ABSTRACT

Title of Dissertation: TACKLING DATA SCARCITY
IN HUMAN-CENTRIC VISION:
FROM UAV-BASED DETECTION
TO MULTIMODAL POSE UNDERSTANDING

Yi-Ting Shen
Doctor of Philosophy, 2026

Dissertation Directed by: Professor Shuvra S. Bhattacharyya
Department of Electrical and Computer Engineering, and
Institute for Advanced Computer Studies

Human-centric vision tasks, such as human detection and pose understanding, are fundamentally constrained by limited data availability and the high cost of annotation, posing significant challenges for data-driven learning approaches. These challenges are further amplified in UAV-based and multimodal scenarios. In UAV-based settings, large variations in viewpoint, scale, and imaging conditions demand substantially more data to learn robust and generalizable representations. In multimodal settings, annotation becomes more complex, requiring semantically consistent alignment across modalities (e.g., vision and language), which further limits scalability. This dissertation addresses these challenges through advances in dataset design, synthetic data utilization, and scalable multimodal annotation, supported by novel methods, benchmarks, and systematic analyses.

For UAV-based scenarios, we focus on aerial-view human detection. We first introduce Archangel, a hybrid benchmark that integrates real and synthetic imagery with metadata on camera position and

human pose, enabling fine-grained evaluation under diverse conditions. Our analysis reveals that naively incorporating synthetic data into training is often ineffective due to the domain gap between synthetic and real data. To address this, we propose Progressive Transformation Learning (PTL), which incrementally selects and transforms synthetic data to better align with the real domain based on measured domain discrepancies before incorporating it into training, leading to improved detection performance in data-scarce settings. We further develop SynPoseDiv, a synthetic pose diversification framework that combines a diffusion-based pose generator with a pose-guided image translation model, enhancing synthetic data diversity and improving performance.

For multimodal scenarios, we address the challenge of scalable annotation for composed pose retrieval (CPR). We propose AutoComPose, a framework based on multimodal large language models (LLMs) that enables automatic pose-transition annotation. To evaluate its effectiveness, we construct new CPR benchmarks, demonstrating improved retrieval performance while substantially reducing reliance on manual annotation.

In summary, this dissertation demonstrates that data scarcity in human-centric vision can be effectively mitigated through improved dataset curation, the effective use of synthetic data via progressive sim-to-real transformation, enhanced pose diversity, and automated multimodal annotation. These contributions advance scalable and data-efficient learning in complex visual environments.

TACKLING DATA SCARCITY IN HUMAN-CENTRIC VISION: FROM UAV-BASED DETECTION TO MULTIMODAL POSE UNDERSTANDING

by

Yi-Ting Shen

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2026

Advisory Committee:
Professor Shuvra S. Bhattacharyya, Chair/Advisor
Professor Joseph JaJa
Professor Dinesh Manocha
Dr. Heesung Kwon
Professor Abhinav Shrivastava, Dean's Representative

© Copyright by
Yi-Ting Shen
2026

Acknowledgments

I feel truly fortunate to have received continuous support and encouragement from so many people throughout my Ph.D. journey. Without their guidance and assistance, this dissertation would not have been possible.

First and foremost, I would like to express my sincere gratitude to my advisor, Prof. Shuvra S. Bhattacharyya, for giving me the opportunity to join the DSPCAD research group and for allowing me the freedom to pursue research topics that I am passionate about. I have learned tremendously from you, both professionally and personally. Your attention to detail and your kindness toward others have left a lasting impression on me, and I will carry these values forward in my future career.

In addition, I would ike to extend my deep appreciation to Dr. Heesung Kwon, Dr. Hyungtae Lee, and Dr. Sungmin Eum. It has been a great honor to collaborate with you over the past five years on the Archangel project. This dissertation reflects the collective contributions of our work, which would not have been possible without your support and collaboration. I will always remember our weekly meetings and the many long nights we spent working together toward submission deadlines.

I am also grateful to my dissertation committee members, Prof. Joseph JaJa, Prof. Dinesh Manocha, and Prof. Abhinav Shrivastava, for their time and valuable suggestions, which have significantly improved this dissertation.

I would further like to acknowledge Maj. Kirk Shoemaker, Dr. Frank Skirlo, and all members of DTRA, ARL, and related organizations who have been involved in the Archangel project. It has been an honor and a valuable experience to contribute to this effort.

I would also like to thank Dr. Eungjoo Lee, Dr. Yaesop Lee, Dr. Yan Zhang, Jinsub Yim, Doheon Lee, Joshua Steiner, Rohit Shete, and all my collogues at the University of Maryland and in the DSPCAD research group. Your support and friendship have made this journey both productive and

enjoyable.

Finally, I would like to express my deepest gratitude to my family for their constant support throughout my life. I especially thank my parents for their love and encouragement. Most importantly, I thank my wife, Chiao-Yi (Charlene) Wang, for her companionship for more than a decade. Sharing this journey, from our undergraduate studies to our master's degrees, and now our Ph.D.s, has been one of the most meaningful experiences of my life. I look forward to creating many more memories together the years to come.

The research underlying this dissertation was supported by the Defense Threat Reduction Agency (DTRA) and the DEVCOM Army Research Laboratory (ARL) under Grant No. W911NF2120076. This research was also sponsored in part by the Army Research Office and the Army Research Laboratory (ARL) under Grant No. W911NF2110258. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Office, the Army Research Laboratory (ARL), or the U.S. Government.

Table of Contents

Acknowledgements ii

Table of Contents iv

List of Tables vii

List of Figures ix

List of Abbreviations xii

Chapter 1: Introduction 1

Chapter 2: Archangel — A Hybrid UAV-based Human Detection Benchmark with Position and Pose Metadata 5

2.1 Introduction . . . 5
2.2 Related Work . . . 10
2.2.1 UAV-based Object Detection Datasets . . . 10
2.2.2 UAV-based Object Detection Methods . . . 12
2.3 The Archangel Dataset . . . 14
2.3.1 Archangel-Mannequin . . . 15
2.3.2 Archangel-Synthetic . . . 15
2.3.3 Archangel-Real . . . 17
2.3.4 Importance of Camera Parameters . . . 19
2.4 Experimental Setup . . . 19
2.4.1 Overview . . . 19
2.4.2 Datasets . . . 20
2.4.3 Evaluation . . . 21
2.4.4 Implementation Details . . . 22
2.5 Results . . . 22
2.5.1 Performance of Pre-trained Models . . . 22
2.5.2 Fine-tuning on a Real UAV-based Dataset . . . 23
2.5.3 Fine-tuning on Real and Synthetic Datasets . . . 24
2.6 Ablation Studies . . . 27
2.6.1 Effect of Synthetic Dataset Size . . . 27
2.6.2 Leave-One-Pose-Out Analysis . . . 29
2.6.3 Precise Model Diagnosis: Performance on the Altitude-Radius Grid . . . 30
2.7 Discussion and Future Directions . . . 31
2.7.1 Direct Extensions of This Study . . . 33
2.7.2 UAV-based Visual Representation Learning with Metadata . . . 34
2.7.3 UAV-based Synthetic Data Generation and Augmentation . . . 34
2.8 Conclusion . . . 35

