國立臺灣大學電機資訊學院電子工程學研究所

碩士論文

Graduate Institute of Electronics Engineering
College of Electrical Engineering and Computer Science
National Taiwan University

Master's Thesis

基於注意力引導特徵與領域自適應學習之實時雙目自我視角三維人體姿態估計系統

Real-Time Stereo Egocentric 3D Human Pose Estimation System with Attention-Guided Features and Domain-Adaptive Learning

范詠為

Yung-Wei Fan

指導教授：簡韶逸 博士

Advisor: Shao-Yi Chien, Ph.D.

中華民國 114 年 09 月

September 2025

doi:10.6342/NTU202504485

doi:10.6342/NTU202504485

Real-Time Stereo Egocentric 3D Human Pose Estimation System with Attention-Guided Features and Domain-Adaptive Learning

Yung-Wei Fan

Advisor: Shao-Yi Chien

Graduate Institute of Electronics Engineering

National Taiwan University

Taipei, Taiwan

September 2025

doi:10.6342/NTU202504485

doi:10.6342/NTU202504485

國立臺灣大學碩士學位論文
口試委員會審定書
MASTER'S THESIS ACCEPTANCE CERTIFICATE
NATIONAL TAIWAN UNIVERSITY

基於注意力引導特徵與領域自適應學習之實時雙目自我視角三維人體姿態估計系統

Real-Time Stereo Egocentric 3D Human Pose Estimation System with Attention-Guided Features and Domain-Adaptive Learning

本論文係范詠為（R11943022）在國立臺灣大學電子工程學研究所完成之碩士學位論文，於民國 114 年 2 月 4 日承下列考試委員審查通過及口試及格，特此證明。

The undersigned, appointed by the Graduate Institute of Electronics Engineering on 4 February 2025 have examined a Master's Thesis entitled above presented by Yung-Wei Fan (R11943022) candidate and hereby certify that it is worthy of acceptance.

口試委員 Oral examination committee:

簡韶逸　　施吉昇　　陳宗文

（指導教授 Advisor）

林洲彬

系主任/所長 Director: 江介宏

Master's Thesis Acceptance Certificate

doi:10.6342/NTU202504485

致謝

「走啊，吃飯！回來再想啦。」

耳邊那句熟悉的呼喚，彷彿還迴盪在實驗室的空氣裡。轉眼間，三年的碩班旅程就劃下句點。從德國交換回來，再次踏進實驗室，歡笑聲仍在耳畔縈繞，熟悉的身影歷歷在目。我一邊收拾桌面零散的物件，也一邊輕輕收整著心裡的思緒。這段研究的旅程，滿懷滿足，也充滿感恩。

感謝簡韶逸老師，您總是提醒我們要善用學生身分去做嘗試。在每一次的咪挺和個咪中，耐心給予方向與建議，讓我能在一條安心的道路上自由探索。這條路上難免遭遇挫折，是您溫暖而肯定的話語，驅散了我的自我懷疑，幫助我在岔路口做出正確選擇。也許貢獻有限，但此時此刻的我能大聲地說：我真心喜歡我所做的研究！

這一路上，要感謝的人實在太多，容我無法一一細數。感謝華揚，在我還是專題生時，帶領我進入電腦視覺的世界。謝謝柏憲、立洋、富昇、秉誼、凱翔、昱愷、家甫、名志、哲欽、紹軒、欣妤、子晴、愷縴、奕勳、昱仁、可瀚、譽耀，學長姊們在報告時展現的自信與風采，一直是我崇敬的榜樣。感謝潘軒、嘉偉，你們研究上穩健的步伐，是我追趕的目標與動力。謝謝郁楷，口試前那段埋頭苦幹的日子，我們一邊抱怨不順，一邊互相幫助、鼓勵。而無數次並肩仰望博理館外的夜空，必將成為難忘的回憶。謝謝子軒，即使你這幾年致力於在火舞上登峰造極，我們仍能延續過去的默契，在數輔桌上天馬行空地激盪研究方向，隨性卻總帶給我啟發。謝謝郁瑄、辰淯，你們積極的態度，也常常提醒我要持續向前。能有你們同行，是我的幸運。謝謝冠廷、祐綸、宇倫、孟平、瑭毅、子青、奕節、彥妤、翔誠，有你們在的 421，總是充滿歡樂氣氛。逸平、宇清、玠志、昱翔、根齊、芷寧、亭尹，雖然相處時間不長，祝你們都能找到屬於自己的方向。

另外我還要感謝前女友聽聽兩年來的陪伴，妳總是認真地聽著我傾訴研究上的阻礙。妳說做研究就像一場馬拉松，偶爾停下來喝口水也不太會影響到最終結果！

doi:10.6342/NTU202504485

只要堅持下去，一定有衝過終點線的一天。這些話給了我極大的力量，幫助我度過許多低潮。儘管如今我們不在彼此身邊，依然要帶著彼此的祝福前進。我也要感謝我的家人。你們是我最堅實的後盾，讓我在結束疲憊的一天後能有個溫暖的歸宿。最後，我要謝謝過去的自己。雖然追求盡善盡美常讓你背負壓力，但也因為這份堅持，你才走到了今天。辛苦了，現在的我，為你感到驕傲。

我像往常一樣，輕輕關上實驗室的大門，回首那條空蕩的走廊。但這一次不同了——焦慮、不安與挫折感已不復存在，取而代之的，是一份篤定與期待。因為我知道，我們都將迎向嶄新的篇章。各位，有緣，再相見！

2025.09.15 范詠為

doi:10.6342/NTU202504485

中文摘要

自我視角三維人體姿態估計能夠在虛擬實境（VR）和擴增實境（AR）應用中提供自然且沉浸式的互動，無需依賴外部攝影機或手持控制器。這種方法為全身動作捕捉提供了一種靈活且便攜的解決方案，可無縫整合至頭戴式顯示裝置（HMD）與 AR 眼鏡。然而，現有方法面臨重大挑戰，包括魚眼鏡頭的變形、嚴重的自遮擋，以及視野範圍外的身體部位，這些問題都會降低姿態估計的準確度。此外，許多基於深度學習的方法需要高計算資源與複雜的模型架構設計，使其難以在邊緣設備上實時部署。

在本論文中，我們提出了一個適用於邊緣設備的實時雙目自我視角三維人體姿態估計系統。我們的系統引入了一個注意力引導特徵提取器（AFE），利用多尺度二維熱圖與人體遮罩注意力機制來增強特徵學習。此外，我們開發了一種立體關節混合器（SJM），這是一種簡單的多層感知機（MLP）模型，可整合雙目視覺特徵，同時保持計算效率與準確性。為了提升在真實環境中的適應性，我們引入無監督領域自適應（UDA），包括人體先驗約束與對抗性領域分類器訓練，以減少合成數據與真實世界數據之間的領域差距。

我們在 Xilinx Zynq UltraScale+ MPSoC ZCU104 FPGA 上部署了該系統，並在數據集推理與攝影機串流模式下實現了 24–30 FPS 的實時推理。此技術為可擴展的實時自我視角姿態估計奠定了基礎，並能夠提升 VR/AR 互動、人機介面（HCI）以及運動分析應用的效能。

doi:10.6342/NTU202504485

doi:10.6342/NTU202504485

Abstract

Egocentric 3D human pose estimation enables natural and immersive interaction in virtual reality (VR) and augmented reality (AR) applications, eliminating the need for external cameras or handheld controllers. This approach provides a compact and mobile solution for full-body motion capture, allowing seamless integration with head-mounted display (HMD) and AR glasses. However, existing methods face significant challenges, including fisheye lens distortion, severe self-occlusion, and out-of-view body parts, which degrade estimation accuracy. Furthermore, many deep learning-based approaches require high computational resources and complex model architecture design, making real-time deployment on edge devices impractical.

In this thesis, we propose a real-time stereo egocentric 3D human pose estimation system optimized for edge-device deployment. Our system introduces an Attention-Guided Feature Extractor (AFE) that utilizes multi-scale 2D heatmaps and human mask attention to enhance feature learning. Additionally, we develop a Stereo Joint Mixer (SJM), a simple MLP-based model that integrates stereo visual features while preserving computational efficiency and accuracy. To improve robustness in real-world environments, we incorporate unsupervised domain adaptation (UDA), including human prior constraints and adversarial domain classifier training, reducing the domain gap between synthetic and real-world data.

We implement the system on the Xilinx Zynq UltraScale+ MPSoC ZCU104 FPGA, achieving real-time inference at 24-30 FPS in both dataset inference and camera streaming modes. This technology paves the way for scalable, real-time ego-

doi:10.6342/NTU202504485

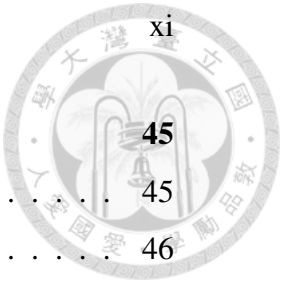
centric pose estimation, enabling enhanced VR/AR interaction, human-computer interfaces (HCI), and motion analysis applications.

doi:10.6342/NTU202504485

Contents

doi:10.6342/NTU202504485

doi:10.6342/NTU202504485



doi:10.6342/NTU202504485

doi:10.6342/NTU202504485

List of Figures

doi:10.6342/NTU202504485

doi:10.6342/NTU202504485

List of Tables

doi:10.6342/NTU202504485

doi:10.6342/NTU202504485

Chapter 1

Introduction

With the rapid advancements in Virtual Reality (VR) and Augmented Reality (AR) technologies, human-computer interaction has entered a new era where physical movements play a pivotal role. This evolution is largely supported by head-mounted display (HMD), which allow users to experience immersive environments and natural interactions. These capabilities have enabled a wide range of applications, including entertainment, healthcare, fitness, industrial training, and remote-controlled robotics. As these applications grow in complexity and interactivity, the need for accurate and efficient human pose estimation methods becomes increasingly critical to ensure seamless integration with interactive systems.

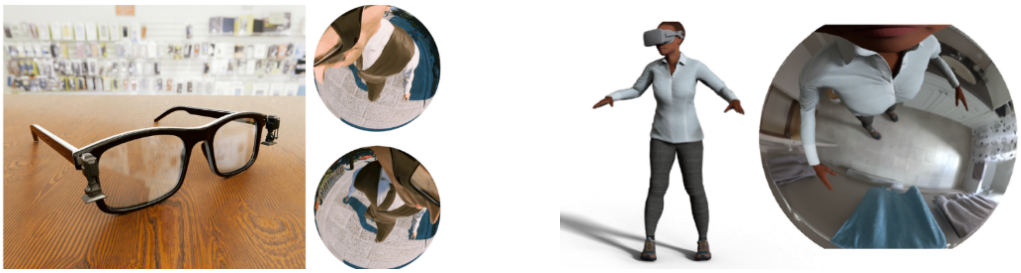
Traditionally, these systems use head-mounted devices and handheld controllers to track movements and infer the posture of other body parts through inverse kinematics. However, these methods are often imprecise and fail to enable true hands-free interaction. Recently, deep learning-based computer vision has shown promising advancements, making it possible to predict human poses directly from visual inputs. This shift has positioned vision-based human pose estimation as a compelling area of research.

doi:10.6342/NTU202504485

1.1 Egocentric 3D Human Pose Estimation

Human pose estimation has been extensively studied in computer vision, where the goal is to predict the 3D coordinates of key body joints based on visual inputs. In general, this is achieved using external cameras placed at fixed positions to capture subjects from multiple angles in a third-person view. Neural networks are then applied to predict 3D human poses based on these visual inputs.

To integrate human pose estimation with HMD, a recent mainstream approach involves mounting downward-facing fisheye cameras on the device [1, 2] (see Figure 1.1). The wide field of view provided by fisheye lenses maximizes body coverage, enabling the capture of as much of the user's body as possible for pose estimation. This approach eliminates the need for external cameras and adapts well to diverse environments. Egocentric views also reduce occlusions from external objects, such as crowded scenes and furniture, by moving with the user to maintain visibility of key body parts.



(a) Setup on eyeglasses [1]. (b) Integration with an HMD [2].

Figure 1.1: **Illustrations of downward-facing fisheye camera setups [1, 2].**

1.2 Challenges

While egocentric 3D human pose estimation offers significant advantages, it cannot directly adopt methods developed for traditional third-person views due to substantial differences in camera viewpoints. These differences introduce new challenges

doi:10.6342/NTU202504485

that must be addressed to achieve accurate and robust performance.

A primary challenge arises from the distortion caused by fisheye lenses. While fisheye cameras provide a wide field of view, their non-linear distortions complicate pose estimation. Another major challenge is self-occlusion, where parts of the body block others from view. For example, the torso may obscure the lower body, making it difficult to track leg movements accurately, especially during dynamic actions such as squatting or bending. A third challenge involves out-of-view body parts. Since the camera is head-mounted and faces downward, body parts such as raised arms can extend beyond its field of vision, leading to incomplete data and errors.

Beyond viewpoint-related issues, building an applicable system presents additional challenges, particularly in terms of real-time processing and robustness. Real-time processing is essential for smooth and interactive applications, especially in virtual environments where delays can disrupt user experience and immersion. However, most existing models are too large [1, 14, 15] (see Table 1.1), tested only on GPUs, and rely on complex designs [15, 7, 4], which make them unsuitable for deployment on edge devices, such as HMD or AR glasses, due to their limited computational resources. Robustness is equally critical but constrained by the difficulty of obtaining 3D pose ground truth labels, which often require costly motion capture systems like VICON [16]. Consequently, most datasets are synthetic [1, 2], while real-world datasets are either unlabeled [17] or collected in controlled lab environments and limited in size [3, 7]. This domain gap—caused by differences in environment, background, and lighting—frequently leads to model failures and implausible pose predictions, posing challenges for practical deployment.

1.3 Contribution

In this work, we address the critical challenges of egocentric 3D human pose estimation by developing a real-time and robust system optimized for deployment

doi:10.6342/NTU202504485

Table 1.1: **Comparison of model size and computational operations among existing methods.**

Existing Method	Params	FLOPs
Ego3DPose [15]	178.4M	55.6G
EgoGlass [14]	107.3M	16.1G
UnrealEgo [1]	106.8M	27.1G
EgoPoseFormer [4]	14.1M	7.3G

on edge devices. Our contributions span model architecture design, learning strategies, and deployment considerations.

First, we present a lightweight yet effective model architecture based on a baseline pose estimator. To improve feature extraction and enhance the model's ability to capture joint information while differentiating human regions from background noise, we incorporate multi-scale 2D heatmaps and human mask-guided attention. Furthermore, to maintain compatibility with resource-constrained hardware and to improve accuracy, we propose the Stereo Joint Mixer—an MLP-Mixer-inspired structure [5]. This design efficiently processes stereo features spatial and channel-wise using simple multi-layer perceptrons (MLPs), overcoming the hardware deployment limitations of transformer-based models while maintaining high accuracy and practicality.

Second, we introduce a domain-adaptive training framework to bridge the domain gap between synthetic and real-world datasets. This framework improves robustness and generalization by addressing environmental variations, such as lighting and background differences commonly encountered in practical applications, as well as human skeleton constraints to ensure plausible and consistent pose predictions. Together, these methods advance the feasibility of egocentric pose estimation systems and provide a foundation for further exploration in real-world scenarios.

Our contributions are summarized as follows:

doi:10.6342/NTU202504485

- We propose a simple yet effective model architecture that enhances feature extraction using multi-scale 2D heatmaps and human mask-guided attention. Additionally, we introduce the Stereo Joint Mixer, inspired by MLP-Mixer [5], to efficiently process stereo features with simple MLP layers, balancing computational efficiency and accuracy.

- We introduce a domain-adaptive training framework to mitigate the domain gap between synthetic and real-world datasets, improving robustness and generalization across diverse environments.

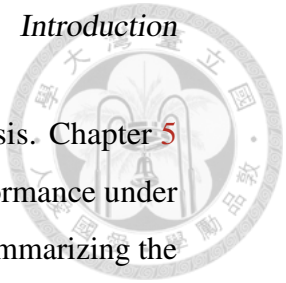
- We develop a camera-streaming system implemented on the ZCU104 FPGA board, enabling egocentric 3D human pose estimation in real-time on resource-constrained devices with efficient deployment flow.

Our proposed model architecture outperforms existing methods on the UnrealEgo dataset [1], achieving competitive results with only a slight margin behind state-of-the-art transformer-based approaches. Furthermore, our camera-streaming system implemented on the ZCU104 FPGA achieves 24 to 30 FPS, enabling real-time visualization of the user's current pose when wearing the device. This demonstrates the system's potential scalability for VR and AR applications.

1.4 Thesis Organization

In this chapter, we introduce the background and motivation for egocentric 3D human pose estimation, highlighting its importance in enabling immersive and interactive applications. We also discuss the technical challenges posed by egocentric viewpoints and resource-constrained hardware environments. The remainder of this thesis is organized as follows. Chapter 2 reviews related work, including existing approaches to egocentric 3D human pose estimation and domain adaptation techniques. Chapter 3 describes the proposed model architecture design and training techniques in detail. Chapter 4 presents experimental results and

doi:10.6342/NTU202504485



evaluations, focusing on model validation and performance analysis. Chapter 5 details the complete system implementation and evaluates its performance under real-world conditions. Finally, Chapter 6 concludes the thesis by summarizing the contributions, presenting the limitations of the proposed approach, and discussing potential directions for future research.

doi:10.6342/NTU202504485

Chapter 2

Related Work

In this chapter, we review the foundational and recent advancements in 3D human pose estimation in Section 2.1, which provide the basis for egocentric pose estimation. Section 2.2 explores egocentric 3D human pose estimation, including its categorization into monocular and stereo methods, and the latest advancements in each. Section 2.3 introduces MLP-based architectures, highlighting their simplicity and challenges in data requirements. Finally, Section 2.4 reviews unsupervised domain adaptation (UDA) techniques and their role in bridging the domain gap between synthetic and real-world datasets.

