful

information, potentially misleading the model during subsequent reconstruction stages. A more effective approach might have involved including bloodiness as part of the original annotations for each keypoint in the dataset. This would have allowed the model to learn to associate specific bloodiness levels with keypoint occlusions during training, providing a more robust representation. Unfortunately, the POV-Surgery dataset does not include such annotations, limiting the potential impact of this feature.

Additionally, it is important to note that OHRSA-Net underperformed compared to THOR-Net. While the two other extensions presented earlier significantly enhanced THOR-Net's performance, applying these extensions to OHRSA-Net was unlikely to yield better results than those achieved with the extended THOR-Net. The weaker baseline performance of OHRSA-Net made it improbable that integrating these successful extensions would surpass the improvements already realized with THOR-Net. This also justified the decision not to implement the bloodiness feature within THOR-Net, as the limitations observed in OHRSA-Net would likely persist.

It is also worth highlighting that the bloodiness experiments were conducted exclusively on OHRSA-Net, as it was the initial model used for implementing this feature. The overall poorer performance of OHRSA-Net, combined with the lack of meaningful improvements from the bloodiness experiments, underscores the need for future research to explore alternative approaches.

5.2.6 Base Models Training Using Pre-trained Checkpoints

One promising approach to improve the accuracy of THOR-Net and OHRSA-Net involved fine-tuning these models using a pre-trained THOR-Net checkpoint from the HO-3D dataset [15]. Instead of training the models from scratch, this strategy leverages a previously trained model that may have already developed a generalized understanding of hand representation. If the pre-trained checkpoint is sufficiently robust and capable of generalization, it could accelerate convergence during training and enhance the models' performance on specific tasks.

HO-3D Dataset The HO-3D dataset, short for Hand Object 3D, is a widely used dataset designed for 3D hand-object interaction understanding. It contains a large collection of images depicting hands interacting with various objects in diverse real-world settings. The dataset includes RGB images, depth maps, and 3D annotations for both hands and objects, providing a resource for tasks like 3D hand pose estimation and hand-object interaction analysis. The annotated hand poses span various activities, offering a rich representation of hand configurations under occlusion and interaction with objects.

For these experiments, version 2 of the HO-3D dataset (HO-3D v2) was utilized,

which includes updated and improved annotations, enhancing the quality of the training data.

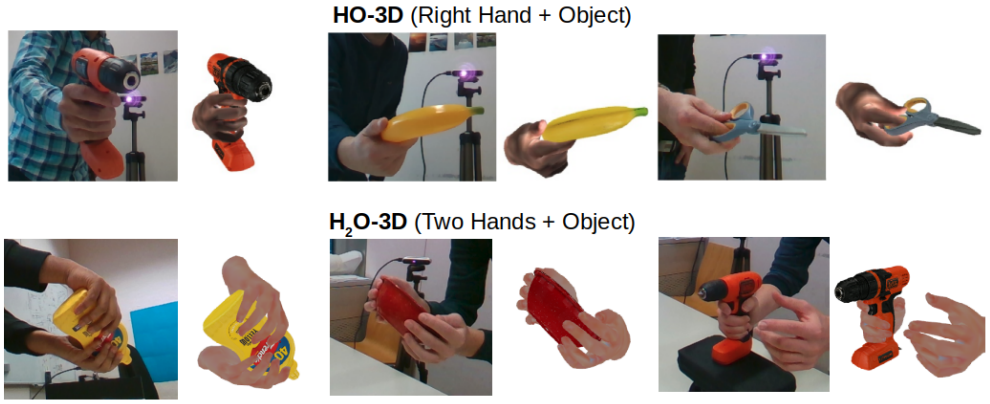


Figure 5.7: HO-3D Dataset

	MPJPE ↓	PA-MPJPE ↓	PVE ↓	PA-PVE ↓	Average Error Change
THOR-Net	0.00009613	0.0001629	0.0004685	0.0003494	
THOR-Net pre-trained on HO-3D dataset	0.00056731	0.00018166	0.00043800	0.00003268	**+59.96%**

Table 5.7: Experimental Results for THOR-Net with HO-3D Pre-training.