Chapter 3: Progressive Transformation Learning for Leveraging Virtual Images in Training 36
3.1 Introduction . . . 36
3.2 Related Work . . . 39
3.2.1 Leveraging Virtual Images in Training . . . 39
3.2.2 Progressive Learning . . . 41
3.3 Method . . . 42
3.3.1 Measuring Domain Gap between Real and Virtual Images . . . 42
3.3.2 Progressive Transformation Learning . . . 44
3.4 Experiments . . . 47
3.4.1 Experimental Setup . . . 47
3.4.2 Properties of Progressive Learning . . . 48
3.4.3 Results on Low-Shot Learning . . . 50
3.4.4 Results on Cross-Domain Detection . . . 54
3.5 Discussion . . . 55
3.6 Additional Details . . . 56
3.6.1 Modeling Representation Space of a Detector . . . 56
3.6.2 Implementation Details . . . 58
3.6.3 Dataset Details . . . 60
3.6.4 Additional Analyses . . . 63

Chapter 4: Diversifying Human Pose in Synthetic Data for Aerial-view Human Detection 71
4.1 Introduction . . . 71
4.2 Related Work . . . 74
4.2.1 Synthesizing Human Appearances . . . 74
4.2.2 Pose Generation . . . 74
4.2.3 Generating Images of Novel Poses . . . 75
4.3 The SynPoseDiv Framework . . . 75
4.3.1 Pose Generator . . . 75
4.3.2 Target Pose Selection . . . 76
4.3.3 Image Translator . . . 78
4.3.4 Post-Processing: Noisy Image Filtering and Occlusion Insertion . . . 79
4.4 Implementation Details . . . 79
4.4.1 Measuring Pose Similarity . . . 79
4.4.2 Pose Generator . . . 80
4.4.3 Image Translator . . . 81
4.5 Experiments . . . 81
4.5.1 Experimental Setup . . . 81
4.5.2 Main Results . . . 83
4.5.3 Ablation Study on Post-Processing . . . 83
4.5.4 Scaling Behavior of Synthetic Data . . . 84
4.6 Conclusion . . . 85
4.7 Additional Details . . . 86
4.7.1 Progressive Pose Search . . . 86
4.7.2 Pose Transformation . . . 87
4.7.3 Image Generation . . . 89

4.7.4 Training Details . . . 91
4.7.5 Additional Experiments . . . 92
4.7.6 Pose-Diversified Synthetic Data Samples . . . 94

Chapter 5: AutoComPose — Automatic Generation of Pose Transition Descriptions for Composed Pose Retrieval Using Multimodal LLMs 96
5.1 Introduction . . . 96
5.2 Related Work . . . 100
5.2.1 Pose Retrieval . . . 100
5.2.2 Composed Image Retrieval . . . 101
5.2.3 Harnessing MLLMs for Vision Tasks . . . 102
5.3 Method . . . 103
5.3.1 Generating Pose Transition Descriptions . . . 103
5.3.2 Cyclic CPR Model Training . . . 105
5.4 Experiments . . . 107
5.4.1 Datasets and Experimental Setup . . . 107
5.4.2 Main Results . . . 110
5.4.3 Additional Analyses . . . 113
5.4.4 Qualitative Results . . . 115
5.5 Conclusion and Broader Impact . . . 115
5.6 Additional Details . . . 116
5.6.1 Additional Experiments . . . 116
5.6.2 Dataset Details . . . 117
5.6.3 MLLM Prompting Strategy . . . 121

Chapter 6: Conclusion and Future Work 126
6.1 Conclusion . . . 126
6.2 Broader Impact . . . 128
6.3 Limitations and Future Work . . . 128
6.3.1 Generalization to More Diverse Synthetic and Real-World Datasets . . . 128
6.3.2 A Deeper Understanding of Progressive Transformation Learning . . . 129
6.3.3 Controllable Composed Pose Retrieval . . . 130

Bibliography 132

List of Tables

2.1 Comparison of recent UAV-based datasets (1k = 1,000). 12
2.2 Model complexity of the YOLOv5 variants used in this chapter. 14
2.3 Comparison of the three sub-datasets that comprise Archangel. 14

3.1 Low-shot learning accuracy using 20 and 50 real images. AP@.5 and AP@[.5:.95] are reported in each bin. For PTL, the margin relative to the baseline accuracy is shown below the reported accuracy. The best accuracy for each setting is highlighted in bold. R and V denote real and virtual image sets, respectively. 52
3.2 PTL ablation experiments. Models are trained on the VisDrone dataset using the 50-shot setup. 52
3.3 Cross-domain detection accuracy under 3×3 cross-domain setups. For each setup, the datasets on the left and right of the arrow indicate training and test sets, respectively. PTL and baseline accuracies (without virtual images) are reported. The top row corresponds to same-domain setups and serves as reference performance for cross-domain results. All setups within each column use the same test dataset and low-shot regime. . 55
3.4 Training configurations used in the experiments. 60
3.5 Wall-clock training time (minutes) for the VisDrone 20-shot setting. Experiments are conducted using GeForce RTX 2080 Ti GPUs. 63
3.6 Comparison of distance metrics (VisDrone, 20-shot). 64
3.7 Comparison of transformation candidate selection strategies (VisDrone, 50-shot). . . . 64

4.1 Detection accuracy of the pose-diversified synthetic dataset ('p-div') and the original synthetic dataset ('orig'). 82
4.2 Effect of post-processing on image quality and detection accuracy (Vis-20). 82
4.3 Detection accuracy using Okutama-Action and ICG for training. 93
4.4 Impact of varying the number of final target poses in constructing the pose-diversified synthetic dataset. Detection accuracies ('acc') are reported under the same-domain Vis-20 setup. 94

5.1 Comparison of pose transition descriptions produced by human annotators and AutoComPose on FIXMYPOSE and PoseFixCPR datasets. The best and second-best results in each column are highlighted. 110
5.2 Comparison between the rule-based method and AutoComPose on PoseFixCPR, highlighting the advantages of AutoComPose in providing less constrained and more generalizable pose transition descriptions. 111
5.3 Evaluation on AIST-CPR using descriptions generated via AutoComPose. 112
5.4 Ablation study demonstrating the impact of data augmentation and cyclic loss guidance on retrieval accuracy. The (-) symbol denotes the removal of the corresponding component. SW and MI denote swapping and mirroring, respectively. 113
5.5 Impact of paraphrasing. The numbers 1, 3, and 5 indicate the number of paraphrased descriptions per pose transition. 114

5.6 Effects of paraphrasing and cyclic training on human-annotated and rule-based descriptions (CLIP-RN50 setting). 115
5.7 Additional ablation study on AutoComPose. Results are reported under the CLIP-RN50 setting. 117

List of Figures

2.1 Examples of images from *Archangel-Synthetic* (top), *Archangel-Mannequin* (middle), and *Archangel-Real* (bottom). Each image is annotated with UAV position [height, radius]: [20m, 20m] (left), [35m, 20m] (middle), and [50m, 20m] (right). Instances are labeled by pose: *stand* (green), *squat* or *kneel* (red), and *prone* (cyan). 7