2.1 3D Human Pose Estimation

3D human pose estimation has been a focal area of research due to its broad applications in computer vision, virtual reality, and human-computer interaction. Traditional methods predominantly relied on multi-view systems and depth sensors to infer 3D poses from 2D keypoints [18, 19]. With the emergence of deep learning and the availability of large-scale 2D and 3D training datasets [20, 19], convolutional neural networks (CNNs) have achieved significant advancements in pose estimation accuracy [21, 22, 23, 24, 25, 26].

Two primary approaches have emerged in recent studies. The first approach

doi:10.6342/NTU202504485

directly predicts 3D joint positions from input images [21, 22, 23, 24], while the second follows a two-stage pipeline—first estimating 2D poses and subsequently lifting them into 3D space [25, 26]. The former approach necessitates supervision using 3D pose labels corresponding to input images, which are often difficult to obtain in in-the-wild scenarios. To address this limitation, researchers have developed photorealistic synthetic datasets [27] leveraging parametric human body models [28] and computer graphics rendering.

Conversely, the two-stage approach benefits from the easier accessibility of 2D pose annotations for training the initial 2D keypoint detection network [29, 30, 31]. It then utilizes large-scale 3D motion capture datasets for 3D pose lifting, eliminating the need for paired 3D pose labels with images. Even simple architectures have demonstrated competitive performance under this paradigm [25].

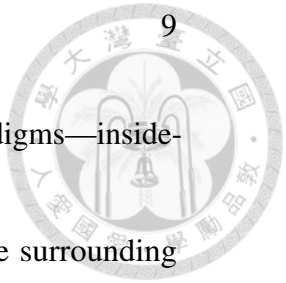
Current research trends focus on integrating 2D and 3D datasets while exploring weakly supervised, self-supervised, and even unsupervised learning frameworks to further reduce dependency on annotated data [32, 33, 34, 35, 36, 37]. With the rise of Transformer architectures, researchers are also investigating their potential applications in 3D human pose estimation [38, 39, 40, 41, 42], leveraging their superior capacity for modeling spatial dependencies and improving accuracy.

2.2 Egocentric 3D Human Pose Estimation

Egocentric 3D human pose estimation has gained increasing attention due to its potential for capturing body motion from a first-person perspective. Early approaches were limited to estimating upper-body movements and often relied on RGB-D input [43, 44], which restricts their applicability in in-the-wild scenarios. Capturing full-body motion poses greater challenges, making it a primary focus in current research.

As introduced in Section 2.1, 3D human pose estimation methods primarily adopt an outside-in approach, where external cameras observe the subject. In

doi:10.6342/NTU202504485



contrast, egocentric methods can be categorized into two main paradigms—inside-out and inside-in approaches.

Inside-out approaches estimate human poses by observing the surrounding environment. For example, Shiratori et al. [45] proposed systems utilizing 16 limb-mounted cameras combined with structure-from-motion (SfM) techniques have been employed to infer body poses based on environmental features. However, these methods often require precise calibration, static backgrounds, and are prone to errors caused by motion blur and dynamic scene elements. To simplify setups, Jiang et al. [46] reduced the number of cameras required by utilizing a chest-mounted camera to capture poses, but relying on external environmental information often results in reduced accuracy due to limited visibility of the subject's body.

Inside-in approaches, on the other hand, estimate poses directly by observing the body itself using wearable sensors or cameras. Von Marcard et al. [47] proposed systems using inertial measurement units (IMUs) can operate without cameras but require complex calibration procedures, and accuracy is further reduced as the number of sensors decreases. ControllerPose [48] introduced an alternative design, using handheld VR controllers combined with cameras, offer full-body capture but lack hands-free operation. Recently, the use of head-mounted devices equipped with downward-facing monocular or stereo fisheye cameras has emerged as the leading approach, enabling compact and practical solutions for egocentric 3D pose estimation.

2.2.1 Monocular Egocentric 3D Human Pose Estimation

Mo2Cap2 [49] proposed a real-time compact setup by mounting a fisheye camera on a baseball cap. They introduced a synthetic dataset for training and tested their model on real-world images. However, the synthetic data suffered from noticeable differences in lighting and background quality compared to real-world scenarios, limiting its performance. xR-EgoPose [2] provided a more realistic synthetic dataset and proposed an encoder-decoder architecture with 2D heatmap

doi:10.6342/NTU202504485

reconstruction, as illustrated in Figure 2.1, achieving promising results. Nevertheless, they fine-tuned their model using a subset of real-world test sets collected in controlled lab environments, which does not align well with scenarios where acquiring accurate 3D ground truth labels is challenging. SelfPose [50], an extension of xR-EgoPose, introduced an additional branch to predict per-joint rotations, enhancing the model's learning capacity.

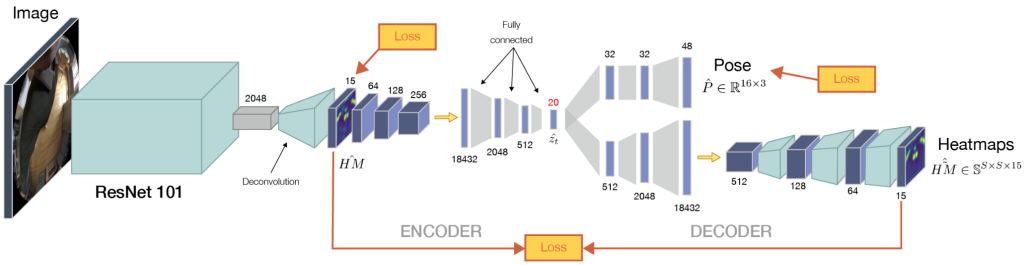


Figure 2.1: **The architecture of xR-EgoPose [2].**

Wang et al. [51] adopted a spatial-temporal optimization approach to predict global poses, achieving high precision and temporal stability. However, their reliance on backend optimization limited its suitability for real-time systems. EgoPW [52] was the first to leverage external views, combining methods of third-person views with egocentric inputs. Using weak external supervision, it improved learning by generating pseudo labels and applying adversarial losses to capture shared features between external and egocentric views. EgoFish3D [17] also incorporated third-person cameras and employed a self-supervised learning approach, although it required a multi-stage training strategy for convergence. Additionally, Wang et al. [53] utilized scene depth constraints to address issues like body-floating and body-environment penetration, improving interaction modeling with the scene. Other works have focused on whole-body pose estimation [54], predicting human meshes [55], or leveraging self-attention architectures to enhance learning [56].

doi:10.6342/NTU202504485

2.2.2 Stereo Egocentric 3D Human Pose Estimation

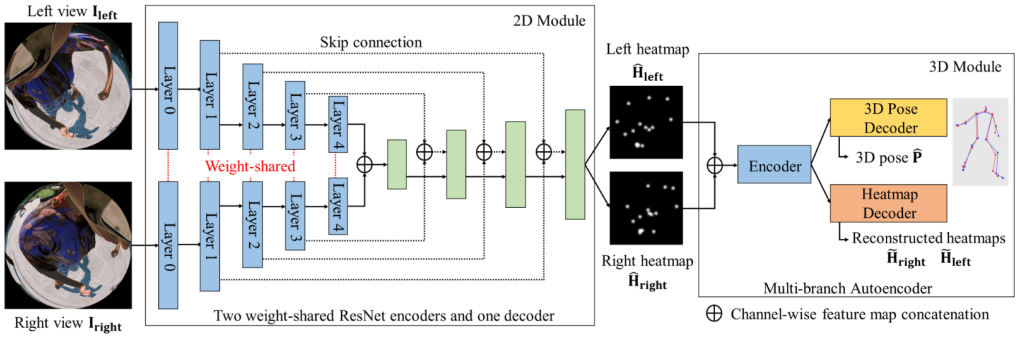
While monocular approaches require only a single fisheye camera, resulting in simpler setups and broader research adoption, they suffer from insufficient depth information. Different depths may project to similar 2D image patterns, causing ambiguities. In contrast, stereo approaches leverage depth information, providing a significant advantage. EgoCap [3] pioneered the use of stereo fisheye cameras mounted on a helmet at a 25 cm distance from the face, capturing most of the body. However, this setup was considered impractical and bulky, as shown in Figure 2.2. Reducing the distance between the camera and the body increases occlusion challenges, making the task more complex. EgoGlass [14] utilized two Raspberry Pi cameras and introduced pseudo-limb masks to address self-occlusion under limited viewpoints. However, their dataset remains unpublished, limiting reproducibility.

Figure 2.2: **The device proposed by EgoCap [3].**

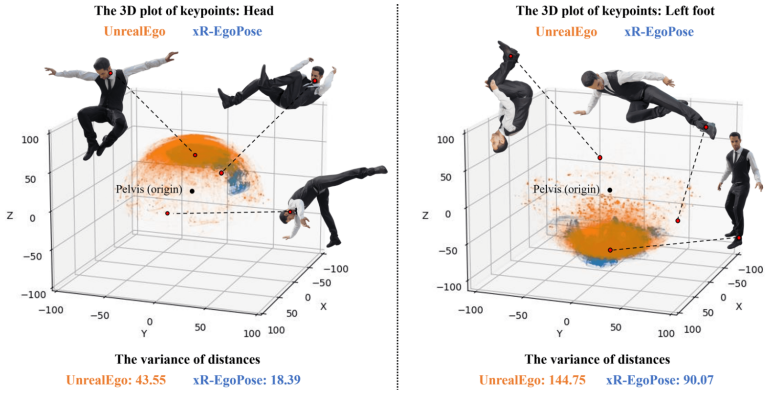
UnrealEgo [1] leveraged UnrealEngine [57] to create a large-scale stereo dataset with realistic scenarios and prioritized motion diversity, offering broader motion types and joint ranges than xR-EgoPose (Figure 2.3b). The architecture of UnrealEgo is depicted in Figure 2.3a. They demonstrated that weight-sharing feature extractors perform better and they also found that separate training was required to improve heatmaps, revealing limitations of using MSE loss for heatmap predictions. Ego3DPose [15] proposed perspective embedding heatmaps and two-path feature aggregation, incorporating limb information to enhance learning of stereo corre-

doi:10.6342/NTU202504485

spondences and perspectives. However, its model complexity limits scalability. EgoTAP [58] presented a heatmap-to-3D pose lifting method using ViT encoders to tackle inefficiencies in feature embedding in earlier approaches. Nevertheless, this two-stage approach relies heavily on high-quality heatmaps, which may not always be feasible in egocentric settings due to occlusions.



(a)



(b)

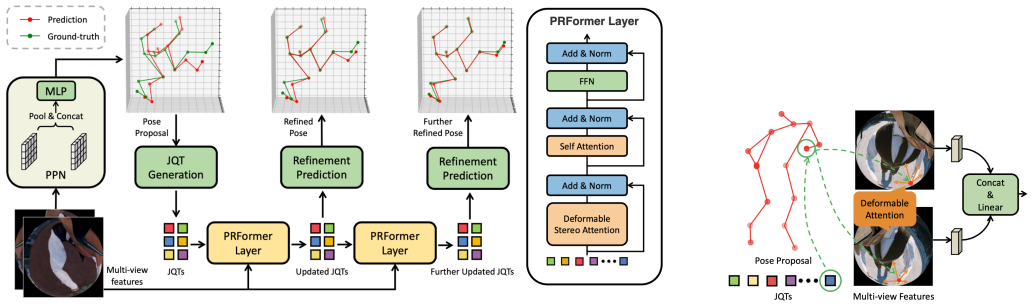
Figure 2.3: **Overview of UnrealEgo [1].** (a) The proposed architecture of UnrealEgo. (b) Motion diversity in the UnrealEgo dataset compared with xR-EgoPose.

UnrealEgo2 [7] introduced a transformer-based architecture for stereo videos, incorporating scene information and temporal context. They also released UnrealEgo2, a larger synthetic dataset, and UnrealEgo-RW, one of the few real-world stereo datasets, positioned closer to the face than EgoCap. While these advances

doi:10.6342/NTU202504485

provide valuable resources, UnrealEgo-RW's collection in lab environments still introduces domain gaps to in-the-wild scenarios.

EgoPoseFormer [4] adopted a DETR-style [59] transformer framework, arguing that using 2D heatmaps as inputs to regress 3D joints fails to fully utilize rich image information (Figure 2.4a). They proposed deformable stereo attention to refine initial poses by projecting them back to images and leveraging local stereo features through cross-attention with joint query tokens (Figure 2.4b). While achieving lightweight and state-of-the-art performance, its reliance on camera intrinsics and device-relative-to-camera-relative pose transformations hinders practical applications. Additionally, transformer-based and deformable convolution approaches introduce challenges for swift deployment on hardware platforms.



(a) (b)

Figure 2.4: **Overview of EgoPoseFormer [4].:** (a) The proposed architecture of EgoPoseFormer. (b) Deformable stereo attention in EgoPoseFormer.

In summary, stereo approaches demonstrate strong potential due to depth information and symmetric setups that make them practical for HMD and AR glasses [60]. However, existing solutions often feature large and complex models, lack real-world labeled datasets, and face deployment challenges. We aim to develop a lightweight and simple model capable of leveraging local stereo image features while maintaining competitive accuracy for practical use.

doi:10.6342/NTU202504485

2.3 MLP-Based Architecture

Recent years have seen a paradigm shift from CNNs to Transformers, driven in part by the availability of increasingly large datasets, as transformers are considered more effective when trained on extensive data. This shift has encouraged researchers to explore whether simple architectures like MLPs can achieve competitive performance. The MLP-Mixer [5] is a prime example of such efforts, featuring a simple design that relies only on matrix multiplications, data layout conversions, and scalar non-linearities to achieve promising results (Figure 2.5). However, the MLP-Mixer has also demonstrated that such architectures require large datasets to succeed; otherwise, they are prone to overfitting. Given this challenge, and the absence of extremely large datasets in our case, we adopt its simplicity while incorporating CNNs for image feature extraction to mitigate potential overfitting issues. This hybrid approach aims to balance architectural efficiency with robustness to dataset size limitations.

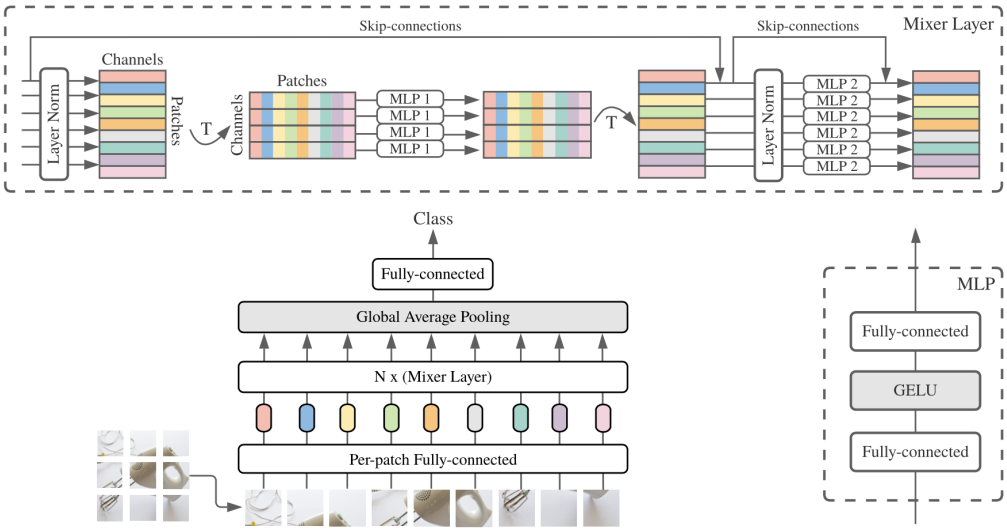


Figure 2.5: **The architecture of MLP-Mixer [5].**

doi:10.6342/NTU202504485

2.4 Unsupervised Domain Adaptation (UDA)

Despite the availability of synthetic datasets, domain gaps between synthetic and real-world data remain a significant challenge, often resulting in suboptimal performance in real-world applications. Unsupervised domain adaptation (UDA) aims to address this issue by aligning the feature distributions of target and source domains. Discrepancy-based methods use predefined statistical measures to minimize the distance between domains, effectively narrowing the gap [61, 62, 63]. On the other hand, adversarial-based approaches incorporate a domain classifier alongside the feature extractor. By applying adversarial loss, the feature extractor is trained to confuse the domain classifier, promoting domain-invariant feature learning. For instance, DANN [6] employs a gradient reversal layer to achieve this alignment, enabling robust adaptation, as shown in Figure 2.6.

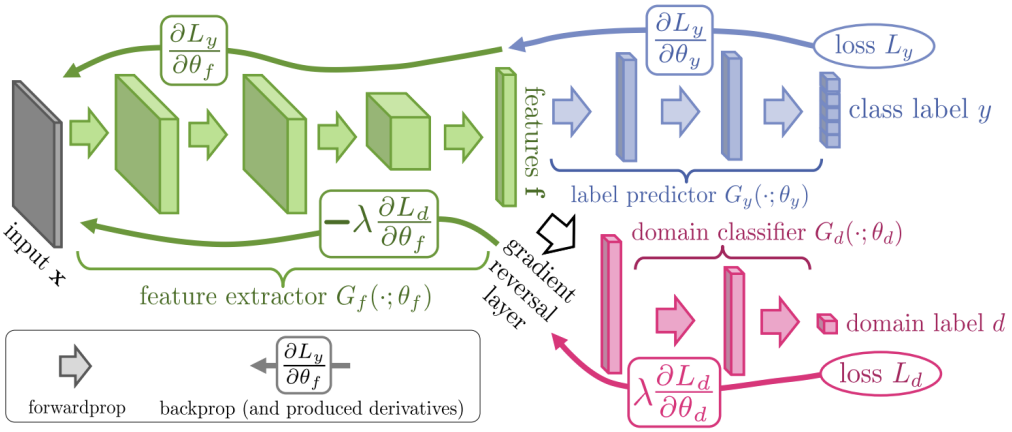


Figure 2.6: **The architecture of DANN [6].**

In 3D human pose estimation, weakly and self-supervised techniques have been widely adopted due to the relative ease of obtaining 2D labels compared to 3D labels. Many researchers leverage 2D labels along with 3D pose reprojection consistency for weak supervision [64, 35, 36, 37]. Zhou et al. [32] introduced geometric constraints derived from the human body to regularize 3D pose predictions without requiring extensive 3D labeled data. However, in egocentric settings,

doi:10.6342/NTU202504485

self-occlusion in images complicates the task of annotating 2D labels, making it challenging to achieve 3D-to-2D consistency learning.

doi:10.6342/NTU202504485

Chapter 3

Method

In this chapter, we provide a comprehensive explanation of our methodology. Section 3.1 introduces the overall architecture of our model, detailing its three core components. Section 3.2 focuses on the Attention-Guided Feature Extractor, elaborating on how multi-scale heatmaps and human masks are utilized for feature enhancement. Section 3.3 presents the Stereo Joint Mixer, describing its role in integrating stereo-attended features and estimating 3D joint positions. Finally, Section 3.4 explains the Domain Adaptation Training Framework, highlighting our approach to human prior loss and adversarial domain classifier training.