	MPJPE ↓	PA-MPJPE ↓	PVE ↓	PA-PVE ↓	Average Error Change
OHRSA-Net	0.00036798	0.00040626	0.00060424	0.00007018	
OHRSA-Net pre-trained on HO-3D dataset	0.00484654	0.00040093	0.00110541	0.00006474	**+343%**

Table 5.8: Experimental Results for OHRSA-Net with HO-3D Pre-training.

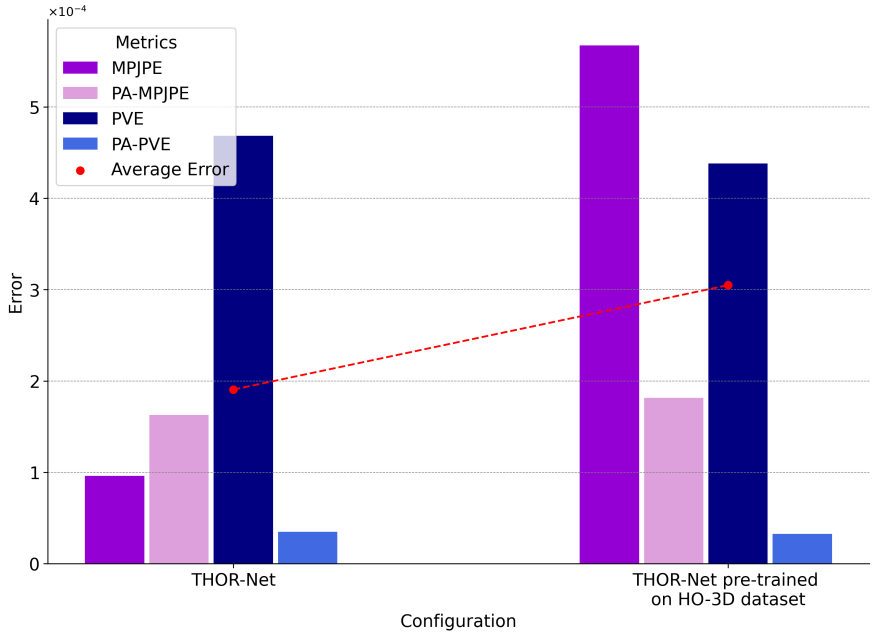


Figure 5.8: Results of Pre-trained THOR-Net

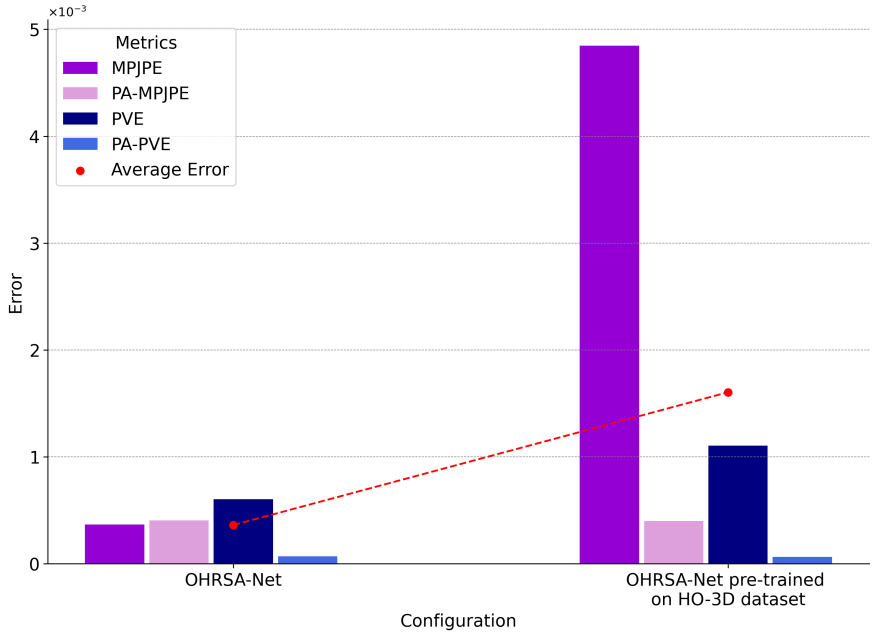


Figure 5.9: Results of Pre-trained OHRSA-Net

Discussion The experiments using a pre-trained model revealed clear limitations in how well the learned representations could generalize. Specifically, the pre-trained model struggled to perform well outside the domain it was originally trained on. As a result, the expected improvements from using a pre-trained model did not materialize.

For OHRSA-Net, the challenges were even greater. Due to architectural differences between OHRSA-Net and THOR-Net, only the parameters of the GraFormer module could be transferred from the HO-3D pre-trained checkpoint. Since the GraFormer is the only part of the architecture that remained unchanged, the rest of OHRSA-Net had to be trained from scratch. This led to much worse performance compared to the original configuration of the model.

Ultimately, fine-tuning with the HO-3D pre-trained model did not lead to any improvements in accuracy for either THOR-Net or OHRSA-Net. The lack of meaningful performance gains made further experiments with this setup unnecessary, as it became clear that this approach was not effective for improving the models.

Chapter 6

Conclusions and Future Works

The goal of this thesis was to explore the domain of hand pose estimation, with a specific focus on its application during surgical tool usage. Through the development and evaluation of models such as THOR-Net and OHRSA-Net, the study investigated the efficacy of various architectural and methodological extensions. While the findings did not culminate in a breakthrough discovery, they provided critical insights into the strengths and limitations of combining Graph Convolutional Neural Networks and Transformers for this application.

The comparative analysis revealed that THOR-Net remains the superior model in terms of overall performance. Several enhancements were successfully integrated into THOR-Net. Among these, the multi-frame integration and hand connectivity extensions demonstrated substantial promise. They effectively mitigated challenges like occlusions caused by surgical tools or the surgeon's body, contributing to more accurate pose estimation. However, the bloodiness level extension did not yield any significant improvements, suggesting that additional strategies may be required to address visual obstructions during surgery.