2.2 Examples of the three pose categories in *Archangel*: *stand* (left), *squat* or *kneel* (middle), and *prone* (right). 8

2.3 Illustration of the data collection process for *Archangel*. For each collection session, multiple objects (real people, mannequins or virtual characters) on the ground were captured by a camera mounted on a UAV (real or simulated). Each objects assumed one of the three predefined poses (*stand*, *kneel*, or *prone*). The UAV circled the objects at predefined altitudes and radii. 14

2.4 Overview of the experimental workflow. Pre-trained YOLOv5 models of varying complexity were fine-tuned on different hybrid sets of UAV-based real and synthetic data, and evaluated on a held-out UAV-based dataset. 20

2.5 AP50 of pre-trained YOLOv5 models evaluated on *Arch-Mann-Eval* and *Arch-Real-Eval*. 23

2.6 AP50 of YOLOv5 models fine-tuned on different subsets of *Arch-Mann-FT37*. Each model was evaluated on *Arch-Mann-Eval* (top) and *Arch-Real-Eval* (bottom). 25

2.7 AP50 of YOLOv5 models fine-tuned on hybrid sets constructed from *Arch-Syn-FT* and different subsets of *Arch-Mann-FT37*. 25

2.8 Improvement in AP50 of YOLOv5 models after adding *Arch-Syn-FT* into the original fine-tuning datasets based on different subsets of *Arch-Mann-FT37*. 26

2.9 AP50 of YOLOv5 models fine-tuned on balanced hybrid sets constructed from *Arch-Syn-FT-B* and different subsets of *Arch-Mann-FT37*. 28

2.10 Improvement in AP50 of YOLOv5 models after adding *Arch-Syn-FT-B* into the original fine-tuning datasets based on different subsets of *Arch-Mann-FT37*. 28

2.11 AP50 comparison of *YOLOv5n6* fine-tuned on hybrid sets constructed from *Arch-Syn-FT*, *Arch-Syn-FT-B*, and different subsets of *Arch-Mann-FT37*. 29

2.12 AP50 of *YOLOv5n6* fine-tuned on balanced hybrid sets constructed from three leave-one-pose-out synthetic fine-tuning datasets (*Arch-Syn-FT-NoSt-B*, *Arch-Syn-FT-NoKn-B*, and *Arch-Syn-FT-NoPr-B*) and different subsets of *Arch-Mann-FT37*. 30

2.13 AP50 comparison across altitude-radius grids for *YOLOv5n6* fine-tuned on hybrid sets constructed from *Arch-Mann-FT2*, *Arch-Syn-FT*, and *Arch-Syn-FT-B*. 32

3.1 Overview of Progressive Transformation Learning. 37

3.2 Progressive Transformation Learning (PTL) pipeline. The red arrow indicates the processing flow of virtual images selected for inclusion in the training set. 42

3.3 Analysis of the use of virtual images during PTL. The top row shows the accumulated distribution of transformation candidates with respect to camera locations (rotation circle radius and altitude relative to the target human along x- and y-axes, respectively) for each PTL iteration. Darker bins indicate a higher number of selected virtual images. The bottom row (x-axis: domain gap; y-axis: number of virtual images) shows the distribution of virtual images across domain gap values, where the domain gap is measured by Eq. 3.4. Results are based on experiments using 50 real images from the VisDrone dataset for training. 49
3.4 Learning curves of AP@.5 and AP@[.5:.95] on the three datasets. 50
3.5 Sample Virtual2Real transformation results (VisDrone, 50-shot). Each set consists of three images: original virtual image (left), transformed image (middle), and transformed image with background (right). 54
3.6 Sample images from each real dataset. 61
3.7 Sample images from the *Archangel-Synthetic* dataset. Metadata for each image is shown on the right. 62
3.8 Accumulated distributions of transformation candidates with respect to camera locations in additional setups. The x- and y-axes denote rotation circle radius and altitude relative to the target human, respectively. Results are shown for all experimental setups except for the setup (VisDrone, 50-shot) presented in Figure 3.3. 66
3.9 Distributions of virtual images with respect to domain gap in additional setups. The x-axis represents the domain gap and the y-axis represents the corresponding number of virtual images. Results are shown for all experimental setups except for the setup (VisDrone, 50-shot) presented in Figure 3.3. 67
3.10 Sample Virtual2Real transformation results (VisDrone, 20-shot). Each set consists of three images: original virtual image (left), transformed image (middle), and transformed image with background (right). 68
3.11 Sample Virtual2Real transformation results (VisDrone, 50-shot). Each set consists of three images: original virtual image (left), transformed image (middle), and transformed image with background (right). 68
3.12 Sample Virtual2Real transformation results (Okutama-Action, 20-shot). Each set consists of three images: original virtual image (left), transformed image (middle), and transformed image with background (right). 69
3.13 Sample Virtual2Real transformation results (Okutama-Action, 50-shot). Each set consists of three images: original virtual image (left), transformed image (middle), and transformed image with background (right). 69
3.14 Sample Virtual2Real transformation results (ICG, 20-shot). Each set consists of three images: original virtual image (left), transformed image (middle), and transformed image with background (right). 70
3.15 Sample Virtual2Real transformation results (ICG, 50-shot). Each set consists of three images: original virtual image (left), transformed image (middle), and transformed image with background (right). 70

4.1 The SynPoseDiv framework generates a set of novel poses using a pose generator. For each synthetic image in the original dataset, a sequence of target poses is selected from the novel pose set through target pose selection. The image translator then generates images corresponding to each target pose in the sequence. A pose-diversified synthetic dataset is constructed by adding the generated images to the original synthetic dataset after post-processing. 72
4.2 Sample poses in $\mathcal{P}_n$. Poses are represented by 3D locations of 17 keypoints. 75
4.3 Sample generated images before applying post-processing are shown on the right. Images of the three poses from the original synthetic dataset are shown on the left. 76
4.4 Filtered noisy images. Each example shows a generated image (left), a target pose (middle), and an estimated pose (right). The pose distance d_{pose} is displayed beneath. . 78
4.5 Sample generated images after occlusion insertion. 78
4.6 Scaling behavior of the synthetic dataset under the Vis-20 setup. The scaling behavior of each dataset is compared by using randomly sampled subsets of 1,080 (1/16th), 4,320 (1/4th), and 17,280 (full) images. 84
4.7 Accuracy changes as PTL progresses under the Vis-20 setup. The scaling behavior is evaluated up to the 20$^{\text{th}}$ PTL iteration (four times longer than the standard PTL). 85
4.8 Overview of the image generation process. 90
4.9 Sample images from the pose-diversified synthetic dataset. The original synthetic dataset (i.e., Archangel-Synthetic) has only three poses: standing, prone, and squatting. 95