3.1 Architecture Overview

The proposed model includes three main components, as illustrated in Figure 3.1. First, the Attention-Guided Feature Extractor utilizes multi-scale 2D heatmaps and generated human masks to guide attention during feature extraction, enhancing spatially relevant representations. Next, the Stereo Joint Mixer employs a simplified MLP-Mixer [5] architecture to learn spatial and channel-wise dependencies within the attended features, effectively integrating information across dimensions. Finally, the Domain Classifier differentiates between real and synthetic image features, incorporating a Gradient Reversal Layer (GRL) [6] to adversarially train the feature

doi:10.6342/NTU202504485

extractor. The GRL reverses gradients during backpropagation, enabling domain-invariant feature learning and facilitating feature alignment.

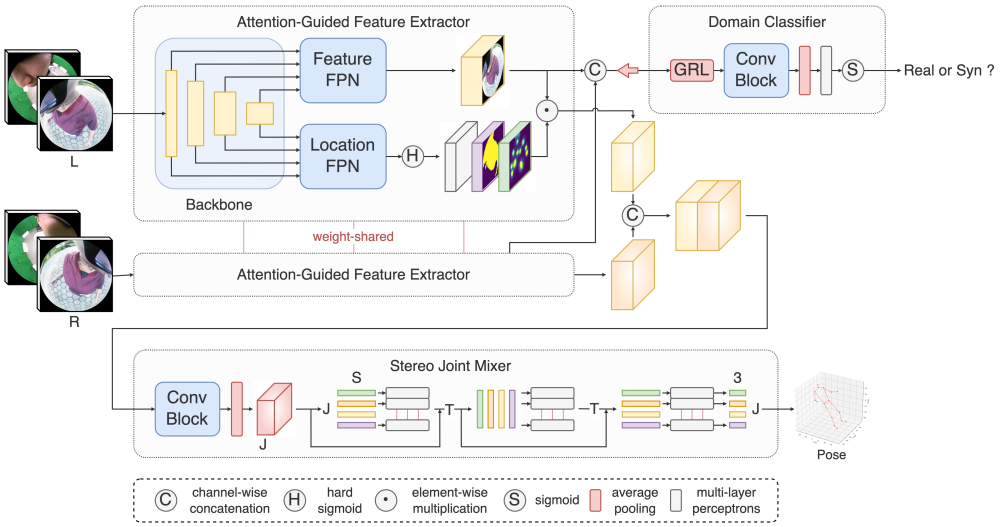


Figure 3.1: **The architecture of our EgoFocus model.**

3.2 Attention-Guided Feature Extractor (AFE)

We first pass stereo image pairs, denoted as $I_L, I_R \in \mathbb{R}^{H\times W\times 3}$ for the left and right images respectively, through an ImageNet-pretrained backbone network to extract downsampled features from four layers. These features are processed through two Feature Pyramid Networks (FPNs) [65]. The first FPN, referred to as the Feature FPN, extracts image features, denoted as F_L and F_R, where $F_L, F_R \in \mathbb{R}^{\tilde{H}\times\tilde{W}\times C}$ and $\tilde{H} = \frac{H}{4}, \tilde{W} = \frac{W}{4}$. The second, termed the Location FPN, focuses on extracting location features, denoted as $L_L, L_R \in \mathbb{R}^{\tilde{H}\times\tilde{W}\times C}$. A hard sigmoid function is applied to ensure values range between 0 and 1, while simultaneously guaranteeing the model is deployable on hardware. These two sets of features are combined using element-wise multiplication to generate attended features, refered to (3.1). The model utilizes shared weights for stereo image inputs, ensuring consistency across both views. This approach, as demonstrated in UnrealEgo [1], leads to

doi:10.6342/NTU202504485

improved performance. After this, the attended features A_L, A_R from the left and right images are concatenated along the channel dimension to form $A \in \mathbb{R}^{\tilde{H}\times\tilde{W}\times 2C}$ (3.2), which is passed into the Stereo Joint Mixer, as covered in Section 3.3.

$$\begin{aligned} A_L &= F_L \odot L_L \\ A_R &= F_R \odot L_R \end{aligned} \tag{3.1}$$

$$\begin{aligned} F &= \text{Concat}(F_L, F_R) \\ L &= \text{Concat}(L_L, L_R) \\ A &= \text{Concat}(A_L, A_R) \end{aligned} \tag{3.2}$$

Furthermore, the channels of the location features are divided into three components—global feature channels (C_g), multi-scale heatmap channels (C_h), and human mask channels (C_m), which will be introduced in detail in the following subsections. The total number of channels of the location features is represented as C, where $C = C_g + C_h + C_m$. To compare with EgoPoseFormer [4], we set C to 128, matching their configuration. Specifically, we manually allocate $C_g = 8$, $C_h = 90$ and $C_m = 30$. The concatenated location features, L, can be splited into $\widehat{G}$, $\widehat{H}$, and $\widehat{M}$, where $\widehat{G} \in \mathbb{R}^{\tilde{H}\times\tilde{W}\times 2C_g}$, $\widehat{M} \in \mathbb{R}^{\tilde{H}\times\tilde{W}\times 2C_h}$, and $\widehat{H} \in \mathbb{R}^{\tilde{H}\times\tilde{W}\times 2C_m}$. By integrating diverse location information, the model enhances learning effectiveness and stability.

3.2.1 Multi-Scale 2D Heatmaps Attention

The UnrealEgo synthetic dataset [1] provides 3D pose labels from each camera and 2D joint coordinates derived through camera intrinsics. In the egocentric view setting, self-occlusion reduces the reliability of 2D annotations, especially for the lower body. Hence, 3D pose labels are necessary to derive accurate 2D projections.

We leverage the UnrealEgo dataset's 2D joint coordinates to generate heatmaps by applying Gaussian kernels with varying standard deviations (σ), producing multi-scale representations. This strategy prevents the model's attention from focusing

doi:10.6342/NTU202504485

too narrowly on small regions. Specifically, we generate six scales of heatmaps, denoted as $N_s = 6$, with each scale containing 15 heatmaps (excluding the head joint), represented as $N_h = 15$. Thus, $C_h = N_s \times N_h = 90$. Through experiments, we determined that using $\sigma = [1, 1, 2, 2, 3, 3]$ for three scales, with each scale having two sets of channels, achieved the best performance. We visualize the generated ground truth multi-scale 2D heatmaps, as shown in Figure 3.2. Detailed experimental results can be found in Section 4.5.3.

These multi-scale heatmaps are then integrated into the mode learning using a local attention loss function $\mathcal{L}_{\text{mshm}}$, as detailed in Section 3.2.3, to optimize feature learning.

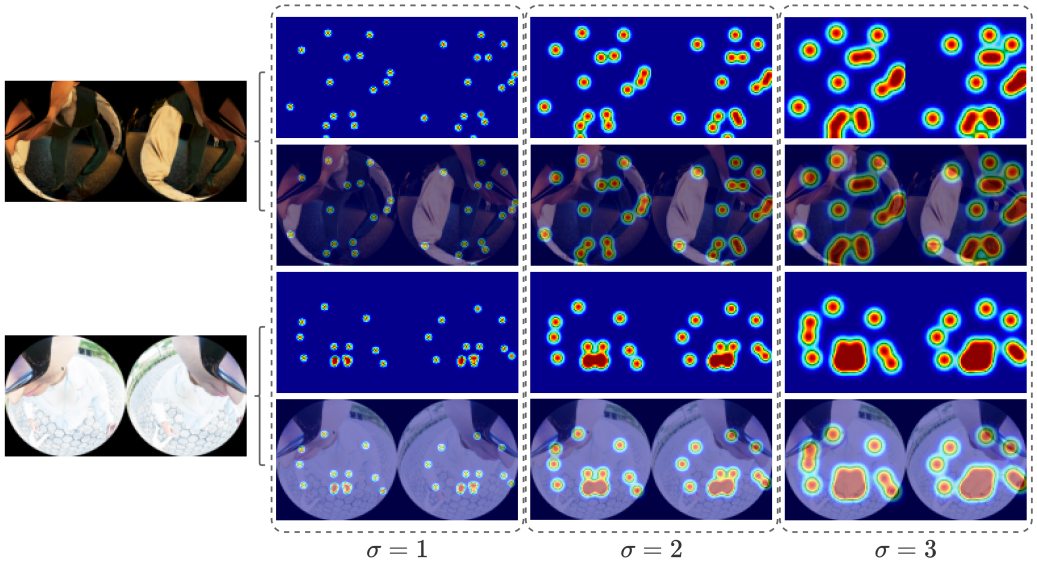


Figure 3.2: **The visualization of the ground truth multi-scale 2D heatmaps.** For each image pair, the top row shows the heatmaps in different scales, while the bottom row presents the results of overlaying the heatmaps semi-transparently on the corresponding images.

doi:10.6342/NTU202504485

3.2.2 Human Mask Attention

Beyond 2D joint positions, body regions such as limbs also carry pose-related information. Additionally, using body regions for attention helps reduce the impact of environmental noise, especially in cluttered environments that may cause incorrect predictions by the model. We generate human masks using the Segment Anything Model (SAM) [66]. Note that the 2D joint coordinates serve as anchors to guide mask generation, ensuring accurate segmentation.

We visualize the generated human masks, as shown in Figure 3.3. While the masks are guided by ground truth 2D joint coordinates, failures may occur. For example, Figure 3.3b illustrates partial failures on the left side and extensive failures on the right side caused by poor lighting conditions.

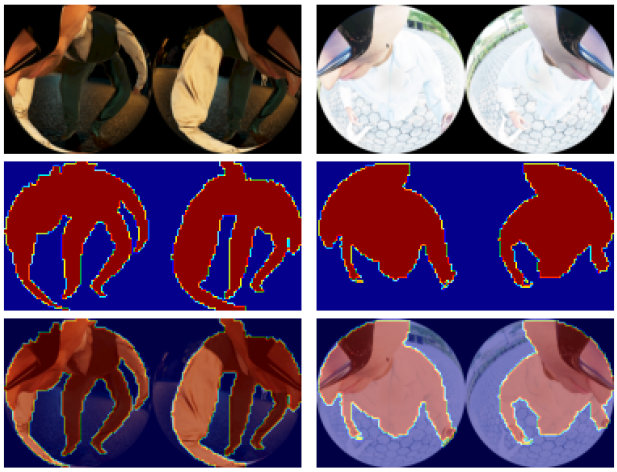
The generated human masks are then incorporated into the model to provide additional spatial context, improving robustness against background interference. Similar to the multi-scale 2D heatmaps, the human masks are optimized using the local attention loss function $\mathcal{L}_{\text{mask}}$, as discussed in Section 3.2.3.

3.2.3 Local Attention Loss

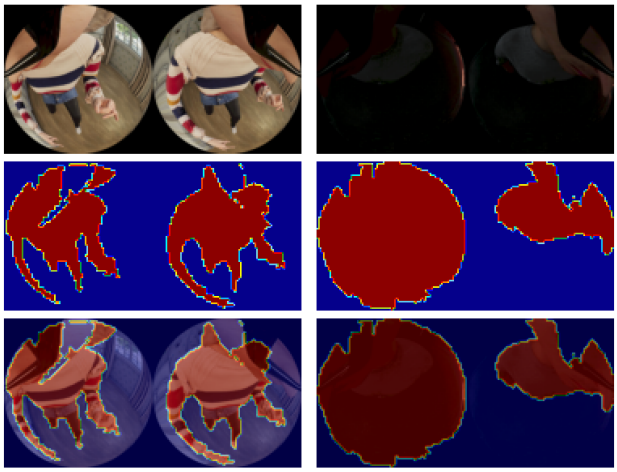
Previous approaches, such as xR-EgoPose [2] and UnrealEgo [1], utilize mean squared error (MSE) loss as the heatmap loss function. However, due to the small active regions in joint heatmaps, MSE loss converges slowly and is adversely affected by pose loss, hindering end-to-end training. For instance, UnrealEgo employs a two-stage training process to address these challenges.

To overcome this limitation, we propose a novel loss function, for supervising both heatmaps (3.3) and masks (3.4), which integrates high-value loss and low-value regularization. The high-value loss targets regions with high heatmap confidence or nonzero mask values, using a log-softmax (denoted as LS) operation to encourage higher values in these areas. Note that the multi-scale heatmap loss is further normalized by different σ of Gaussian kernels. Conversely, the low-value regularization penalizes regions with low heatmap confidence or zero mask values,

doi:10.6342/NTU202504485



(a) Success cases of generated human masks.



(b) Failure cases of generated human masks.

Figure 3.3: **The visualization of the generated human mask.** For each image pair, the medium row shows the mask, while the bottom row presents the results of overlaying the mask semi-transparently on the corresponding images. In Figure (b), The left side shows partial failures due to segmentation errors, while the right side illustrates extensive failures caused by poor lighting conditions.

driving predictions closer to zero. Note that H and M represent concatenated ground truth heatmaps and the SAM-generated human mask. This combined loss enhances feature learning by maintaining focus on key regions while suppressing

doi:10.6342/NTU202504485

noise, thereby enabling robust end-to-end training. We will discuss a comparison between local attention loss and MSE loss in Section 4.5.2.

$$\mathcal{L}_{\text{mshm}} = \frac{1}{2C_h} \sum_{i=1}^{2C_h} \sum_{(\tilde{H},\tilde{W})} \left(-\frac{\text{LS}(\widehat{\mathbf{H}}_i)}{\sigma_i^2} \cdot \mathbf{1}_{\mathbf{H}_i > 0.5} + \lambda_{\text{reg}} \cdot \widehat{\mathbf{H}}_i \cdot \mathbf{1}_{\mathbf{H}_i \leq 0.5} \right), \lambda_{\text{reg}} = 0.0001 \tag{3.3}$$

$$\mathcal{L}_{\text{mask}} = \frac{1}{2C_m} \sum_{i=1}^{2C_m} \sum_{(\tilde{H},\tilde{W})} \left(-\text{LS}(\widehat{\mathbf{M}}_i) \cdot \mathbf{1}_{\mathbf{M}=1} + \lambda_{\text{reg}} \cdot \widehat{\mathbf{M}}_i \cdot \mathbf{1}_{\mathbf{M}=0} \right), \lambda_{\text{reg}} = 0.01 \tag{3.4}$$

3.3 Stereo Joint Mixer (SJM)

The attended features A are first processed through a convolution block to reduce the channel dimension from the feature space $2C$ to the number of joints $J = 16$. This step encodes spatial information for each joint by integrating stereo features. Next, average pooling with a size of 4 is applied to decrease resolution and computational complexity. The resulting feature representation of size $J \times S$, where $S = \frac{\tilde{H}}{4} \times \frac{\tilde{W}}{4} = 256$, is then passed through a 2-layer MLPs. This 2-layer MLPs operates on the spatial dimension, which corresponds to S, and is shared across all joints. Afterward, the output is transposed and processed by another 2-layer MLPs shared across the spatial dimension. This mechanism effectively captures dependencies between joints and spatial features. Additionally, skip connections are utilized to enhance learning stability. Unlike MLP-Mixer [5], which employs multiple layers, we use only one layer while achieving sufficient accuracy. Finally, a 3-layer MLPs projects the spatial dimension to three dimensions, representing the 3D position of each joint. This prediected 3D human pose is denoted as $\widehat{P} \in \mathbb{R}^{J \times 3}$. We employ pose MPJPE loss $\mathcal{L}_{\text{pose}}$ (3.5) and bone vector loss $\mathcal{L}_{\text{bone}}$ (3.6) using ground truth 3D pose P in UnrealEgo syntetic dataset.

doi:10.6342/NTU202504485

$$\mathcal{L}_{\text{pose}} = \frac{1}{J}\sum_{i=1}^{J} \|\widehat{\mathbf{P}}_i - \mathbf{P}_i\|_2 \tag{3.5}$$

$$\mathcal{L}_{\text{bone}} = \frac{1}{J-1}\sum_{i=1}^{J-1} \|\widehat{\mathbf{P}}_i - \widehat{\mathbf{P}}_i^{\text{parent}} - (\mathbf{P}_i - \mathbf{P}_i^{\text{parent}})\|_2 \tag{3.6}$$

3.4 Domain Adaptation Training Framework

3.4.1 Human Prior Loss

To ensure plausible pose predictions for the target domain without ground truth 3D pose labels, we introduce the Human Prior Loss. This loss function $\mathcal{L}_{\text{prior}}$ is designed to regularize predictions by enforcing prior knowledge of human skeletal structure (3.7). Specifically, the loss evaluates bone lengths and bone angles against predefined distributions, where the mean and standard deviation are calculated from the UnrealEgo training set, denoted as μ_ℓ, σ_ℓ for bone lengths and μ_θ, σ_θ for bone angles. Predicted bone lengths $\hat{\ell}$ and angles $\hat{\theta}$ deviating by more than twice the standard deviation from their respective means are penalized. Bone length and angle deviations exceeding twice their standard deviation are penalized quadratically after normalization, ensuring alignment with realistic human poses.

$$\mathcal{L}_{\text{prior}} = \frac{1}{J}\sum_{i=1}^{J}\left[\left(\frac{|\hat{\ell}_i - \mu_\ell|}{\sigma_\ell}\right)^2 \cdot \mathbf{1}_{|\hat{\ell}_i-\mu_\ell|>2\sigma_\ell} + \left(\frac{|\hat{\theta}_i - \mu_\theta|}{\sigma_\theta}\right)^2 \cdot \mathbf{1}_{|\hat{\theta}_i-\mu_\theta|>2\sigma_\theta}\right] \tag{3.7}$$

3.4.2 Adversarial Domain Classifier Training

Our approach employs a domain classifier with domain loss, inspired by the Domain-Adversarial Neural Network (DANN) framework [6]. We apply the domain classifier to image features F rather than attended features A, as the primary differences between the source and target domains are related to environmental factors such as background and lighting. Using attended features, which are sparse, results in less effective domain alignment.

doi:10.6342/NTU202504485

We feed the image features from the source and target domains, denoted as F_{src} and F_{tgt}, respectively, into the domain classifier. The domain classifier architecture begins with a gradient reversal layer (GRL), which reverses the gradient during backpropagation. This mechanism enables the feature extractor to align the features of the two domains, confusing the domain classifier and promoting domain-invariant features.