OHRSA-Net, while not outperforming THOR-Net, offered meaningful advancements in certain areas. By replacing KeypointRCNN with YOLO v8 Pose for keypoint detection, the model achieved remarkable speed improvements, significantly reducing inference time for 2D keypoints prediction. However, the inability of OHRSA-Net to surpass THOR-Net's overall accuracy suggests that these speed gains come with trade-offs in predictive precision.

The thesis also highlighted broader challenges within this field. The POV-Surgery dataset introduced significant difficulties due to its diverse and intricate surgical scenarios, which greatly complicated the model's learning process. This challenge was compounded by the annotations, where the original dataset's large

numerical values caused the loss function to stagnate. Only after transforming and scaling these values down to a smaller range did the models begin to converge during training. Additionally, hardware limitations constrained the number of experiments, as model training was highly time-intensive, with some configurations requiring up to five days for adequate convergence. These issues highlight the pressing need for more efficient training pipelines and better computational resources for future research.

Looking forward, there are several directions that future work could take to build upon the foundations laid by this thesis. One promising avenue is the integration of tool pose estimation into the current framework. The POV-Surgery dataset lacks object keypoint annotations, limiting its application to hand-only tasks. Adding such annotations could enable models to predict both hand and tool poses simultaneously, offering a more comprehensive understanding of surgical activities. Similarly, advancing the multi-frame integration capabilities of OHRSA-Net would be invaluable. Due to limitations in Ultralytics' implementation of YOLO v8 Pose, it was not feasible to process multiple frames concurrently. If an open-source implementation of YOLO v8 Pose becomes available, this limitation could be addressed, allowing for seamless feature extraction across multiple frames.