5.1 Training composed pose retrieval models requires a large and diverse set of human pose transition descriptions, demanding significant human annotation efforts. We propose AutoComPose, a framework that automatically generates pose transition descriptions using multimodal large language models (MLLMs), enhancing pose retrieval performance. 97
5.2 AutoComPose generates pose transition descriptions through a three-stage process: In Stage I, it extracts body part-based transition descriptions as an intermediate representation for human body movements to enhance expressiveness. In Stage II, it integrates these body part-based descriptions and diversifies them through paraphrasing. Finally, in Stage III, it generates descriptions for input pose pairs after applying swapping, mirroring, both transformations to further improve diversity. 100
5.3 During retrieval model training, AutoComPose incorporates a cycle constraint by ensuring that the composed features of input reference images (I_{ref}) and transition descriptions ($T_{ref \to tgt}$) can be transformed back to the reference image features along a cyclic path using the reverse transition descriptions ($T_{tgt \to ref}$), which are obtained during the generation step. 103
5.4 Example CPR results. Ranked retrieval results are shown for a given reference image (Ref) and a transitional pose description (green box) as the query. Red boxes indicate the correct target images. The top, middle, and bottom rows correspond to results from AIST-CPR, FIXMYPOSE, and PoseFixCPR, respectively. 116
5.5 Additional CPR results on FIXMYPOSE. Ranked retrieval results are shown for a given reference image (Ref) and a transitional pose description (green box) as the query. Red boxes indicate the correct target images. 118

5.6 Additional CPR results on PoseFixCPR. Ranked retrieval results are shown for a given reference image (Ref) and a transitional pose description (green box) as the query. Red boxes indicate the correct target images. 118
5.7 Additional CPR results on AIST-CPR. Ranked retrieval results are shown for a given reference image (Ref) and a transitional pose description (green box) as the query. Red boxes indicate the correct target images. 119
5.8 Example pose pairs from three datasets. The top, middle, and bottom rows correspond to AIST-CPR, FIXMYPOSE, and PoseFixCPR, respectively. 120
5.9 Word clouds generated from human-annotated texts (top) and AutoComPose-generated texts (bottom) across three datasets: FIXMYPOSE (left), PoseFixCPR (middle), and AIST-CPR (right). 120
5.10 Prompt used to remove environment-related descriptions in the FIXMYPOSE dataset. . 122
5.11 Sampled assets used for constructing PoseFixCPR. Highlighted assets (red boxes) were used to render the bottom-right image. 123
5.12 Sampled images from PoseFixCPR. 123
5.13 Prompt for generating body part-based pose transition descriptions. 124
5.14 Prompt for integrating and diversifying body part-based pose transition descriptions. . . 124
5.15 Modified Prompt-2 used in the Stage I ablation study. The main modification, which removes body part-specific queries, is highlighted in red. 125
5.16 Modified Prompt-3 used in the Stage I ablation study. The main modification, which removes body part-specific queries, is highlighted in red. 125

List of Abbreviations

ARL	DEVCOM Army Research Laboratory
ARO	Army Research Office
AP	Average Precision
AutoComPose	Automatic Generation of Composed Pose Descriptions
BBC	Batch-Based Contrastive Loss
CPR	Composed Pose Retrieval
CNN	Convolutional Neural Network
CIR	Composed Image Retrieval
CLIP	Contrastive Language-Image Pre-training
DTRA	Defense Threat Reduction Agency
D2	Dawn Dove
EO	Electro-Optical
FOV	Field of View
FPS	Frames per Second
FPN	Feature Pyramid Network
FID	Fréchet Inception Distance
GUI	Graphical User Interface
GCS	Ground Control Station
GPS	Global Positioning System
GPU	Graphics Processing Unit
GAN	Generative Adversarial Network
GDA	Gaussian Discriminant Analysis
GPT	Generative Pre-trained Transformer
HFOV	Horizontal Field of View
HDRI	High-Dynamic-Range Imaging
IR	Infrared
IRB	Institutional Review Board
IoU	Intersection over Union
JSON	JavaScript Object Notation

LLM	Large Language Model
LDA	Linear Discriminant Analysis
MLLM	Multimodal Large Language Model
ML	Machine Learning
MoCap	Motion Capture
MI	Mirroring
NVMe	Non-Volatile Memory Express
PTL	Progressive Transformation Learning
PETG	Polyethylene Terephthalate Glycol
PT-FT	Pre-training and Fine-tuning
ROS	Robot Operating System
RN	ResNet
SynPoseDiv	Synthetic Pose Diversification
SOTA	State-of-the-Art
SVM	Support Vector Machine
SUV	Sport Utility Vehicle
SSD	Solid-State Drive
SMPL	Skinned Multi-Person Linear Model
SMPL-X	SMPL Expressive
SW	Swapping
TB	Terabyte
Tgt	Target
UAV	Unmanned Aerial Vehicle
VLM	Vision-Language Model
ViT	Vision Transformer

Chapter 1: Introduction

Human-centric vision tasks, such as human detection [1], pose estimation [2], and pose retrieval [3], are central to a wide range of applications, including surveillance [4], search and rescue [5], sports analysis [6], and rehabilitation [7]. In recent years, data-driven approaches, particularly deep learning methods, have driven substantial progress in these domains through increasingly powerful models and the availability of large-scale datasets [8]. However, these methods remain heavily dependent on access to abundant, high-quality annotated data. In many real-world scenarios, collecting such data is challenging, expensive, and often not scalable.

Unmanned aerial vehicle (UAV)-based vision [9, 10, 11] scenarios present a particularly demanding case. Compared with conventional ground-based imagery, UAV imagery exhibits much larger variation in viewpoint, scale, and image quality [12]. As a result, human appearance can vary drastically depending on the UAV's position and perspective. Developing robust and generalizable models under these conditions requires datasets that not only capture broad visual diversity but also provide metadata describing the relationship between scene content and imaging conditions. However, existing UAV-based datasets often lack such detailed metadata, limiting both model development and systematic analysis. This limitation motivates the exploration of alternative data sources that can provide richer control and annotation.

Synthetic data has emerged as a promising solution [13, 14, 15, 16]. Modern rendering tools [17, 18] enable scalable data generation with minimal annotation cost by providing precise control over factors such as viewpoint, human appearance, pose, and scene composition. However, naively incorporating synthetic data into training pipelines is often insufficient due to the visual discrepancy between synthetic and real imagery [19, 20]. Accurately measuring this discrepancy, commonly referred to as the domain gap, and effectively aligning synthetic and real data through sim-to-real transformation

become key challenges in fully exploiting synthetic data for model training.

Beyond domain alignment, diversity is equally critical in constructing effective synthetic datasets. In human-centric vision, human pose is a primary source of appearance variation [21]. Poses are highly diverse, structurally complex, and infeasible to enumerate exhaustively, especially in aerial-view scenarios. Nevertheless, existing synthetic data pipelines typically rely on limited, closed-source, and handcrafted character and motion assets, restricting scalability and coverage of plausible pose variations. Enhancing pose diversity in a scalable manner is therefore essential for improving the performance of models trained with synthetic data, particularly under limited real-data conditions.

A related challenge arises in multimodal human-centric tasks [22], where learning depends on semantically aligned annotations across modalities such as vision and language. For instance, in composed pose retrieval (CPR) [23], models must jointly interpret visual inputs and natural language descriptions of pose transitions. Generating such annotations requires fine-grained descriptions of articulated body movements, which are expensive to produce and difficult to scale manually. As multimodal learning becomes increasingly important, scalable and reliable alternatives to manual annotation are needed.