Following the gradient reversal layer, the domain classifier consists of convolutional blocks, average pooling, and a 3-layer MLPs with a dropout rate of 0.5, as illustrated in Figure 3.1. The final output is a sigmoid activation that predicts the probability of the features belonging to the target domain. Since this process is only applied during training and does not need to be deployed, we use a standard sigmoid activation instead of a hard sigmoid.

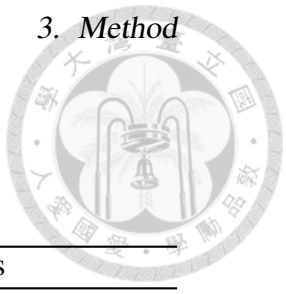
The predicted probabilities $\hat{d}$ are compared against domain labels d using binary cross-entropy (BCE) loss $\mathcal{L}_{\text{domain}}$ (3.8). The complete training process is detailed in Algorithm 1. To address the imbalance between the source and target domain data, as the source domain has significantly more data, we balance the datasets during training. Finally, our total loss is defined as (3.9).

$$\mathcal{L}_{\text{domain}} = -\left[d \log(\hat{d}) + (1-d) \log(1-\hat{d})\right] \tag{3.8}$$

$$\mathcal{L} = \underbrace{\lambda_{\text{mh}}\mathcal{L}_{\text{mshm}} + \lambda_{\text{ma}}\mathcal{L}_{\text{mask}} + \lambda_{\text{p}}\mathcal{L}_{\text{pose}} + \lambda_{\text{b}}\mathcal{L}_{\text{bone}}}_{\text{source}} + \underbrace{\lambda_{\text{pr}}\mathcal{L}_{\text{prior}}}_{\text{target}} + \underbrace{\lambda_{\text{d}}\mathcal{L}_{\text{domain}}}_{\text{both}} \tag{3.9}$$

During source-only training, we set λ_{pr} and λ_{d} to 0 to exclude target domain-related losses. The specific values of λ for each component will be provided in Section 4.2.

doi:10.6342/NTU202504485



Algorithm 1: Unsupervised Domain Adaptation Training Process

Data: *Target data*, *Source data*

Definitions:

AFE = Attention-Guided Feature Extractor;

SJM = Stereo Joint Mixer;

DC = Domain Classifier

r = Balance Ratio;

foreach *Target data* **do**

  Step 1: Train with source data to balance datasets;

  foreach *Source data in range* $r-1$ **do**

    Source images $\xrightarrow{\text{AFE}} F_{\text{src}}, L_{\text{src}}, A_{\text{src}}$;

    $A_{\text{src}} \xrightarrow{\text{SJM}} \hat{P}_{\text{src}}$;

    Compute source losses: $\mathcal{L}_{\text{mshm}}, \mathcal{L}_{\text{mask}}, \mathcal{L}_{\text{pose}}, \mathcal{L}_{\text{bone}}$;

  end

  Step 2: Train with paired source and target data;

  Select one pair: (*Source data*, *Target data*);

  Source images $\xrightarrow{\text{AFE}} F_{\text{src}}, L_{\text{src}}, A_{\text{src}}$;

  Target images $\xrightarrow{\text{AFE}} F_{\text{tgt}}, L_{\text{tgt}}, A_{\text{tgt}}$;

  $A_{\text{src}} \xrightarrow{\text{SJM}} \hat{P}_{\text{src}}$;

  $A_{\text{tgt}} \xrightarrow{\text{SJM}} \hat{P}_{\text{tgt}}$;

  $F_{\text{src}}, F_{\text{tgt}} \xrightarrow{\text{DC}} \hat{d}_{\text{src}}, \hat{d}_{\text{tgt}}$;

  Compute source losses: $\mathcal{L}_{\text{mshm}}, \mathcal{L}_{\text{mask}}, \mathcal{L}_{\text{pose}}, \mathcal{L}_{\text{bone}}$;

  Compute target loss: $\mathcal{L}_{\text{prior}}$;

  Compute domain loss: $\mathcal{L}_{\text{domain}}$;

end

doi:10.6342/NTU202504485

Chapter 4

Experiments

In this chapter, we describe the experimental setup, datasets, implementation details, and results of our study. Section 4.1 introduces the datasets employed for training, evaluation, and domain adaptation, including synthetic and real-world datasets. Section 4.2 outlines the training and implementation details. Section 4.3 presents the main results, focusing on performance evaluation and domain adaptation. Section 4.4 provides qualitative results, visualizing model predictions and highlighting attention features. Finally, Section 4.5 conducts ablation studies to examine the impact of different design choices on model performance.

4.1 Datasets

4.1.1 UnrealEgo Dataset

UnrealEgo is a synthetic dataset provided by Akada et al. [1]. It is constructed using UnrealEngine [57], a professional game development platform, to simulate a realistic environment with 3D human models sourced from RenderPeople [67]. The dataset features a pair of downward-facing virtual fisheye cameras mounted on a glasses frame. The cameras are positioned 12 cm apart, providing a wide field of view of 170 degrees.

UnrealEgo is notable for its realism compared to previous synthetic datasets

doi:10.6342/NTU202504485

such as Mo2Cap2 [49] and xR-EgoPose [2] (see Figure 4.2a).. It offers large-scale data with high motion diversity, making it suitable for training and evaluating pose estimation models. The dataset includes 17 realistic 3D human models, comprising nine females and eight males, with diverse skin tones and clothing types. Additionally, it features 14 realistic 3D environments, including both indoor and outdoor scenes.

The dataset provides about 450,000 image pairs with a resolution of 1024x1024 pixels, captured at 25 frames per second. The dataset is divided into 357,317 training, 46,155 validation, and 48,138 test image pairs. Each image is labeled with 3D joint positions, camera positions, and 2D coordinates of reprojected joint positions in the fisheye view.

We employed UnrealEgo for both training and evaluation purposes. For training, we utilized its 3D pose labels and 2D joint position labels. The 2D joint positions were used to generate heatmaps based on Gaussian kernels and to create human masks, which served as anchors for Segment Anything Model (SAM) [66] . For evaluation, we tested on UnrealEgo to assess the performance improvement of our proposed model—combining attention-guided features and a stereo joint mixer—over the baseline model, and also compared the result with existing stereo methods.

4.1.2 UnrealEgo-RW (Real-World) Dataset

UnrealEgo-RW is a real-world dataset developed by the same team that created UnrealEgo (see Figure 4.2b). This dataset uses a custom device based on a helmet, equipped with two RIBCAGE RX0 II cameras and two FUJINON FE185C057HA-1 fisheye lenses. The cameras are positioned 12 cm apart and placed 2 cm away from the user's face. The captured images are cropped to match the 170-degree field of view setup used in UnrealEgo. The dataset includes recordings of 16 subjects in a multi-view motion capture studio, providing ground truth 3D poses for 16 joints. The dataset contains about 130,000 image pairs at a resolution of

doi:10.6342/NTU202504485

872x872 pixels, captured at 25 frames per second, which is divided into 76,315 training, 30,083 validation, and 24,012 test image pairs. Since the test set does not provide ground truth labels, evaluations are performed on the validation set.

Earlier datasets, such as EgoCap [3], used camera setups positioned 25 cm away from the face, which were less compact and impractical for application purposes. In contrast, this dataset is the first egocentric stereo real-world dataset, offering a more practical design. We utilized UnrealEgo-RW primarily for evaluation purposes in our domain adaptation framework, leveraging its ground truth annotations to assess performance and validate the generalizability of our models.

4.1.3 Our RealEgo (Unlabeled) Dataset

While UnrealEgo-RW captures real-world data, it is limited to a controlled laboratory environment, which may not reflect real-life scenarios. To address this limitation, we collected our own dataset, RealEgo, specifically for domain adaptation purposes (see Figure 4.2c).

Our setup features a lightweight and affordable device based on a bicycle helmet, equipped with two ELP-USBFHD01M [68] camera modules with 180-degree fisheye lenses, as shown in Figure 4.1. This design shares similarities with UnrealEgo-RW in that the cameras are positioned 12 cm apart, but they are approximately 8 cm away from the eyes, compared to 2 cm in UnrealEgo-RW. While the distance is slightly greater, it remains significantly closer than EgoCap's 25 cm, offering a practical balance between portability and usability.

RealEgo consists of about 96,000 image pairs, each with a resolution of 1024x1024 pixels, cropped using UnrealEgo's circular mask. The dataset spans two indoor environments and includes 15 subjects performing actions at three self-defined motion levels:

1. **Easy:** Minimal movement or upper-limb actions, such as standing, waving, and clapping.

doi:10.6342/NTU202504485

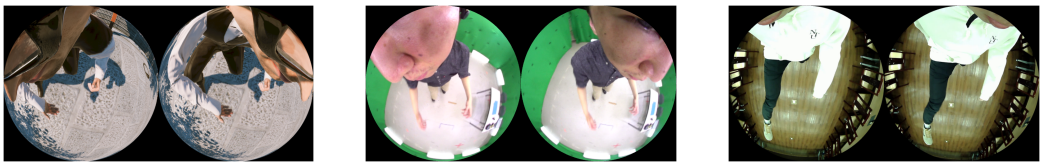
2. **Medium:** Larger or leg-involved movements, such as kicking, walking, and punching.

3. **Hard:** Full-body movements and occluded actions, including squatting, stretching, and arbitrary motions.

(a) Front-view (b) Side-view (c) Fisheye camera [68]

Figure 4.1: **Our EgoVision device.**

These motion levels can be utilized for downstream tasks in future research. Additionally, the continuous image sequences in our dataset make it well-suited for studying temporal information in motion analysis. For details, each sequence is recorded at 30 frames per second with a duration of 20 seconds. The test set comprises subjects S1 and S3, totaling 19,776 image pairs, while the training set includes the remaining subjects with 76,315 image pairs. Since RealEgo does not include ground truth annotations, we primarily used it to visualize results within our domain adaptation framework, demonstrating its applicability in adapting models to real-world conditions.



(a) UnrealEgo [1] (b) UnrealEgo-RW [7] (c) Our RealEgo

Figure 4.2: **Example of the stereo image pair from each dataset [1, 7].**

doi:10.6342/NTU202504485

4.2 Training and Implementation Details

Our experiments were implemented using PyTorch and conducted on an RTX 2080Ti GPU. Input images were resized to 256x256, while ground-truth 2D heatmaps and human masks were resized to 64x64. All input images were normalized using the standard ImageNet mean and standard deviation. For 3D pose representation, we used device-relative coordinates, computed as the average 3D pose from the left and right camera perspectives.

We conducted three types of training experiments. First, we trained the model on the UnrealEgo dataset without a domain classifier to validate the effectiveness of our model structure. Next, we performed UDA training, with the source domain as UnrealEgo and the target domain as UnrealEgo-RW, following the process outlined in Section 3.4.2. Finally, we applied UDA training again while with the source domain as UnrealEgo and the target domain as RealEgo, which is the model for our deployment on final system in Chapter 5.

We used ResNet18 as the backbone, initialized with pre-trained weights from ImageNet. The training spanned 12 epochs, starting with a learning rate of 0.001 and a weight decay of 0.0005. A multi-step scheduler reduced the learning rate by a factor of 0.1 at the 8th and 10th epochs. The AdamW optimizer was employed for optimization. These settings are largely consistent with those used in EgoPoseFormer [4].

The batch size was set to 32 for training without UDA and reduced to 16 for UDA training due to memory constraints. The loss weights were configured as follows: $\lambda_{\text{mh}} = 0.5$, $\lambda_{\text{ma}} = 0.001$, $\lambda_{\text{p}} = 1.0$, $\lambda_{\text{b}} = 0.1$, $\lambda_{\text{d}} = 0.5$, and $\lambda_{\text{pr}} = 0.05$. In the Domain Classifier, the Gradient Reversal Layer (GRL) was warm-started from 0.5x to 1.0x over 1000 iterations.

For model selection, we used the best-performing model on the validation set when ground truth was available. Otherwise, the model from the final epoch was selected.

doi:10.6342/NTU202504485

4.3 Main Results

We evaluate model performance using two widely adopted metrics: Mean Per Joint Position Error (MPJPE) and Procrustes Aligned Mean Per Joint Position Error (PA-MPJPE). MPJPE (4.1) measures the mean Euclidean distance between corresponding joints of the predicted and ground truth poses, while PA-MPJPE (4.2) improves upon MPJPE by first applying Procrustes alignment, which eliminates differences due to translation, rotation, and scale.

$$\text{MPJPE} = \frac{1}{J}\sum_{i=1}^{J}\|\widehat{\mathbf{P}}_i - \mathbf{P}_i\|_2 \tag{4.1}$$

$$\text{PA-MPJPE} = \frac{1}{J}\sum_{i=1}^{J}\|\widehat{\mathbf{P}}_i^{pa} - \mathbf{P}_i\|_2 \tag{4.2}$$

where $\widehat{\mathbf{P}}_i$, $\widehat{\mathbf{P}}_i^{pa}$ and $\mathbf{P}_i$ represent the predicted, Procrustes-aligned and ground truth positions of joint i, respectively, and J is the total number of joints.

We first evaluate the performance of our model trained on the UnrealEgo dataset [1]. As shown in Table 4.1, our approach outperforms existing methods and demonstrates competitive results, trailing only slightly behind EgoPoseFormer [4] on PA-MPJPE.

Table 4.1: **Comparison of model performance on the UnrealEgo dataset.**

Method	MPJPE (mm) ↓	PA-MPJPE (mm) ↓
SimpleEgo [69]	80.0	67.6
UnrealEgo [1]	79.0	59.2
Ego3DPose [15]	60.8	48.4
UnrealEgo2 [7]	50.0	40.5
EgoTAP [58]	41.4	35.4
Ours (EgoFocus)	37.7	34.7
EgoPoseFormer [4]	34.5	33.4

doi:10.6342/NTU202504485

To validate the effectiveness of each component in our architecture, we further analyze the impact of individual submodules. We use the Pose Proposal Network from EgoPoseFormer [4] as a hardware-friendly baseline architecture, illustrated in Figure 4.3. This baseline comprises a feature extractor, followed by global average pooling and multi-layer perceptrons for human pose prediction. To highlight the improvements introduced by our optimized Attention-Guided Feature Extractor and Stereo Joint Mixer, we present comparative results in Table 4.2. We observe that incorporating the Attention-Guided Feature Extractor and using location features for attention leads to significant performance improvements. Additionally, the Stereo Joint Mixer also reduces errors by effectively capturing both channel-wise and spatial-wise features. When combining both components, our model achieves a 43.3% reduction in MPJPE and a 37.4% reduction in PA-MPJPE comparing to the baseline model, demonstrating the effectiveness of these enhancements.

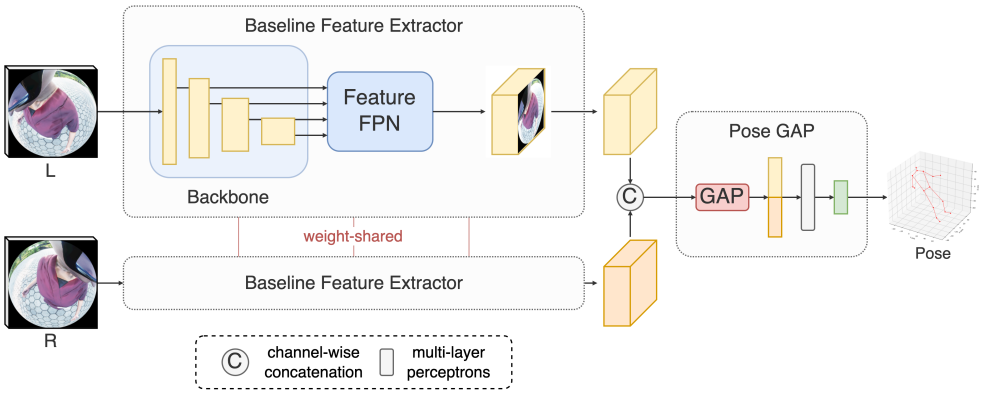


Figure 4.3: **The architecture of our baseline model.**

Next, we assess the effectiveness of our Unsupervised Domain Adaptation (UDA) framework. Since no prior work has explored UDA for stereo egocentric 3D human pose estimation, we compare our method against models directly trained on UnrealEgo. Please refer to Table 4.3. Due to the absence of ground truth labels in the UnrealEgo-RW test set, evaluations are conducted on the validation set.