Another critical direction involves rethinking the use of the MANO model for hand representation. Instead of directly predicting keypoints and vertices, future models could leverage MANO's parametric design to estimate hand pose and shape using a reduced set of parameters. This approach would simplify the learning process and potentially improve accuracy. Moreover, exploring open-source alternatives to YOLO v8 Pose that can be fully integrated into larger PyTorch-based architectures would eliminate the need for a two-stage training process, streamlining the development pipeline.

Bibliography

[1] Ahmed Tawfik Aboukhadra, Jameel Malik, Ahmed Elhayek, Nadia Robertini, and Didier Stricker. *THOR-Net: End-to-end Graformer-based Realistic Two Hands and Object Reconstruction with Self-supervision*. 2022. arXiv: 2210.13853 [cs.CV]. URL: https://arxiv.org/abs/2210.13853 (cit. on pp. 3, 18).

[2] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. «Faster r-cnn: Towards real-time object detection with region proposal networks». In: *Advances in neural information processing systems* 28 (2015) (cit. on p. 5).

[3] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. «Rich feature hierarchies for accurate object detection and semantic segmentation». In: *Computer Vision and Pattern Recognition*. 2014 (cit. on p. 5).

[4] Ross Girshick. «Fast R-CNN». In: *International Conference on Computer Vision (ICCV)*. 2015 (cit. on p. 5).

[5] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. «You Only Look Once: Unified, Real-Time Object Detection». In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 779–788. DOI: 10.1109/CVPR.2016.91 (cit. on p. 6).

[6] Saidul Islam, Hanae Elmekki, Ahmed Elsebai, Jamal Bentahar, Najat Drawel, Gaith Rjoub, and Witold Pedrycz. *A Comprehensive Survey on Applications of Transformers for Deep Learning Tasks*. 2023. arXiv: 2306.07303 [cs.LG]. URL: https://arxiv.org/abs/2306.07303 (cit. on p. 11).

[7] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. *Do Transformers Really Perform Bad for Graph Representation?* 2021. arXiv: 2106.05234 [cs.LG]. URL: https://arxiv.org/abs/2106.05234 (cit. on p. 15).

[8] Kevin Lin, Lijuan Wang, and Zicheng Liu. «Mesh Graphormer». In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2021, pp. 12939–12948 (cit. on p. 15).

[9] Weixi Zhao, Yunjie Tian, Qixiang Ye, Jianbin Jiao, and Weiqiang Wang. *GraFormer: Graph Convolution Transformer for 3D Pose Estimation*. 2021. arXiv: 2109.08364 [cs.CV]. URL: https://arxiv.org/abs/2109.08364 (cit. on p. 15).

[10] Rui Wang, Sophokles Ktistakis, Siwei Zhang, Mirko Meboldt, and Quentin Lohmeyer. «POV-Surgery: A Dataset for Egocentric Hand and Tool Pose Estimation During Surgical Activities». In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. 2023, pp. 440–450 (cit. on pp. 21, 24).

[11] Shaowei Liu, Hanwen Jiang, Jiarui Xu, Sifei Liu, and Xiaolong Wang. «Semi-Supervised 3D Hand-Object Poses Estimation with Interactions in Time». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2021 (cit. on p. 24).

[12] JoonKyu Park, Yeonguk Oh, Gyeongsik Moon, Hongsuk Choi, and Kyoung Mu Lee. «HandOccNet: Occlusion-Robust 3D Hand Mesh Estimation Network». In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022 (cit. on pp. 24, 28).

[13] Hao Xu, Tianyu Wang, Xiao Tang, and Chi-Wing Fu. «H2ONet: Hand-Occlusion-and-Orientation-Aware Network for Real-Time 3D Hand Mesh Reconstruction». In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2023, pp. 17048–17058 (cit. on pp. 24, 33, 40).

[14] Ultralytics. *Ultralytics Pose Documentation*. Accessed: 2024-10-19. 2024. URL: https://docs.ultralytics.com/tasks/pose/ (cit. on p. 47).

[15] Shreyas Hampali, Mahdi Rad, Markus Oberweger, and Vincent Lepetit. «HOnnotate: A Method for 3D Annotation of Hand and Object Poses». In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020 (cit. on p. 62).