The central goal of this dissertation is to tackle data scarcity in human-centric vision through improved dataset construction, effective utilization of synthetic data, and automated multimodal annotation. Specifically, the work first introduces a new benchmark for UAV-based human detection that combines real and synthetic imagery with rich metadata. It then explores how progressive sim-to-real transformation enables more effective use of synthetic data for detector training, and how synthetic datasets can be further enhanced by increasing pose diversity through generative modeling. Finally, it addresses scalable annotation in multimodal learning by proposing an automated framework for generating pose transition descriptions for CPR.

The remainder of this dissertation is organized as follows. Chapter 2 introduces **Archangel** [24], a hybrid UAV-based human detection benchmark that integrates real, mannequin, and synthetic imagery with aligned imaging conditions. It presents the dataset design, metadata structure, and extensive analyses using UAV-oriented object detectors [25], with particular emphasis on the joint use of real and synthetic data.

Chapter 3 proposes **Progressive Transformation Learning (PTL)** [26], a framework that incrementally selects and transforms synthetic data to better align with the real domain based on measured domain discrepancies before incorporating it into training. By iterating over transformation candidate selection, virtual-to-real transformation, set updating, and detector training, this chapter demonstrates how progressive learning enables more effective use of synthetic data in both in-domain and cross-domain settings.

Chapter 4 further studies the role of pose diversity in synthetic datasets. It presents **SynPose-Div** [27], a framework that diversifies poses in an existing synthetic dataset by integrating a diffusion-based pose generator, a target pose selection strategy, and a pose-guided image translation process. The results show that increased pose diversity improves both detection performance and scalability across multiple benchmarks and training regimes.

Chapter 5 shifts focus to multimodal human pose understanding and investigates the challenge of scalable annotation for CPR. It introduces **AutoComPose** [28], a three-stage framework based on multimodal large language models (MLLMs) for automatically generating structured, diverse, and expressive pose transition descriptions. A cyclic training strategy is further employed to improve the robustness. Experimental results demonstrate that automated annotations can reduce reliance on manual labeling while achieving superior retrieval performance on newly collected CPR benchmarks.

Finally, Chapter 6 concludes the dissertation by summarizing the key contributions, discussing

the broader impact of this work on human-centric vision, and outlining promising directions for future research.

In summary, this dissertation contributes to both UAV-based human detection and multimodal pose retrieval. It demonstrates that data scarcity in human-centric vision can be effectively mitigated through improved dataset curation, effective synthetic data generation and utilization, and automated multimodal annotation. These contributions advance the development of scalable learning frameworks for human understanding in complex visual environments.

Chapter 2: Archangel — A Hybrid UAV-based Human Detection Benchmark with Position and Pose Metadata

2.1 Introduction

With the recent rapid advancement in edge computing technology coupled with resource-limited mobile platforms, particularly unmanned aerial vehicles (UAVs) with electro-optical (EO) sensor payloads, a wide range of UAV-enabled applications have been more prevalent. Notable examples include UAV-enabled search and rescue in disaster management [29], aerial surveillance and reconnaissance for civilian and military purposes [30], precision agriculture [31], traffic analysis [32], and intelligent transportation applications [33]. Lately, owing to the remarkable progress of artificial intelligence and machine learning technology, tailored to distinct constraints of small UAV platforms, these UAV-based applications have been frequently providing promising solutions and successfully achieving the goals and operational requirements of their corresponding applications. Central to the above UAV-based applications are streamlined plug-ins that can effectively interrogate real-time imagery, captured with UAVs, and provide image/video analytics relevant to UAV-based scene understanding, particularly object detection and recognition.

Recently, extensive efforts in object detection and recognition have led to extraordinary advances in the perception accuracy in various challenges associated with large-scale object detection benchmarks that were captured primarily with ground-based cameras [34, 35, 36]. Compared to ground-based object detection and recognition, UAV-based object detection poses unique and severe challenges, as UAV flight inevitably results in a wider range of variations in the conditions for capturing images, including the altitudes and viewing angles of cameras, system turbulence, and weather events. These variations lead to more drastic variations in object appearances/attributes, and thus pose addi-

tional challenges to onboard detection models in the location and recognition of objects of interest. In general, changes in object appearances/attributes, caused by varying the image collection conditions, entail three major dependencies: (1) pose dependency caused by changes in the UAV position or camera viewing angle, (2) scale dependency owing to the distance between the UAV and object, and (3) image quality dependency due to UAV turbulence or various weather conditions.

We argue that developing object detection models which can adequately learn features invariant to these dependencies is key to substantially enhancing object detection accuracy in UAV-based perception. This requires curating UAV-based datasets that include images and metadata that carefully depict the full spectrum of correlations between the target scene on the ground and the camera on the UAV, whose imaging conditions constantly change as the UAV navigates the entire range of given operational requirements. Therefore, it is imperative to have a UAV-based object detection benchmark carefully curated with object poses, UAV positions, and weather information in the form of metadata for accurate model validation and verification.

Existing UAV-based benchmarks, such as VisDrone [9], UAVDT [37], Okutama-Action [10], and Standford Drone Dataset [38], provide limited metadata. Despite containing a wide variety of scenes captured using UAVs under different circumstances with various types of objects of interest, they do not provide a complete set of metadata, such as object poses and UAV positions, for each image in the datasets. This significant lack of information about how objects on the ground are projected through the camera lens as a function of UAV positions can lead to considerable limitations in learning about the objects in UAV-based images, as the appearances/attributes of the objects are subject to large variations.

To overcome the limitation resulting from lack of metadata, we introduce a large-scale UAV-based dataset, called *Archangel*, collected with comprehensive position and pose information by the