The results indicate that the Unsupervised Domain Adaptation (UDA) frame-

doi:10.6342/NTU202504485

Table 4.2: **Comparative results of submodules in the architecture.** Note that the experiments of model containing AFE were conducted with $\sigma = [1, 2, 3, 4, 5, 6]$, which is not our best model. For the model using Baseline Feature Extractor (FE), only pose MPJPE loss and bone vector loss were used.

Method	MPJPE (mm) ↓	PA-MPJPE (mm) ↓
Baseline (FE+GAP)	67.418	56.153
Location Attention Feature (AFE+GAP)	49.100	42.074
Enhanced Pose Estimator (FE+SJM)	43.842	39.121
All (AFE+SJM)	**38.222**	**35.157**

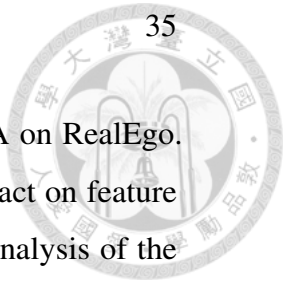
work provides noticeable performance improvements when applied to the stereo egocentric 3D human pose estimation task. However, the improvements are relatively modest, suggesting that while UDA helps to bridge the domain gap, it does not fully address the challenges posed by domain shifts. Direct training on UnrealEgo-RW can be considered the lower bound of error, as it utilizes ground truth annotations. However, collecting 3D pose annotations is highly resource-intensive, making heavy reliance on ground truth impractical for real-world applications.

Table 4.3: **Effectiveness of Unsupervised Domain Adaptation.** Note that evaluations were conducted on UnrealEgo-RW's validation set due to the absence of ground truth labels on test set.

Training Method	MPJPE (mm) ↓	PA-MPJPE (mm) ↓
Train on UnrealEgo	201.426	129.452
Train with UDA	158.693	100.359
Train on UnrealEgo-RW	68.994	52.375

For further insights into the visual quality of predictions and the specific areas where the UDA framework excels or falls short, we will discuss qualitative results

doi:10.6342/NTU202504485



in detail in Section 4.4, including the visualization result of UDA on RealEgo. This will provide a clearer understanding of the framework's impact on feature alignment and pose estimation accuracy. Finally, for a detailed analysis of the overall system performance, please refer to Section 5.3.

4.4 Qualitative Results

We visualize the predicted poses and attention features to demonstrate that our model effectively learns key regions. To visualize the attention features, we first compute the absolute values of location features across the same type, sum them along the channel dimension, and then normalize and overlay them onto the original input image. This process highlights areas of focus during the model's prediction process.

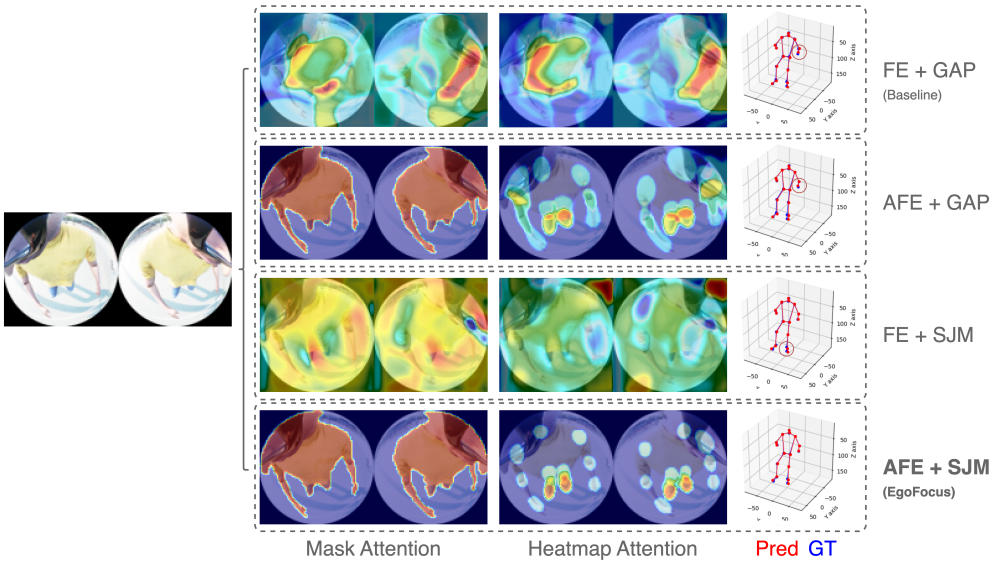
First, we compare the results from the baseline model to our improved architectures. Figure 4.4 illustrates that the baseline model does not focus on specific regions, causing it to learn features unrelated to human poses and resulting in larger errors in pose estimation. In contrast, the improved model clearly focuses on human joints and relevant areas, achieving better alignment with human pose structures.

Second, Figure 4.5 and Figure 4.6 demonstrates results on the UnrealEgo-RW and RealEgo datasets, respectively, highlighting the domain gap caused by training solely on synthetic data. Applying domain adversarial loss and human prior loss effectively reduces this gap, constraining predictions to reasonable human poses.

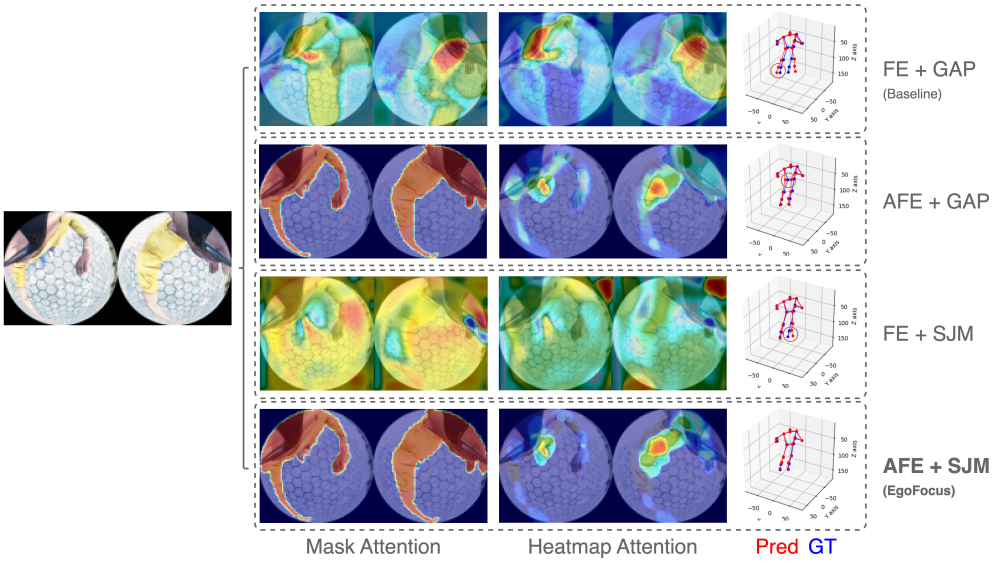
We also generate the t-SNE visualization by randomly sampling 4000 data points from each dataset. We then extract stereo image features and apply global average pooling to reduce the dimension before performing t-SNE. The t-SNE visualizations in Figure 4.7 and Figure 4.8 show that the feature extractor, after UDA, successfully mixes source and target image feature representations.

We also present one of the failure cases in Figure 4.9, revealing instances where

doi:10.6342/NTU202504485

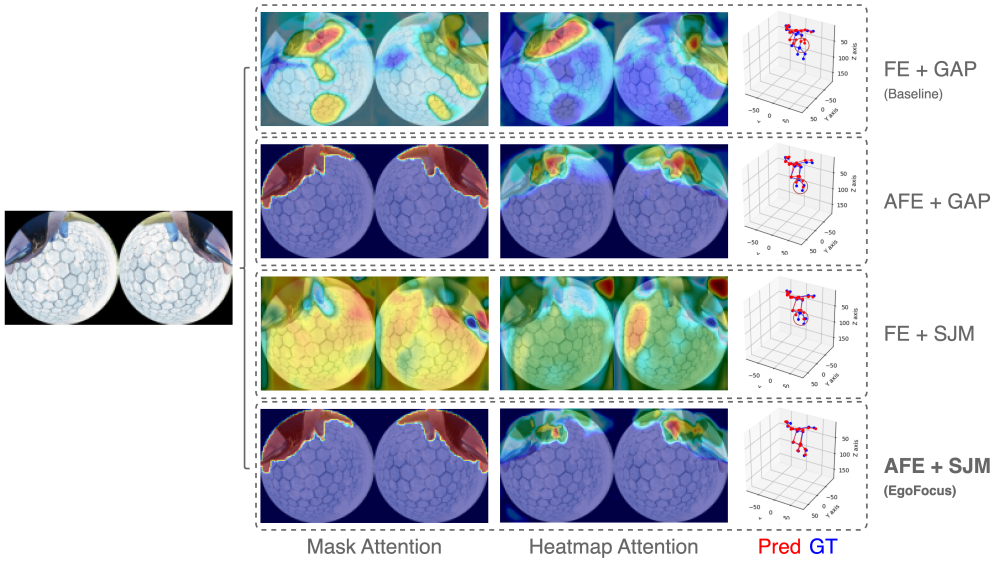


(a)

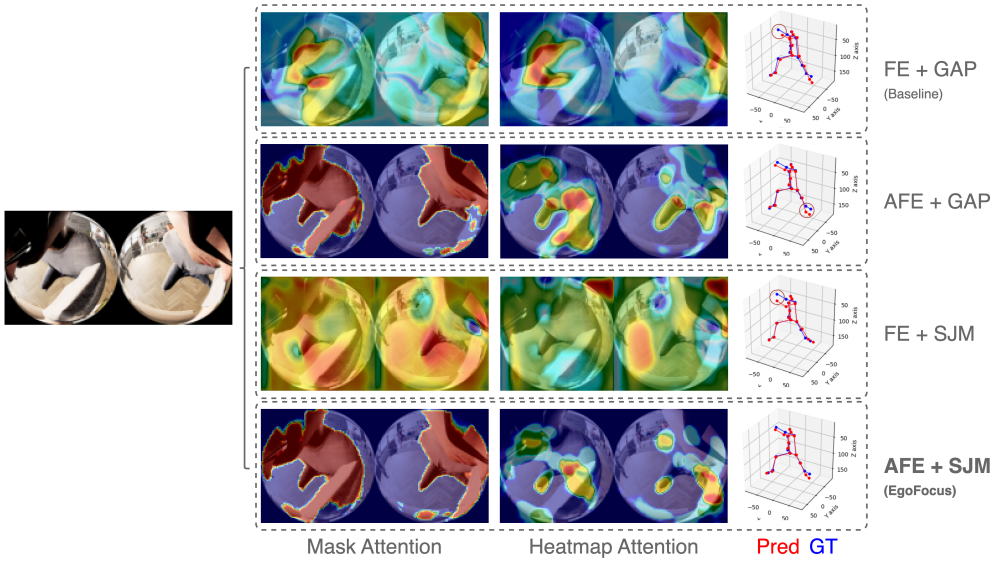


(b)

doi:10.6342/NTU202504485



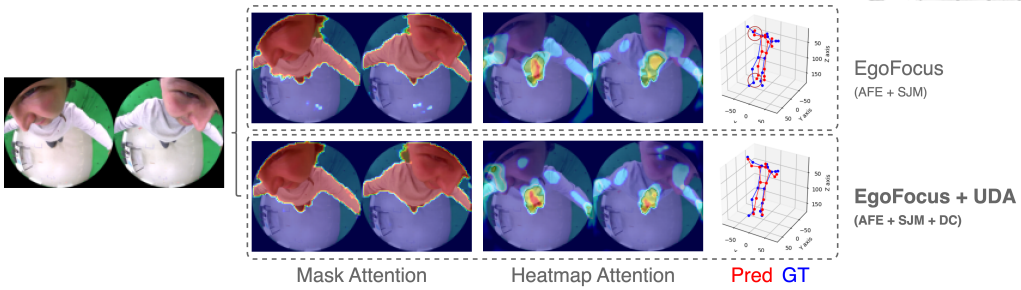
(c)



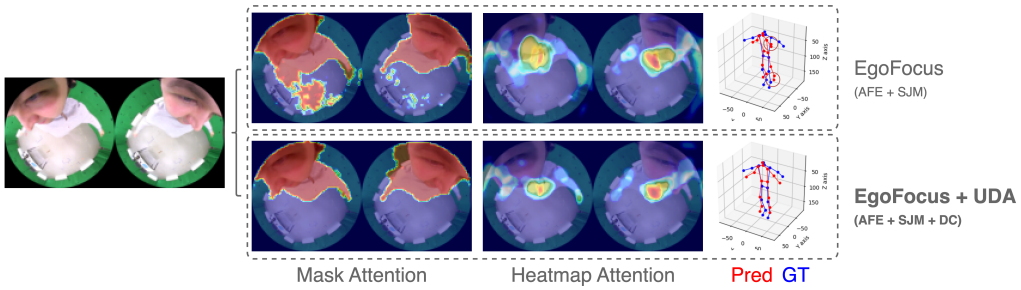
(d)

Figure 4.4: **Visualization of model predictions on UnrealEgo from baseline to improved architecture.**

doi:10.6342/NTU202504485



(a)



(b)

Figure 4.5: **Visualization of model predictions on the UnrealEgo-RW validation dataset.**

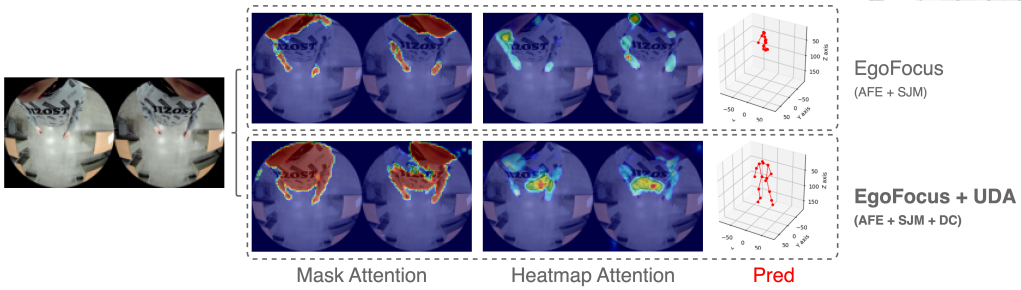
UDA fails to handle certain inputs, providing directions for future improvements.

Finally, it is worth mentioning that we generate human masks using the Segment Anything Model [66], which may occasionally exhibit minor artifacts, as shown in Figure 3.3b. However, as illustrated in Figure 4.10, the model demonstrates robustness against such imperfections, effectively tolerating noise to produce reliable human masks.

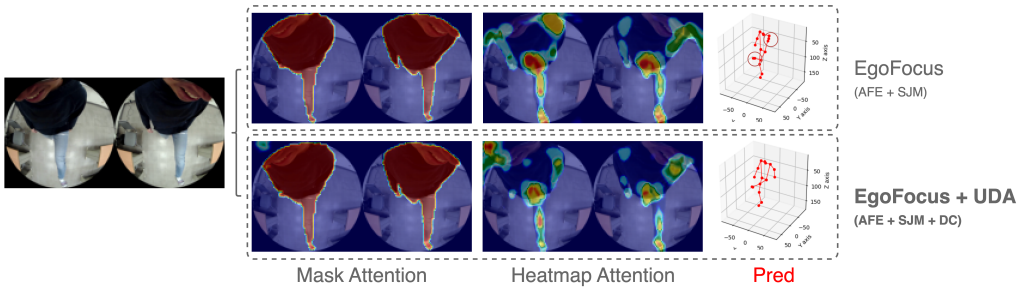
4.5 Ablation Study

We conduct ablation studies to investigate how different design choices impact model performance and capabilities.

doi:10.6342/NTU202504485

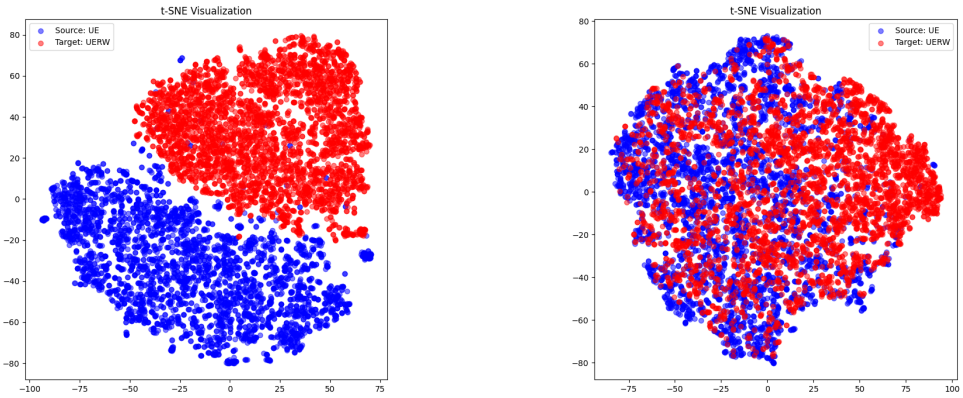


(a)



(b)

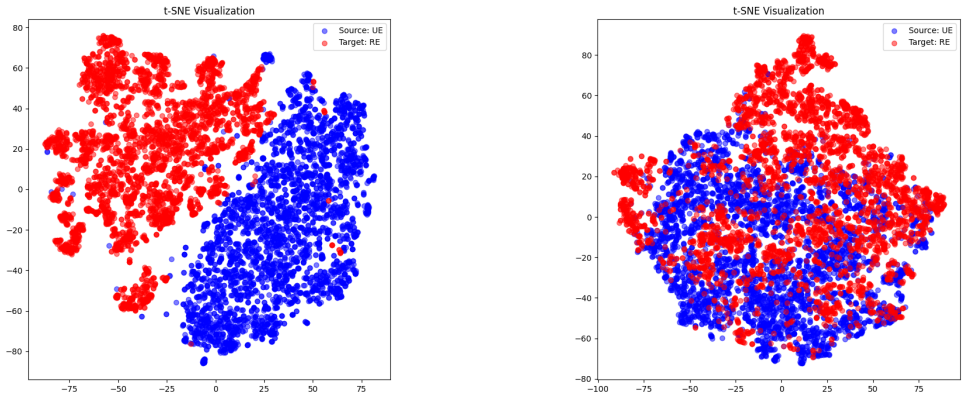
Figure 4.6: **Visualization of model predictions on the RealEgo test dataset.**



(a) Before UDA. (b) After UDA.

Figure 4.7: **Visualization of t-SNE on the UnrealEgo and UnrealEgo-RW test datasets.**

doi:10.6342/NTU202504485



(a) Before UDA. (b) After UDA.

Figure 4.8: **Visualization of t-SNE on the UnrealEgo and RealEgo test datasets.**

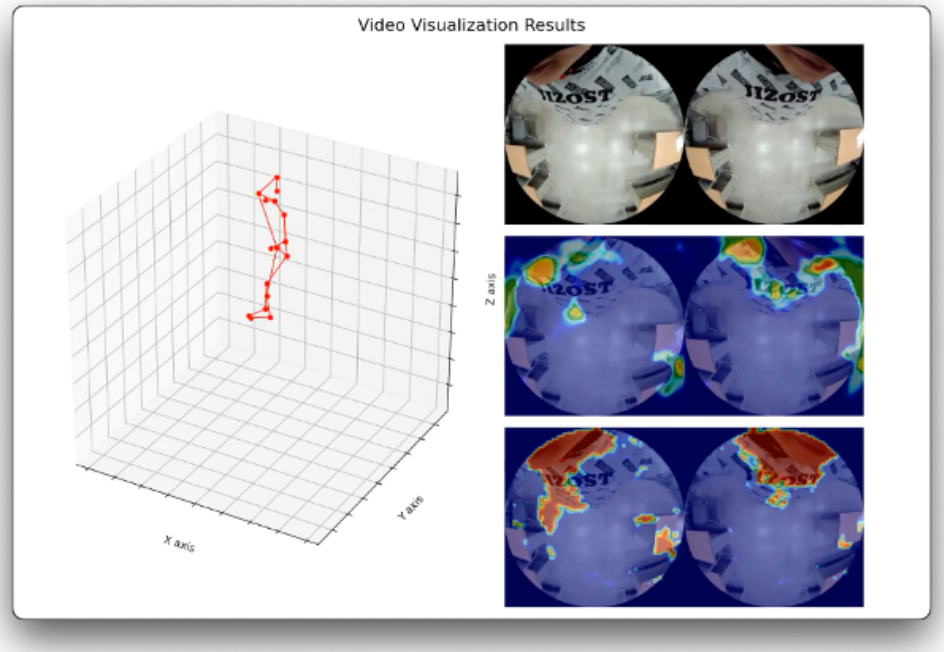


Figure 4.9: **Visualization of one of the UDA failure cases.**

4.5.1 Location Feature Types

We propose using both 2D heatmaps and human masks to enhance model performance. As shown in Table 4.4, using either feature alone provides some improve-

doi:10.6342/NTU202504485

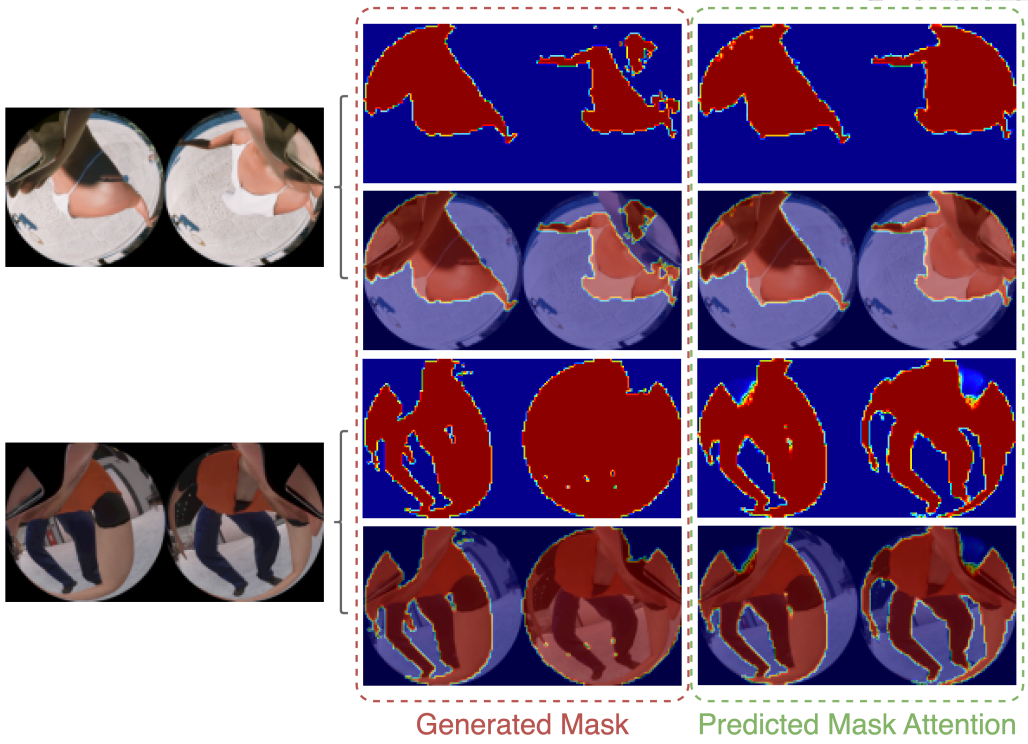


Figure 4.10: **Comparison of generated human masks and predicted mask attention maps.**

ment to the model. Specifically, the use of heatmaps results in greater performance gains, as they leverage precise 2D joint coordinates that provide more useful information. Combining both features across different channels leads to further improvements, showcasing the complementary benefits of these two types of location features.

4.5.2 Local Attention Loss

We compare our proposed local attention loss with the commonly used MSE loss in previous work like xR-EgoPose [2] and UnrealEgo [1]. Table 4.5 highlights the superior performance of local attention loss. Additionally, Figure 4.11 demonstrates that local attention loss results in clearer joint regions, indicating stronger constraints in heatmaps with small high-value regions, such as human joint heatmaps, compared to MSE loss.

doi:10.6342/NTU202504485

Table 4.4: **Ablation - Location feature types.** Note that experiments with multi-scale heatmaps were conducted with $\sigma = [1, 2, 3, 4, 5, 6]$.

+Mask	+Mshm	MPJPE (mm) ↓	PA-MPJPE (mm) ↓
		43.842	39.121
v		40.956	37.251
	v	38.610	35.444
v	v	**38.222**	**35.157**

Table 4.5: **Ablation - Local attention loss.** Note that all these experiments were conducted with $\sigma = [1, 2, 3, 4, 5, 6]$.

Loss Type	MPJPE (mm) ↓	PA-MPJPE (mm) ↓
MSE Loss	40.101	36.565
Local Attention Loss	**38.222**	**35.157**

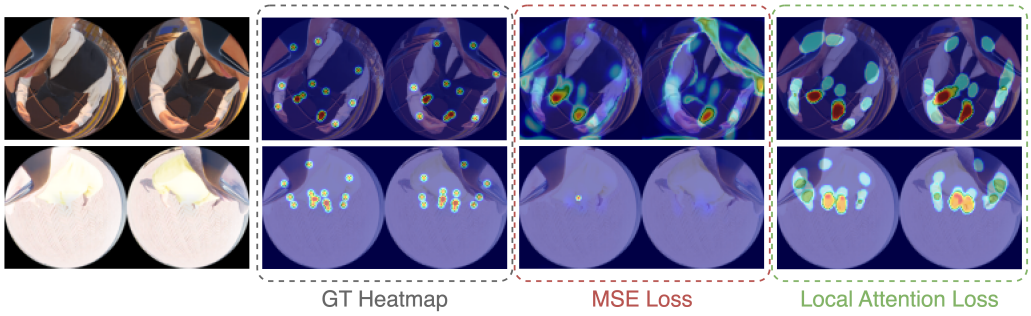


Figure 4.11: **The visualization of predicted heatmap attention maps with different loss types.**

4.5.3 Multi-Scale 2D Heamaps

We validate the benefits of using multi-scale heatmaps for attention. Table 4.6 reveals that smaller heatmaps enhance accuracy, while incorporating multiple scales achieves the best results. Furthermore, in Section 5.3, we will mention that multi-scale heatmaps training maintain accuracy in post-quantization, demonstrating better stability compared to single-scale heatmaps.

doi:10.6342/NTU202504485

Table 4.6: **Ablation study - Multi-scale 2D heatmaps.**

Scales (σ)	MPJPE (mm) ↓	PA-MPJPE (mm) ↓
$\sigma = [6, 6, 6, 6, 6, 6]$	38.477	35.383
$\sigma = [1, 2, 3, 4, 5, 6]$	38.222	35.157
$\sigma = [1, 1, 1, 1, 1, 1]$	37.885	34.949
$\sigma = [1, 1, 2, 2, 3, 3]$	**37.659**	**34.697**

4.5.4 Separation of Image Feature and Location Feature

We also evaluated the performance of our proposed local attention loss of multi-scale 2D heatmaps and human mask on both the Baseline Feature Extractor and the Attention-Guided Feature Extractor. As shown in Table 4.7, separating image features from location features and performing element-wise multiplication yields better results than directly using location features. This improvement highlights the complementary information between the two types of features and demonstrates their combined effectiveness in reducing errors.

Table 4.7: **Ablation - Separation of image feature and location feature.** Note that all these experiments were conducted with $\sigma = [1, 2, 3, 4, 5, 6]$.

Feature Extractor Type	MPJPE (mm) ↓	PA-MPJPE (mm) ↓
Baseline FE (w/ local attention loss)	39.340	35.765
Attention-Guided FE	**38.222**	**35.157**

4.5.5 UDA Loss Types

Next, we compared the effects of two types of losses in Unsupervised Domain Adaptation (UDA): domain loss and human prior loss. As shown in Table 4.8, both types of losses contribute positively to improving performance on unseen target domains, demonstrating their effectiveness in reducing the domain gap.

doi:10.6342/NTU202504485

Table 4.8: **Ablation - UDA loss types.**

+Domain Loss	+Prior Loss	MPJPE (mm) ↓	PA-MPJPE (mm) ↓
		201.426	129.452
v		173.714	107.439
	v	169.459	108.487
v	v	**158.693**	**100.359**

4.5.6 Computational Efficiency

Finally, we compare our model's size and computational cost with EgoPoseFormer [4], the most lightweight existing method. We found that the convolution block in the Stereo Joint Mixer (SJM) accounts for the majority of computational overhead. To verify that our model can achieve comparable performance with lower computational costs than EgoPoseFormer, we reduce the number of layers and channels in this convolution block. The results, presented in Table 4.9, demonstrate that our model can achieves a favorable balance between efficiency and accuracy.

Table 4.9: **Ablation - Computational efficiency.**

Method	PA-MPJPE (mm) ↓	Params (M)	GFLOPs
Ours (EgoFocus)	34.7	14.4	9.6
Ours (w/ lightweight SJM)	35.5	**13.5**	**5.9**
EgoPoseFormer [4]	**33.4**	14.1	7.3

doi:10.6342/NTU202504485

Chapter 5

System

In this chapter, we present the implementation and evaluation of our neural network deployment on the Xilinx ZCU104 FPGA platform. It covers the system architecture, implementation process, and performance evaluation. Section 5.1 provides an overview of the system, including the hardware platform, development tools, and operational modes. Section 5.2 details the implementation process, starting with the tools and frameworks, followed by the deployment workflow from model quantization to hardware deployment, and concluding with explanations of the Python demo code for dataset inference mode and camera streaming mode, including multi-threading optimizations. Section 5.3 evaluates system performance, including discussions on accuracy after quantization, overall system speed, and rendering results.

5.1 System Overview

We implemented our system on the Xilinx Zynq UltraScale+ MPSoC ZCU104 FPGA [70], which operates on the PYNQ [71, 72, 73] framework. The development process utilized Vitis-AI [74, 12] for model quantization and compilation onto the Xilinx Deep Learning Processor Unit (DPU) [13]. This approach offers the advantage of quickly deploying customized neural network architectures onto

doi:10.6342/NTU202504485

hardware, reducing unnecessary development time.

Our system supports two operational modes. The first is the dataset inference mode, as shown in Figure 5.1a, using the test dataset described in Section 4.1 as input. The second is the camera streaming mode, illustrated in Figure 5.1b, utilizing input frames captured by two fisheye cameras connected to the board.

In terms of application, the FPGA simulates resource-limited environments, such as HMD or AR glasses. To better understand its computational capabilities, we compared the performance of a single DPU with NVIDIA Jetson embedded edge AI platforms and Apple's on-chip NPU, as shown in Figure 5.2. The data indicates that high-end mobile chip NPUs significantly outperform the ZCU104 DPU in TOPS. Furthermore, since the Jetson Nano and TX2 GPUs only support floating-point computation, we estimate that the computational capability of a single ZCU104 DPU falls between these two devices. Building on this computational analysis, the FPGA is employed in a resource-limited environment where real-time human pose data is generated and transmitted to a PC or mobile device for further downstream processing. Given that the human pose data is lightweight, this approach minimizes bandwidth usage and facilitates efficient communication in edge computing scenarios.

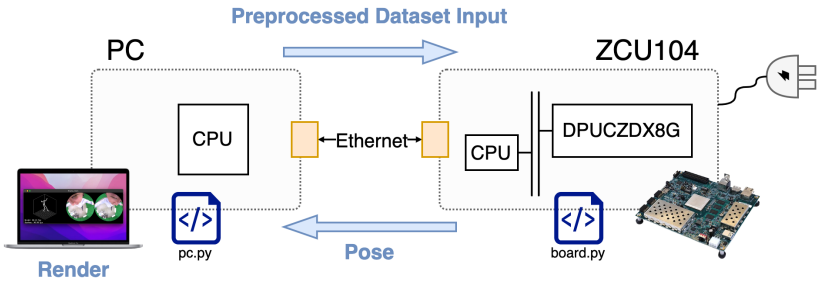
5.2 Implementation Details

5.2.1 Tools and Frameworks

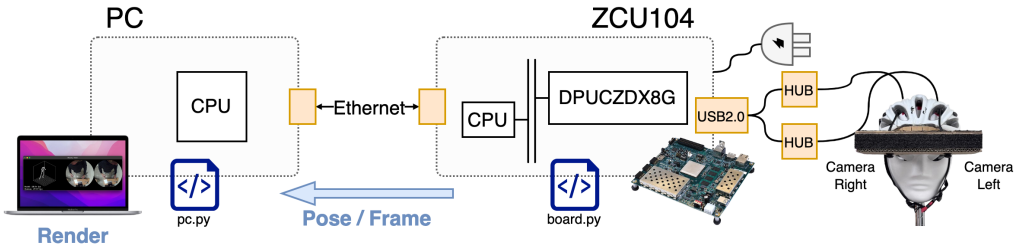
There are multiple approaches to deploying neural networks on FPGA platforms. One traditional method involves using Register Transfer Level (RTL) to design custom CNN accelerators and integrating them with other IPs through Vivado to generate a bitstream, which is then programmed onto the FPGA. However, this method requires extensive hardware design, making it both time- and labor-intensive.

In recent years, High-Level Synthesis (HLS) has gained popularity. HLS tools,

doi:10.6342/NTU202504485



(a) Dataset inference mode.



(b) Camera streaming mode.

Figure 5.1: **Two operational modes of our system.**

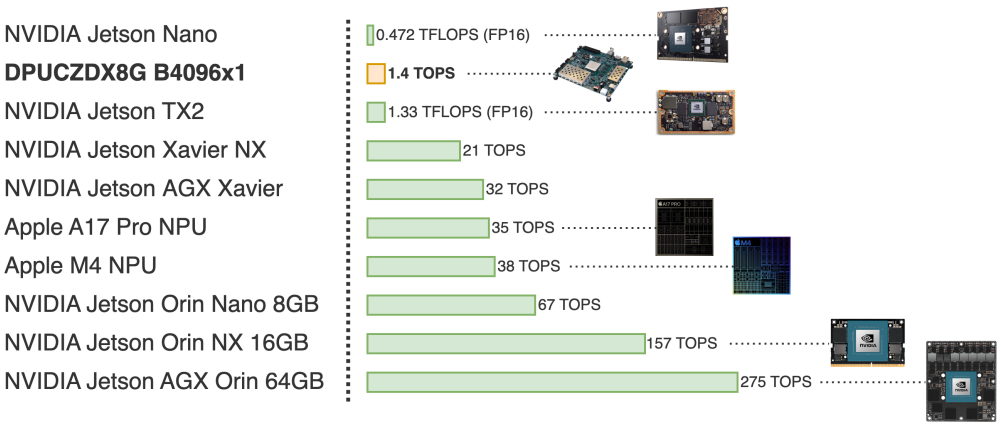


Figure 5.2: **Comparison of peak compute performance across edge devices [8, 9, 10, 11].**

such as Vitis HLS, enable developers to write designs in C/C++ and convert them into Vivado IPs. This approach significantly reduces development time while

doi:10.6342/NTU202504485

providing flexibility in resource allocation and scheduling for processing elements. HLS delivers performance comparable to RTL design but still requires a predefined model architecture and operator types before implementation. Therefore, it may not be ideal for simultaneous algorithm and hardware co-design.

An alternative framework is FINN [75, 76], which focuses on few-bit quantized neural networks. FINN supports CNNs, fully connected layers, and pooling layers, generating dataflow-style hardware architectures that vary from model to model. Its strengths include rapid development for specific models and a user-friendly interface based on Brevitas [77], a PyTorch library for Quantization-Aware Training (QAT) and Post-Training Quantization (PQT). However, FINN has limitations, such as supporting fewer operator types than Vitis-AI, lower bit-width quantization targeting ultra-lightweight models, and limited example use cases. Consequently, we did not select FINN for this project.