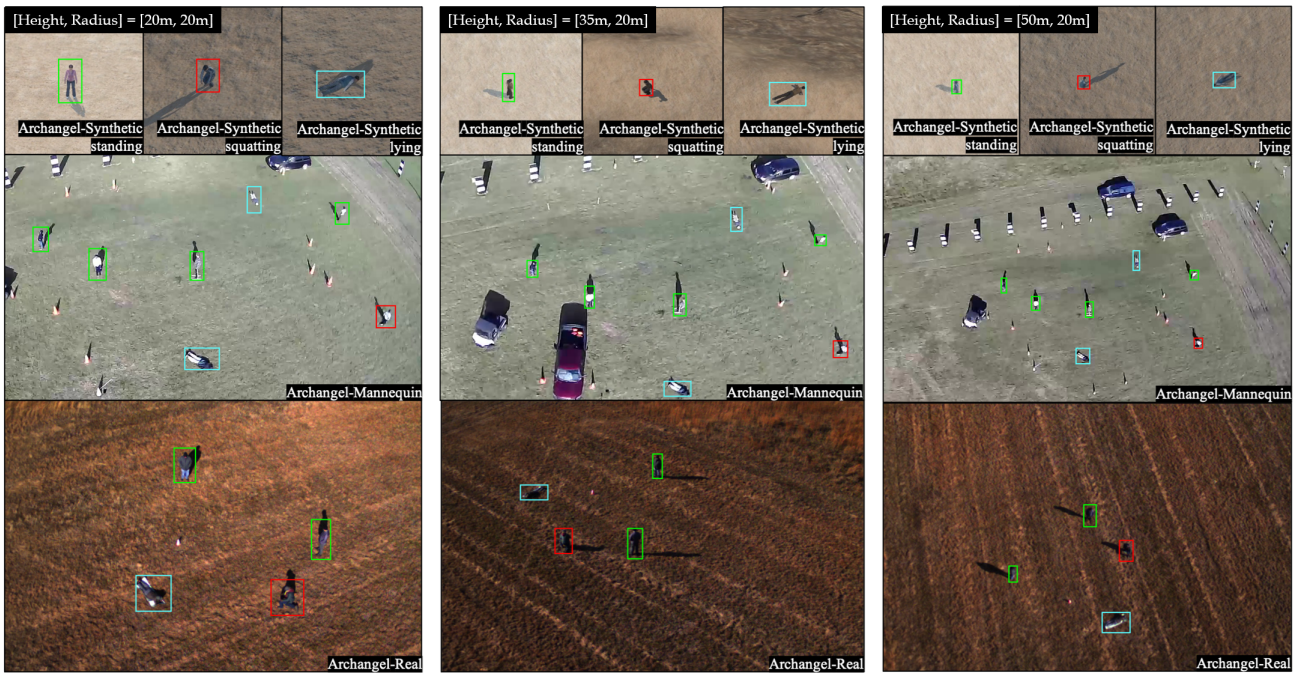


Figure 2.1: Examples of images from *Archangel-Synthetic* (top), *Archangel-Mannequin* (middle), and *Archangel-Real* (bottom). Each image is annotated with UAV position [height, radius]: [20m, 20m] (left), [35m, 20m] (middle), and [50m, 20m] (right). Instances are labeled by pose: *stand* (green), *squat* or *kneel* (red), and *prone* (cyan).

DEVCOM Army Research Laboratory (ARL) (Fig. 2.1). *Archangel* comprises three sub-datasets: *Archangel-Real* [39], *Archangel-Mannequin* [39] and *Archangel-Synthetic* [40]. *Archangel-Real* comprises video sequences captured by a UAV flying at various altitudes and radii of rotation circles. It sets a group of real humans as targets, and each human is in one of three possible poses (i.e., *stand*, *kneel* or *squat*, and *prone*) (Fig. 2.2). The terms *kneel* and *squat* are used interchangeably throughout this chapter. Similarly, *Archangel-Mannequin* sets a group of mannequins and different types of vehicles as targets. The imaging conditions for these two sub-datasets, such as UAV altitudes, ranges to targets, and object poses, are the same. Unlike *Archangel-Real* and *Archangel-Mannequin*, which were collected in real-world environments, *Archangel-Synthetic* was generated using the Unity game engine [18]. It includes a number of different virtual characters, who are in the same poses as described above and rendered with diverse illumination conditions. *Archangel-Synthetic* is designed for augmenting



Figure 2.2: Examples of the three pose categories in *Archangel*: *stand* (left), *squat* or *kneel* (middle), and *prone* (right).

the other two real sub-datasets and for studying various issues tied to optimizing machine learning (ML) models using synthetic data, such as synthetic data augmentation and domain adaptation [41], with respect to UAV-based scene understanding.

In addition to collecting a new dataset, we further characterize *Archangel* using state-of-the-art (SOTA) object detection models, specifically the YOLOv5 family [25] which has five different levels of architectural complexity and is pre-trained on MS-COCO [36], a large-scale ground-based object detection dataset. In this chapter, we focus on the YOLOv5 models with lower complexity (i.e., $YOLOv5n6$, $YOLOv5s6$, $YOLOv5m6$) since they are able to run on computing resource typically available on small UAV platforms (Table 2.2). For each model, we evaluate its human detection performance using *Archangel-Real*. Since we programmed our UAVs to circle around the human targets at various altitudes and radii during data collection (Fig. 2.3), the detection accuracy can be compared across the whole range of UAV positions.

Furthermore, we optimize the pre-trained YOLOv5 models with different fine-tuning strategies using various hybrid combinations of subsets from *Archangel-Mannequin* and *Archangel-Synthetic*. These fine-tuning strategies have been designed to provide valuable guidelines for leveraging synthetic data in training ML models to boost their performance. A comprehensive performance comparison between the baseline and the optimized YOLOv5 models is presented to demonstrate how incorporating a combination of real and synthetic data can enhance detection performance across varying UAV positions (Sec. 2.5 and 2.6). One of the critical findings from the comparative performance analysis indicates that if the real and synthetic data used for fine-tuning is balanced with respect to the amount of data, a significant performance boost can be achieved even with a low-complexity model, such as $YOLOv5n6$. Furthermore, the optimization based on the real and synthetic data is much more effective on infrequent object poses that are rarely seen in the original dataset for pre-training. For example, *prone* from *Archangel* is not often seen in MS-COCO [36], so the performance improvement on this pose is more evident.

Material in this chapter was published in [24]. It was extended from our previous studies [39, 40], where we mainly focused on introducing new datasets separately without extensive data analysis and characterization. Specifically, the major scope of this chapter is to extensively study the three sub-datasets jointly as a unified UAV-based benchmark with metadata for human detection. In summary, the contributions of this chapter are as follows:

1. We present a unified *Archangel* dataset after substantially restructuring the three sub-datasets since our previous publications [39, 40], including additional labeling *Archangel-Real* and an expansion of the range of *Archangel-Synthetic*. To the best of our knowledge, *Archangel* is the first UAV-based object detection dataset which contains real and synthetic sub-datasets captured

with similar imaging conditions and includes an extensive set of metadata (e.g., object poses and UAV positions).

2. We conduct extensive data analysis on *Archangel* by jointly analyzing its three sub-datasets (*Archangel-Synthetic*, *Archangel-Mannequin* and *Archangel-Real*). In particular, we provide several important guidelines on exploiting real and synthetic data together to improve UAV-based object detectors.

2.2 Related Work

2.2.1 UAV-based Object Detection Datasets

There is an increasing number of large-scale benchmarks for object detection, utilizing images captured with some fixed or moving cameras on the ground [1, 35, 36, 42, 43, 44, 45], yet relatively few datasets have been collected with UAVs. Moreover, these UAV-based object detection datasets all have their own limitations. For instance, VisDrone [9] is one of the major datasets for UAV-based object detection. It consists of images captured with UAVs in dozens of different scenarios, in which ten categories of objects were selected and carefully labelled. While it does have an advantage in data diversity, VisDrone does not provide any metadata, such as UAV positions. In contrast to VisDrone, UAVDT [37], another well-known benchmark for UAV-based detection and tracking, provides three different kinds of UAV-specific metadata (i.e., weather condition, flying altitude and viewing angle) and a few object attributes such as vehicle category. However, the annotations for the metadata are coarse (i.e., 3 categories for each). Also, the dataset does not contain the human category. Unlike UAVDT, Okutama-Action [10] is composed of images from aerial views and contains humans

in different human poses. However, it provides limited metadata regarding UAV altitudes and camera viewing angles. A212-Haze [46], which is the first real haze and object detection dataset with in-situ smoke measurements aligned to aerial imagery and is used in the recent UG^2+ Challenge [47], provides UAV position metadata. However, A212-Haze does not include a synthetic subset, limiting its usage for facilitating studies on how to improve UAV-based object detectors by using synthetic data. More recently, DGTA [48] generates synthetic datasets associated with existing UAV-based object detection datasets and provides UAV position metadata for the generated datasets. Nevertheless, the existing UAV-based datasets they use, such as VisDrone [9], are still lack of metadata, limiting DGTA's usage for precise model diagnosis.

There are two other types of datasets which are closely related to UAV-based object detection. First, datasets such as DOTA [49] include aerial images collected from satellites or aircraft. Although they are usually curated for remote sensing applications, detecting objects in such datasets also suffers from severe variations in the scale and orientation of the objects. Second, some UAV-based datasets are designed for certain vision tasks strongly associated with object detection. For instance, CARPK [50] is a large-scale car parking lot dataset designed for object counting. MOR-UAV [51] is a large-scale moving object recognition dataset comprising videos captured by a UAV in various environments, such as urban areas and highways. Stanford Drone Dataset [38] is used for analyzing various object trajectories in the real world from the top-view. UAV123 [52] is used for low altitude UAV-based object tracking. Similar to *Archangel*, UAV123 also includes synthetic data generated by a photo-realistic simulator.

A comprehensive investigation of recent UAV-based datasets is shown in Table 2.1. Note that *Archangel* is the first ever UAV-based dataset collection not only containing both real and synthetic data but also providing an extensive set of metadata, including object poses and UAV positions.

Table 2.1: Comparison of recent UAV-based datasets (1k = 1,000).

General Information								Metadata		
Name	Tasks	Year	#Clips	#Images	Resolution	Syn/Real	Human	Obj. Poses	UAV Pos.	Lighting Cond.
Stanford Drone Dataset [38]	TF	2016	60	929.5k	1400×1904	R	✓	-	-	-
UAV123 [52]	OT	2016	123	112.6k	1280×720	S+R	✓	-	-	-
Okutama-Action [10]	OD, AR	2017	43	77.4k	3840×2160	R	✓	✓	-	-
CARPK [50]	OC	2017	-	1.4k	1280×720	R	-	-	-	-
UAVDT [37]	OD, OT	2018	100	80k	1024×540	R	-	-	✓	✓
VisDrone [9]	OD, OT	2018	263	179.3k, static: 10k	various	R	✓	-	-	-
DroneSURF [53]	FRD	2019	200	411.5k	1280×720	R	✓	-	-	-
AU-AIR [54]	OD	2020	8	32.8k	1920×1080	R	✓	-	✓	-
MOR-UAV [51]	MOR	2020	30	10.9k	various	R	-	-	-	-
DOTA [49]	OD	2021	-	11.3k	various	R	-	-	-	-
UAV-Human [55]	AR, PE PR ATR	2021	AR: 67.4k	PE: 22.5k PR: 41.3k ATR: 22.3k	1920×1080	R	✓	✓	-	-
SeaDroneSee [56]	OD, OT	2022	SOT: 208 MOT: 22	OD: 5.6k MOT: 54.1k	various	R	✓	-	✓	-
A212-Haze [46]	IER, OD	2022	-	1k	1845×1500	R	✓	-	✓	-
DGTA [48]	OD	2022	-	VisDrone: 50k SeaDroneSee: 100k Cattle: 50k	3840×2160	S	✓	-	✓	-
Archangel-Real	OD	2022	69	41.4k	1304×978	R	✓	✓	✓	-
Archangel-Mannequin	OD	2022	598	178.8k	1920×1080	R	✓	✓	✓	-
Archangel-Synthetic	OD	2022	-	4423.7k	512×512	S	✓	✓	✓	✓

Notation	Description	Notation	Description	Notation	Description	Notation	Description
TF	Trajectory Forecasting	OT	Object Tracking	OD	Object Detection	AR	Action Recognition
OC	Object Counting	FRD	Face Recognition & Detection	PE	Pose Estimation	MOR	Moving Object Recognition
PR	Person Re-identification	ATR	Attribute Recognition	SOT/MOT	Single/Multiple Object Tracking	IER	Image Enhancement & Restoration

2.2.2 UAV-based Object Detection Methods

With the rapid development of generic object detection methods [57] and the aforementioned UAV-based object detection benchmarks (Table 2.1), the detection accuracy of UAV-based object detectors has improved significantly over the past few years. In addition to common issues for generic object detection, UAV-based object detection has its own unique challenges [58]. In general, all of the challenges can be roughly divided into three categories. First, objects in UAV-based images are usually much smaller [59]. Therefore, many solutions have been proposed to address this problem to date. For example, Liu et al. [60] proposed HRDNet that fused information from both high- and low-resolution inputs to simultaneously preserve features of small objects and maintain computational costs. Similarly, Liu et al. [61] introduced a multi-branch and parallel structure (MPFPN) to extract more powerful features for tiny object detection. Besides the scale of an object, target objects in UAV-

based datasets are usually crowded and sparsely distributed, reducing both the accuracy and efficiency of an object detector. Thus, Yang et al. [62] proposed ClusDet that performed object cluster proposal first before detecting objects. Finally, UAV-based datasets contain many UAV-specific nuisances [12], such as varying UAV altitudes and viewing angles. These nuisances cause tremendous variations in object appearances, causing degraded detection performance. To address this issue, Wu et al. [12] proposed to adopt adversarial training to learn domain-robust features from UAV-specific nuisances coarsely annotated by the authors. In this chapter, we posit that UAV-based object detection can be further enhanced by providing UAV-based benchmarks with a set of fine-grained metadata, such as that contained in *Archangel*.

In addition to improving the detection accuracy, reducing the computational cost to achieve real-time on-board processing is also very important for UAV-based object detection approaches. One way to improve latency is to skip unnecessary computation. For instance, Ammour et al. [63] proposed to extract candidate regions of target objects first via over-segmentation. After that, only windows around the candidate regions were sent to the pre-trained CNN and linear SVM for feature extraction and classification. Another way of reducing computational overhead is to use more efficient one-stage object detectors, such as YOLO [25, 64], RetinaNet [65], CenterNet [66], and EfficientDet [67]. These one-stage object detectors directly classify and locate objects without generating region proposals, resulting in improved latency. As an example, Liu et al. [68] adapted the original network architecture of YOLO by making it more suitable for UAV-based object detection. In this chapter, we also utilize YOLOv5 [25] for all the experiments due to the advantage of its low complexity (Table 2.2).

Table 2.2: Model complexity of the YOLOv5 variants used in this chapter.