Instead, we adopted the Vitis-AI framework, as shown in Figure 5.3, which differs significantly from FINN. Rather than designing custom hardware architectures for each model, Vitis-AI employs a pre-built DPU that accelerates a wide range of operations. This highly integrated flow streamlines the process, enabling rapid deployment and seamless switching between different model architectures on the FPGA without regenerating bitstreams or reconfiguring the hardware platform, which benefits simultaneous software algorithm development and hardware integration. Given these advantages, we identified Vitis-AI flow as the most efficient and practical approach for our implementation.

The Xilinx Deep Learning Processor Unit (DPU) is a programmable engine specifically designed to accelerate convolutional neural networks on FPGA platforms. It features a dedicated instruction set optimized for deep learning workloads, enabling efficient execution of various CNN architectures such as VGG, ResNet, GoogLeNet, YOLO, SSD, MobileNet, and FPN. To deploy models on the DPU, executable xmodel files are generated using the Vitis-AI Compiler, which translates neural network computations into optimized instructions for the DPU. The DPU

doi:10.6342/NTU202504485

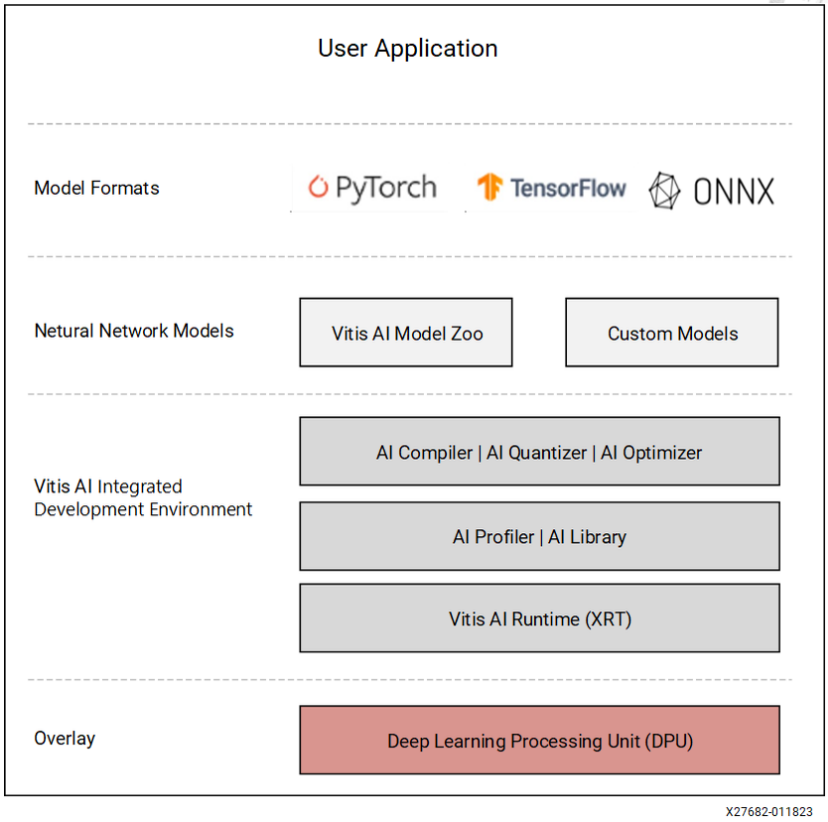
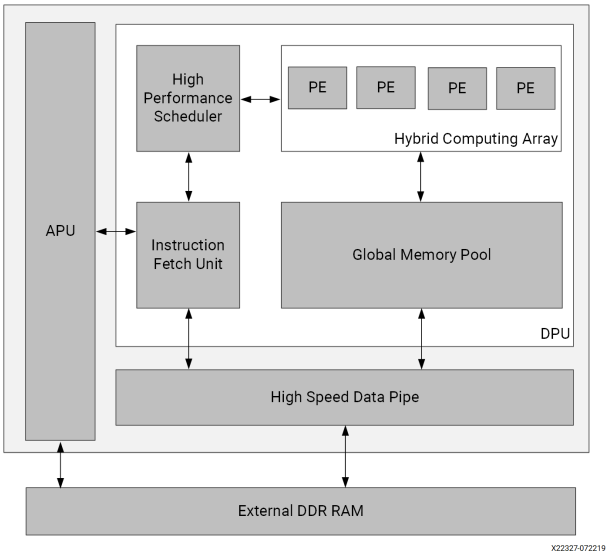


Figure 5.3: **Vitis-AI framework [12].**

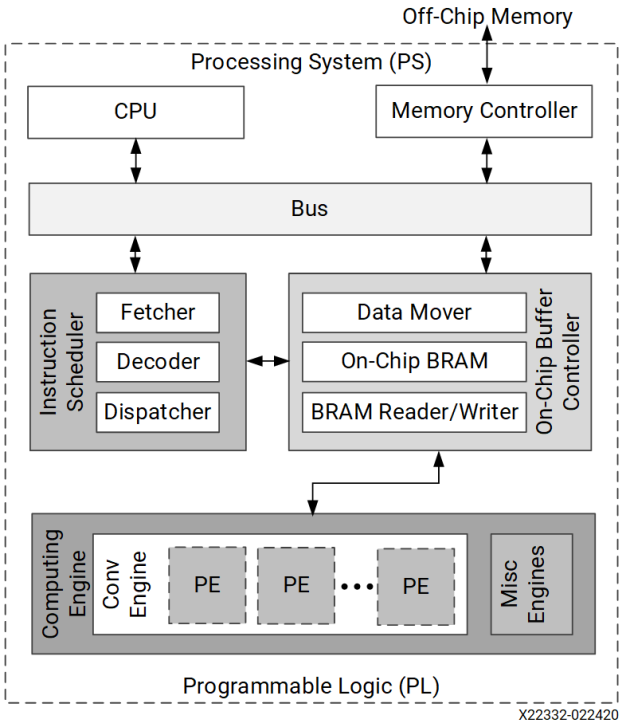
then executes these compiled instructions, efficiently handling key operations such as convolution, activation, and pooling. For our implementation, we used a single DPUCZDX8G with the B4096 configuration, as illustrated in Figure 5.4. The detailed hardware resource utilization for this setup is provided in Table 5.1.

The board operates on PYNQ 3.0.1, an open-source AMD project that integrates Python APIs and Jupyter notebooks, simplifying the use of hardware IPs for system-level applications. Running on a Linux environment, we installed the DPU-PYNQ package [78] to facilitate loading the DPU overlay. Note that the default DPU overlay includes two DPUCZDX8G cores, but we used only one to simulate a computationally constrained environment.

doi:10.6342/NTU202504485



(a) DPUCZDX8G top-level block diagram.



(b) DPUCZDX8G hardware architecture.

Figure 5.4: **Xilinx DPUCZDX8G overview [13].**

doi:10.6342/NTU202504485

Table 5.1: **Resource Utilization of single DPUCZDX8G on ZCU104.**

Resource	Used	Total	Utilization (%)
LUT	50194	228081	21.95%
REG	98142	457397	21.41%
BRAM	82	312	26.28%
URAM	46	96	47.92%
DSP	710	1728	41.09%

5.2.2 Deployment Workflow

The process before deployment is divided into two main stages: model quantization and compilation. Both stages were executed in the Vitis-AI 2.5 Docker environment.

The first stage, quantization, as shown in Figure 5.5, requires the model definition code, pre-trained floating-point weights, and a small subset of calibration data consisting of input images, for which we used a subset of the UnrealEgo test dataset, without the need for labels. The model definition only needs to include the forward function, and it is essential to verify that the model's operators are supported by the DPU [79]. Unsupported operations default to CPU execution. Vitis-AI provides an inspector tool to check whether the model is deployable. By default, we quantized both the weights and inputs to 8-bit integers to match the computational units of the DPU. The impact of quantization on model accuracy is analyzed in Section 5.3.

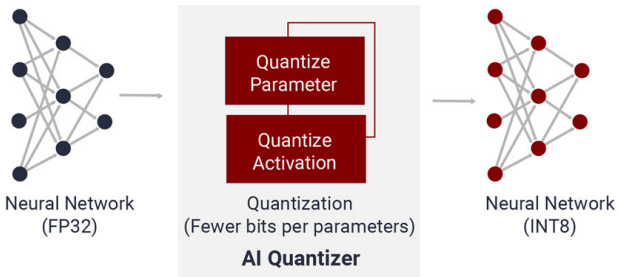


Figure 5.5: **Vitis-AI quantizer [12].**

doi:10.6342/NTU202504485

The second stage, compilation, as illustrated in Figure 5.6, uses the quantized xmodel file along with the corresponding DPU architecture configuration file, arch.json, which bridges the model to the specific DPU hardware. For this project, we used the default DPUCZDX8G/ZCU104/arch.json file, provided in the Vitis-AI Docker environment. The Vitis-AI Compiler offers the vai_c_xir API, enabling straightforward model compilation. After compilation, a deployable xmodel file is generated.

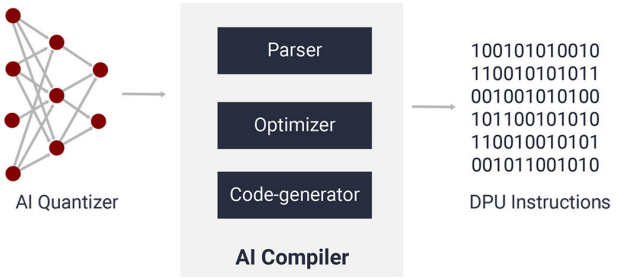


Figure 5.6: **Vitis-AI compiler [12].**

After loading the DPU overlay onto the board using the DPU-PYNQ Python API, the compiled xmodel can be executed directly through Vitis AI Runtime Python API for inference.

5.2.3 Demo Code Implementation

We implemented two demonstration modes: dataset inference mode and camera streaming mode. Both modes leverage threading and queues to optimize throughput and achieve real-time performance.

The dataset inference mode, illustrated in Figure 5.7a, employs a data thread on the PC to load preprocessed dataset inputs, which are pre-scaled and converted to int8 format to minimize bandwidth usage. The send thread then transmits these input images over Ethernet via sockets to the FPGA. Concurrently, receive and render thread process human pose results sent from the FPGA and render them alongside denormalized images on the PC screen. On the FPGA side, a receive

doi:10.6342/NTU202504485

thread collects input data, a runner thread performs inference, and a send thread transmits the human pose results back to the PC.

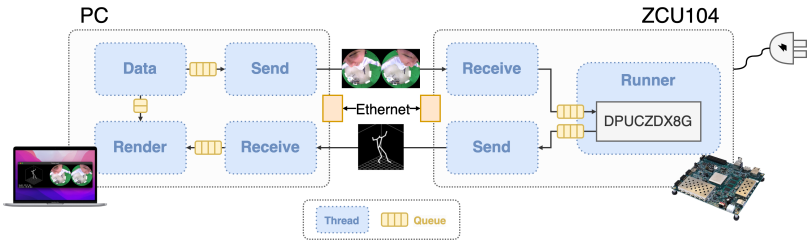
The camera streaming mode, illustrated in Figure 5.7b, differs by streaming frames directly from two fisheye cameras connected to the FPGA. The PC is responsible only for receiving human pose data and frames for rendering. On the FPGA, two camera threads use cv2.VideoCapture to fetch frames from UVC fisheye cameras. Frames (320x240) are cropped to 256x240 and centered within a 256x256 black background to preserve the field of view without requiring time-intensive resizing on board. Next, we use the downscaled UnrealEgo's circular mask to apply masking to the frames. The preprocess thread normalizes, scales, and converts data types before feeding it to the runner thread for inference. Two send threads then transmit the frames and pose data back to the PC. Due to the ZCU104 board's single USB port, a USB hub was used to connect both cameras. However, Linux systems often experience bandwidth allocation issues with multiple UVC cameras, preventing simultaneous access [80]. To address this, we modified and recompiled the Linux uvcvideo driver kernel to resolve the bandwidth limitation.

To evaluate computational performance, we implemented both CPU and DPU inference options within the runner thread. For CPU execution, we used PyTorch to run a floating-point model. For DPU execution, we used the int8 model compiled in Section 5.2.2. Different model architectures may generate varying numbers of DPU subgraphs. The baseline architecture contains a single DPU subgraph, while the improved architecture, or EgoFocus, includes six DPU subgraphs. The segmentation of subgraphs in the improved model was necessary due to transpose and reshape operations within the SJM module that had to be executed on the CPU. However, since the computational bottleneck lies within the backbone, this segmentation had minimal impact on overall inference speed.

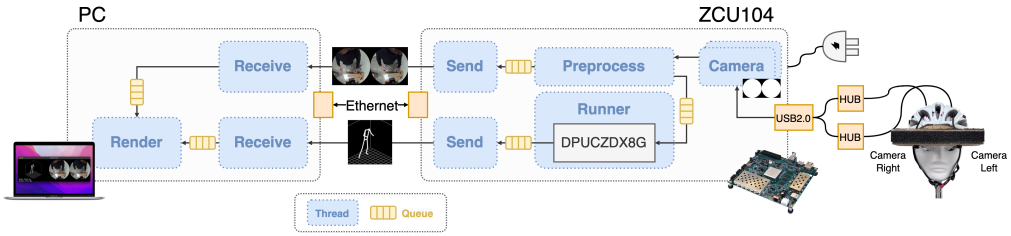
For rendering, we avoided the use of Matplotlib due to its inability to meet real-time requirements, which could slow the system. Instead, we implemented 3D pose rendering using OpenCV from scratch for higher performance. To improve

doi:10.6342/NTU202504485

temporal smoothness in our image-based approach, we applied a 1D Gaussian filter directly to the pose data sequence over time, mitigating jittering effects. We set the smoothing window size to 5 frames with a Gaussian sigma of 3.0, achieving visually smoother and more stable pose transitions.



(a) Dataset inference mode with multi-threading.



(b) Camera streaming mode with multi-threading.

Figure 5.7: **Two operational modes of our system with multi-threading details.**

5.3 Performance Evaluation

We first analyze the accuracy degradation after quantization using the UnrealEgo test set. As shown in Table 5.2, due to precision changes from floating-point to int8, we observe a degradation of approximately 10–15% in accuracy on EgoFocus. For the UDA configurations, the degradation ranges from 15% to 41%, with MPJPE of EgoFocus + UDA_{UERW} experiencing the highest drop. Although the quantized UDA models show greater accuracy degradation on the UnrealEgo synthetic dataset, this does not negatively impact their superior performance on real-world data, where

doi:10.6342/NTU202504485

they continue to exhibit improved generalization and robustness over the original EgoFocus model.

Notably, in Table 5.3, we also examine the impact of quantization on the four different experimental setups described in Section 4.5.3. The results indicate that the accuracy drop in the multi-scale heatmaps configuration is significantly lower than in the single-scale configuration, highlighting the importance of multi-scale heatmaps training in preserving accuracy after quantization.

Table 5.2: **Comparison of accuracy before and after quantization.**

Experiment	Stage	MPJPE (mm) ↓	PA-MPJPE (mm) ↓
EgoFocus	Float	37.659	34.697
	Quantized	43.092	38.176
EgoFocus + UDA$_{\text{UERW}}$	Float	38.867	35.391
	Quantized	54.753	40.942
EgoFocus + UDA$_{\text{RE}}$	Float	39.233	36.010
	Quantized	48.711	42.397

Table 5.3: **Comparison of MPJPE on multi-scale heatmaps training before and after quantization.**

Scales (σ)	MPJPE (mm) ↓	Quantized MPJPE (mm) ↓
$\sigma = [6, 6, 6, 6, 6, 6]$	38.477	48.606 (+26.4%)
$\sigma = [1, 2, 3, 4, 5, 6]$	38.222	43.366 (+13.5%)
$\sigma = [1, 1, 1, 1, 1, 1]$	37.885	46.204 (+21.9%)
$\sigma = [1, 1, 2, 2, 3, 3]$	37.659	43.092 (+14.4%)

Next, we evaluate the system's speed, as shown in Table 5.4. The results indicate that in dataset inference mode, the Xilinx DPU achieves a 2.4x speedup compared to an Intel Core i9-9820X CPU and a 126x speedup over the ARM Cortex-A53 CPU on the ZCU104 board, demonstrating real-time performance.

doi:10.6342/NTU202504485

Due to camera input processing and additional pre-processing overhead, the camera streaming mode runs slightly slower but still maintains real-time capability, making it feasible for practical application.

Table 5.4: **System speed.**

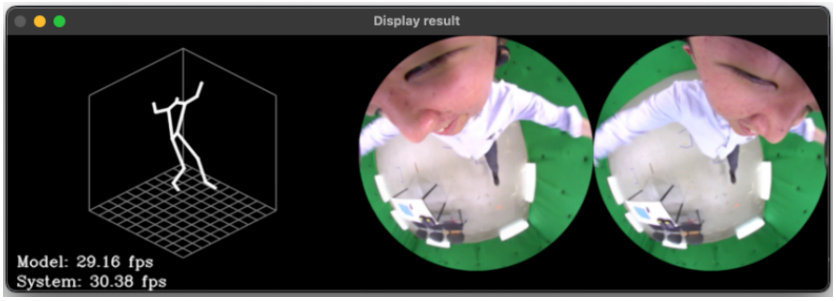
System Mode	Runner	FPS ↑
Dataset Inference	Intel Core i9-9820X CPU	12.61
	ARM Cortex-A53 CPU	0.19
	Xilinx DPU	**30.63**
Camera Streaming	ARM Cortex-A53 CPU	0.17
	Xilinx DPU	**24.12**

Finally, we provide our system device setup images and rendering results, as illustrated in Figure 5.8 and Figure 5.9.

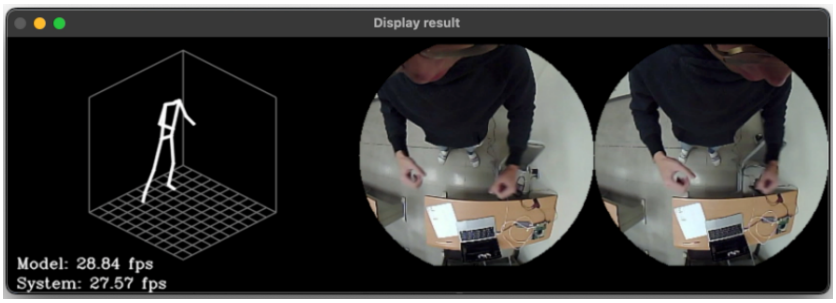
(a) Dataset inference mode device setup. (b) Camera streaming mode device setup.

Figure 5.8: **System device setup.**

doi:10.6342/NTU202504485



(a) Dataset inference mode rendering result on UnrealEgo-RW (EgoFocus + UDA_{UERW}).



(b) Camera streaming mode rendering result in real-world environment (EgoFocus + UDA_{RE}).

Figure 5.9: **System rendering result.**

doi:10.6342/NTU202504485

doi:10.6342/NTU202504485

Chapter 6

Conclusion

In this thesis, we have developed a camera-streaming egocentric 3D human pose estimation system implemented on the Xilinx Zynq UltraScale+ MPSoC ZCU104 FPGA board. Through model architecture design, learning strategies, and deployment considerations, we have successfully optimized the system to enhance its performance and efficiency. Specifically, the Attention-Guided Feature Extractor refines the model's ability to focus on human joints and regions by utilizing multi-scale heatmaps and mask-guided attention. Additionally, the Stereo Joint Mixer efficiently processes channel-wise and spatial-wise stereo features using a simple yet effective architecture. By integrating these two components, our model achieves a significant improvement, reducing MPJPE by 43.3% and PA-MPJPE by 37.4% compared to the baseline. Furthermore, to mitigate the domain gap between synthetic and real-world data, we introduce a domain-adaptive training framework. Finally, our system operates at a real-time speed of 24-30 FPS.

Despite these advancements, the system has several limitations. Due to hardware resource constraints, we adopt an image-based approach rather than a temporal one, which could have provided better handling of out-of-view and self-occlusion issues. Additionally, while our unsupervised domain adaptation strategy helps address the domain gap, the lack of ground truth labels limits its effectiveness, particularly in lower-body joint predictions. Future research could explore the

doi:10.6342/NTU202504485

use of external cameras to generate egocentric pseudo labels based on existing mature 3D human pose estimation techniques. Developing a more robust approach to enhance generalization and accuracy remains an essential direction for future work.

doi:10.6342/NTU202504485

Reference

[1] H. Akada, J. Wang, S. Shimada, M. Takahashi, C. Theobalt, and V. Golyanik, "Unrealego: A new dataset for robust egocentric 3d human motion capture," in *European Conference on Computer Vision*. Springer, 2022, pp. 1–17. xiii, 2, 3, 4, 5, 11, 12, 18, 19, 21, 27, 30, 32, 41

[2] D. Tome, P. Peluse, L. Agapito, and H. Badino, "xr-egopose: Egocentric 3d human pose from an hmd camera," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 7728–7738. xiii, 2, 3, 9, 10, 21, 28, 41

[3] H. Rhodin, C. Richardt, D. Casas, E. Insafutdinov, M. Shafiei, H.-P. Seidel, B. Schiele, and C. Theobalt, "Egocap: egocentric marker-less motion capture with two fisheye cameras," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 6, pp. 1–11, 2016. xiii, 3, 11, 29

[4] C. Yang, A. Tkach, S. Hampali, L. Zhang, E. J. Crowley, and C. Keskin, "Egoposeformer: A simple baseline for stereo egocentric 3d human pose estimation," in *European conference on computer vision*. Springer, 2024, pp. 401–417. xiii, 3, 4, 13, 19, 31, 32, 33, 44

[5] I. O. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A. Steiner, D. Keysers, J. Uszkoreit *et al.*, "Mlp-mixer: An all-mlp architecture for vision," *Advances in neural information processing systems*, vol. 34, pp. 24 261–24 272, 2021. xiii, 4, 5, 14, 17, 23

doi:10.6342/NTU202504485

[6] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. March, and V. Lempitsky, "Domain-adversarial training of neural networks," *Journal of machine learning research*, vol. 17, no. 59, pp. 1–35, 2016. xiii, 15, 17, 24

[7] H. Akada, J. Wang, V. Golyanik, and C. Theobalt, "3d human pose perception from egocentric stereo videos," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 767–776. xiii, 3, 12, 30, 32

[8] "Peak performance of dpuczdx8g." [Online]. Available: https://docs.amd.com/r/4.0-English/pg338-dpu/DPUCZDX8G-Peak-Performance xiv, 47

[9] "Peak performance of nvidia jetson platforms." [Online]. Available: https://developer.nvidia.com/embedded/jetson-modules xiv, 47

[10] "Peak performance of apple a17 pro." [Online]. Available: https://en.wikipedia.org/wiki/Apple_A17 xiv, 47

[11] "Peak performance of apple m4." [Online]. Available: https://en.wikipedia.org/wiki/Apple_M4 xiv, 47

[12] "Vitis-ai documentation." [Online]. Available: https://docs.amd.com/r/2.5-English/ug1414-vitis-ai/ xiv, 45, 49, 51, 52

[13] "Xilinx dpu documentation." [Online]. Available: https://docs.amd.com/r/4.0-English/pg338-dpu/ xiv, 45, 50

[14] D. Zhao, Z. Wei, J. Mahmud, and J.-M. Frahm, "Egoglass: Egocentric-view human pose estimation from an eyeglass frame," in *2021 International Conference on 3D Vision (3DV)*. IEEE, 2021, pp. 32–41. 3, 4, 11

[15] T. Kang, K. Lee, J. Zhang, and Y. Lee, "Ego3dpose: Capturing 3d cues from binocular egocentric views," in *SIGGRAPH Asia 2023 Conference Papers*, 2023, pp. 1–10. 3, 4, 11, 32

doi:10.6342/NTU202504485

[16] "Vicon motion capture system." [Online]. Available: https://www.vicon.com/ 3

[17] Y. Liu, J. Yang, X. Gu, Y. Chen, Y. Guo, and G.-Z. Yang, "Egofish3d: Egocentric 3d pose estimation from a fisheye camera via self-supervised learning," *IEEE Transactions on Multimedia*, vol. 25, pp. 8880–8891, 2023. 3, 10

[18] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from single depth images," in *CVPR 2011*. Ieee, 2011, pp. 1297–1304. 7

[19] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu, "Human3. 6m: Large scale datasets and predictive methods for 3d human sensing in natural environments," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 7, pp. 1325–1339, 2013. 7

[20] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele, "2d human pose estimation: New benchmark and state of the art analysis," in *Proceedings of the IEEE Conference on computer Vision and Pattern Recognition*, 2014, pp. 3686–3693. 7

[21] S. Li and A. B. Chan, "3d human pose estimation from monocular images with deep convolutional neural network," in *Computer Vision–ACCV 2014: 12th Asian Conference on Computer Vision, Singapore, Singapore, November 1-5, 2014, Revised Selected Papers, Part II 12*. Springer, 2015, pp. 332–347. 7, 8

[22] S. Park, J. Hwang, and N. Kwak, "3d human pose estimation using convolutional neural networks with 2d pose information," in *Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III 14*. Springer, 2016, pp. 156–169. 7, 8

doi:10.6342/NTU202504485

[23] G. Pavlakos, X. Zhou, K. G. Derpanis, and K. Daniilidis, "Coarse-to-fine volumetric prediction for single-image 3d human pose," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7025–7034. 7, 8

[24] D. Mehta, H. Rhodin, D. Casas, P. Fua, O. Sotnychenko, W. Xu, and C. Theobalt, "Monocular 3d human pose estimation in the wild using improved cnn supervision," in *2017 international conference on 3D vision (3DV)*. IEEE, 2017, pp. 506–516. 7, 8

[25] J. Martinez, R. Hossain, J. Romero, and J. J. Little, "A simple yet effective baseline for 3d human pose estimation," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2640–2649. 7, 8

[26] F. Moreno-Noguer, "3d human pose estimation from a single image via distance matrix regression," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2823–2832. 7, 8

[27] G. Varol, J. Romero, X. Martin, N. Mahmood, M. J. Black, I. Laptev, and C. Schmid, "Learning from synthetic humans," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 109–117. 8

[28] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black, "Smpl: A skinned multi-person linear model," in *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, 2023, pp. 851–866. 8

[29] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *European conference on computer vision*. Springer, 2016, pp. 483–499. 8

[30] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7291–7299. 8

doi:10.6342/NTU202504485

[31] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, "Convolutional pose machines," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2016, pp. 4724–4732. 8

[32] X. Zhou, Q. Huang, X. Sun, X. Xue, and Y. Wei, "Towards 3d human pose estimation in the wild: a weakly-supervised approach," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 398–407. 8, 15

[33] H.-Y. Tung, H.-W. Tung, E. Yumer, and K. Fragkiadaki, "Self-supervised learning of motion capture," *Advances in neural information processing systems*, vol. 30, 2017. 8

[34] H. Rhodin, M. Salzmann, and P. Fua, "Unsupervised geometry-aware representation for 3d human pose estimation," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 750–767. 8

[35] B. Wandt and B. Rosenhahn, "Repnet: Weakly supervised training of an adversarial reprojection network for 3d human pose estimation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 7782–7791. 8, 15

[36] I. Habibie, W. Xu, D. Mehta, G. Pons-Moll, and C. Theobalt, "In the wild human pose estimation using explicit 2d features and intermediate 3d representations," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 10 905–10 914. 8, 15

[37] C.-H. Chen, A. Tyagi, A. Agrawal, D. Drover, R. Mv, S. Stojanov, and J. M. Rehg, "Unsupervised 3d pose estimation with geometric self-supervision," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 5714–5724. 8, 15

[38] C. Zheng, S. Zhu, M. Mendieta, T. Yang, C. Chen, and Z. Ding, "3d human pose estimation with spatial and temporal transformers," in *Proceedings*

doi:10.6342/NTU202504485

of the IEEE/CVF international conference on computer vision, 2021, pp. 11 656–11 665. 8

[39] D. Shi, X. Wei, L. Li, Y. Ren, and W. Tan, "End-to-end multi-person pose estimation with transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 11 069–11 078. 8

[40] Y. Xu, J. Zhang, Q. Zhang, and D. Tao, "Vitpose: Simple vision transformer baselines for human pose estimation," *Advances in Neural Information Processing Systems*, vol. 35, pp. 38 571–38 584, 2022. 8

[41] Q. Zhao, C. Zheng, M. Liu, P. Wang, and C. Chen, "Poseformerv2: Exploring frequency domain for efficient and robust 3d human pose estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 8877–8886. 8

[42] Y. Xu, J. Zhang, Q. Zhang, and D. Tao, "Vitpose++: Vision transformer for generic body pose estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023. 8

[43] G. Rogez, J. S. Supancic, and D. Ramanan, "First-person pose recognition using egocentric workspaces," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 4325–4333. 8

[44] H. Yonemoto, K. Murasaki, T. Osawa, K. Sudo, J. Shimamura, and Y. Taniguchi, "Egocentric articulated pose tracking for action recognition," in *2015 14th IAPR International Conference on Machine Vision Applications (MVA)*. IEEE, 2015, pp. 98–101. 8

[45] T. Shiratori, H. S. Park, L. Sigal, Y. Sheikh, and J. K. Hodgins, "Motion capture from body-mounted cameras," in *ACM SIGGRAPH 2011 papers*, 2011, pp. 1–10. 9

doi:10.6342/NTU202504485

[46] H. Jiang and K. Grauman, "Seeing invisible poses: Estimating 3d body pose from egocentric video," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017, pp. 3501–3509. 9

[47] T. Von Marcard, B. Rosenhahn, M. J. Black, and G. Pons-Moll, "Sparse inertial poser: Automatic 3d human pose estimation from sparse imus," in *Computer graphics forum*, vol. 36, no. 2. Wiley Online Library, 2017, pp. 349–360. 9

[48] K. Ahuja, V. Shen, C. M. Fang, N. Riopelle, A. Kong, and C. Harrison, "Controllerpose: inside-out body capture with vr controller cameras," in *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, 2022, pp. 1–13. 9

[49] W. Xu, A. Chatterjee, M. Zollhoefer, H. Rhodin, P. Fua, H.-P. Seidel, and C. Theobalt, "Mo 2 cap 2: Real-time mobile 3d motion capture with a cap-mounted fisheye camera," *IEEE transactions on visualization and computer graphics*, vol. 25, no. 5, pp. 2093–2101, 2019. 9, 28

[50] D. Tome, T. Alldieck, P. Peluse, G. Pons-Moll, L. Agapito, H. Badino, and F. De la Torre, "Selfpose: 3d egocentric pose estimation from a headset mounted camera," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 6, pp. 6794–6806, 2020. 10

[51] J. Wang, L. Liu, W. Xu, K. Sarkar, and C. Theobalt, "Estimating egocentric 3d human pose in global space," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 11 500–11 509. 10

[52] J. Wang, L. Liu, W. Xu, K. Sarkar, D. Luvizon, and C. Theobalt, "Estimating egocentric 3d human pose in the wild with external weak supervision," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 13 157–13 166. 10

doi:10.6342/NTU202504485

[53] J. Wang, D. Luvizon, W. Xu, L. Liu, K. Sarkar, and C. Theobalt, "Scene-aware egocentric 3d human pose estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 13 031–13 040. 10

[54] J. Wang, Z. Cao, D. Luvizon, L. Liu, K. Sarkar, D. Tang, T. Beeler, and C. Theobalt, "Egocentric whole-body motion capture with fisheyevit and diffusion-based motion refinement," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 777–787. 10

[55] Y. Liu, J. Yang, X. Gu, Y. Guo, and G.-Z. Yang, "Egohmr: Egocentric human mesh recovery via hierarchical latent diffusion model," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 9807–9813. 10

[56] J. Park, K. Kaai, S. Hossain, N. Sumi, S. Rambhatla, and P. Fieguth, "Domain-guided spatio-temporal self-attention for egocentric 3d pose estimation," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, pp. 1837–1849. 10

[57] "Unrealengine." [Online]. Available: https://www.unrealengine.com/en-US 11, 27

[58] T. Kang and Y. Lee, "Attention-propagation network for egocentric heatmap to 3d pose lifting," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 842–851. 12, 32

[59] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European conference on computer vision*. Springer, 2020, pp. 213–229. 13

[60] Y.-W. Cha, T. Price, Z. Wei, X. Lu, N. Rewkowski, R. Chabra, Z. Qin, H. Kim, Z. Su, Y. Liu *et al.*, "Towards fully mobile 3d face, body, and environment

doi:10.6342/NTU202504485

capture using only head-worn cameras," *IEEE transactions on visualization and computer graphics*, vol. 24, no. 11, pp. 2993–3004, 2018. 13

[61] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, "Deep domain confusion: Maximizing for domain invariance," *arXiv preprint arXiv:1412.3474*, 2014. 15

[62] M. Long, Y. Cao, J. Wang, and M. Jordan, "Learning transferable features with deep adaptation networks," in *International conference on machine learning*. PMLR, 2015, pp. 97–105. 15

[63] B. Sun and K. Saenko, "Deep coral: Correlation alignment for deep domain adaptation," in *Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III 14*. Springer, 2016, pp. 443–450. 15

[64] W. Yang, W. Ouyang, X. Wang, J. Ren, H. Li, and X. Wang, "3d human pose estimation in the wild by adversarial learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 5255–5264. 15

[65] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125. 18

[66] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo *et al.*, "Segment anything," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 4015–4026. 21, 28, 38

[67] "Renderpeople." [Online]. Available: https://renderpeople.com/ 27

doi:10.6342/NTU202504485

[68] "Elp-usbfhd01m camera." [Online]. Available: https://www.webcamerausb.com/elp-full-hd-usb20-webcam-uvc-2mpixels-hd-cmos-ov2710-vga-120fps-720p-60fps-1080p-30fps-usb-camera-module-with-fixed-focus-lens-changeable-p-80.html 29, 30

[69] H. Cuevas-Velasquez, C. Hewitt, S. Aliakbarian, and T. Baltrušaitis, "Simpleego: Predicting probabilistic body pose from egocentric cameras," in *2024 International Conference on 3D Vision (3DV)*. IEEE, 2024, pp. 1446–1455. 32

[70] "Xilinx zynq ultrascale+ mpsoc zcu104." [Online]. Available: https://www.xilinx.com/products/boards-and-kits/zcu104.html 45

[71] "Pynq." [Online]. Available: https://www.pynq.io/ 45

[72] "Pynq github repository." [Online]. Available: https://github.com/Xilinx/PYNQ.git 45

[73] "Pynq documentation." [Online]. Available: https://pynq.readthedocs.io/en/latest 45

[74] "Vitis-ai github repository." [Online]. Available: https://github.com/Xilinx/Vitis-AI.git 45

[75] "Finn github repository." [Online]. Available: https://github.com/Xilinx/finn.git 48

[76] "Finn documentation." [Online]. Available: https://finn.readthedocs.io/en/latest/ 48

[77] "Brevitas github repository." [Online]. Available: https://github.com/Xilinx/brevitas.git 48

[78] "Dpu-pynq github repository." [Online]. Available: https://github.com/Xilinx/DPU-PYNQ.git 49

doi:10.6342/NTU202504485

[79] "Supported operators for the dpu." [Online]. Available: https://docs.amd.com/r/2.5-English/ug1414-vitis-ai/Currently-Supported-Operators 51

[80] "Multiple uvc cameras on linux." [Online]. Available: https://www.thegoodpenguin.co.uk/blog/multiple-uvc-cameras-on-linux/ 53

doi:10.6342/NTU202504485