Model	#Parameters (M)	FLOPs (G)
YOLOv5n6	3.1	4.3
YOLOv5s6	12.3	16.2
YOLOv5m6	35.3	49.1

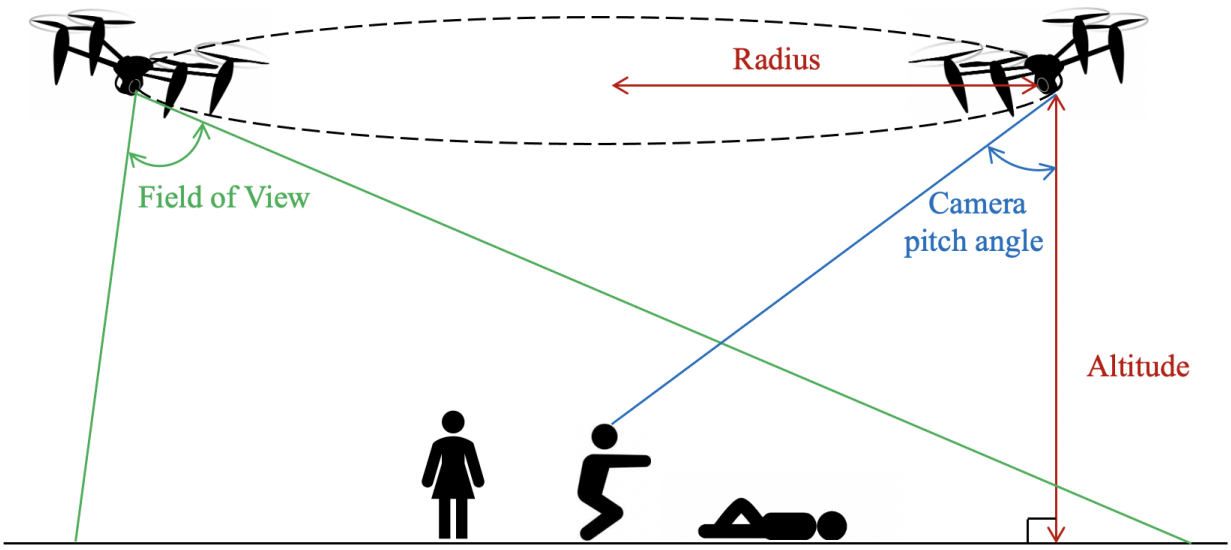


Figure 2.3: Illustration of the data collection process for *Archangel*. For each collection session, multiple objects (real people, mannequins or virtual characters) on the ground were captured by a camera mounted on a UAV (real or simulated). Each objects assumed one of the three predefined poses (*stand*, *kneel*, or *prone*). The UAV circled the objects at predefined altitudes and radii.

2.3 The Archangel Dataset

The data collection process for *Archangel* is illustrated in Fig. 2.3 and a brief comparison of the three sub-datasets is provided in Table 2.3. In the following, we will go through the data collection process of each sub-dataset in detail.

Table 2.3: Comparison of the three sub-datasets that comprise Archangel.

Archangel	Image Size	Targets	Altitude (m)	Radius (m)	Field of View	Camera Pitch Angle
Synthetic	512×512	Virtual characters	[5-80] increment by 5	[5-80] increment by 5	22.5°	various
Mannequin	1920×1080	Mannequins, vehicles	[15-50] increment by 5	[15-50] increment by 5	120°	45°
Real	1304×978	Real people	[15-50] increment by 5	[20-50] increment by 5	45°	22.5°, 45°, 67.5°

2.3.1 Archangel-Mannequin

Target Objects. During this data collection, a group of mannequins were used as human surrogates primarily due to the safety guidelines of the test facility and the difficulty of asking humans to maintain certain strenuous poses, such as *prone* and *squat*, for a long period of time. The mannequins were dressed in casual outfits and positioned in three different poses (i.e., *stand*, *kneel*, and *prone*). Hence, the distribution gap between human attributes and those of mannequins' is not noticeably large. Additionally, the dataset also includes a small group of various types of civilian vehicles as targets, such as sport utility vehicles (SUVs), minivans, and sedans. Each target in the dataset was labeled as *mannequin-standing*, *-kneeling*, *-prone* or *civilian vehicles*.

Data Collection. The imagery was captured using a contractor-built UAV equipped with an onboard electro-optical (EO) camera (ELP-USBFHD01M-L21) with a 1920×1080 pixel array and a lens with approximately 120° field-of-view (FOV). The UAV camera was pitched forward by 45° relative to level flight. During the course of multiple UAV flights, the UAV operated over a wide range of altitudes and radii of rotation circles while keeping the camera pointed inward toward the targets and circling a central point. Both the altitude and radius of the rotation circle were varied from 15-50 meters in 5-meter increments. Since the target objects were stationary and the camera pitch angle was constant, the target objects were spread across different regions of the camera's FOV, resulting in different view angles.

2.3.2 Archangel-Synthetic

Motivation. While *Archangel-Mannequin* provides valuable aerial imagery with pose and position metadata well suited for UAV-based human detection, the imaging conditions of this data collection

were limited. Furthermore, *Archangel-Mannequin* did not incorporate some important factors, such as various human appearances/attributes, extended ranges of UAV altitudes and radii of rotation circles, and different illumination conditions. To overcome these restrictions, a large-scale synthetic imagery (i.e., *Archangel-Synthetic*) dataset containing multiple virtual humans in the same poses as that used in *Archangel-Mannequin* was generated using the Unity game engine [18] to augment *Archangel*.

Data Generation and Labeling. In the Unity-based simulation, a 3D scene is constructed using a terrain asset (i.e., background) and one or more target assets (i.e., virtual characters in different outfits and poses). For the current version of *Archangel-Synthetic*, we use only a simple terrain model (i.e., desert), but we plan to integrate more complex terrain models in future work.

For each target asset, we first created a Unity project and configured the lighting and camera parameters in a virtual 3D environment. A Unity terrain asset (i.e., the desert background) and the target asset were then added to the 3D environment. After the 3D environment was configured as above, a C# script was then used to control the position and viewing angle of the camera as it circled around the target. At each step, the camera was pointed at the center of the target. The script was iterated to encompass the whole range of UAV camera altitudes, the radii of the circles, and the camera viewing angles relative to the target, thus producing imagery captured at various camera-to-target distances and camera pitch angles. Additionally, the sun angle was varied to generate synthetic images captured at different times of a day with corresponding illumination conditions. This resulted in large-scale synthetic imagery with significant target pose, scale, and illumination variations.

To synthesize images and annotations from the virtual 3D environment constructed above, we used an open source software asset, Image Synthesis for Machine Learning [69]. Specifically, the software produced an image segmentation mask where each target object in a synthetic image was as-

signed a unique scalar value. To generate the bounding box annotations for each target, a Python script was used to parse each segmentation mask, identify each target object, and measure the center, width, and height of the tightest bounding box encompassing the target. Additionally, the target category, the camera position, the target orientation relative to the camera, the camera-to-target distance, the camera pitch angle, and the number of pixels inside the segmentation mask were recorded in a single JavaScript Object Notation (JSON) file for each trial.

Properties. *Archangel-Synthetic* includes mountainous desert terrain and eight different virtual characters, each in three different poses (i.e., *stand*, *squat*, and *prone*). In a single trial of synthetic data generation (i.e., a virtual character with a certain pose), both the altitude of the camera and the radius of the rotation circle were varied from 5-80 meters in 5-meter increments. Additionally, the camera viewing angle relative to the character was varied from 0°-358° in 2° increments and four different sun angles were simulated. This resulted in over 4.4M images included in *Archangel-Synthetic*. Each image contains 512×512 pixels with horizontal and vertical fields-of-view of 22.5°.

2.3.3 Archangel-Real

System Design. The dataset was collected with an ARL-designed UAV platform called the Dawn Dove (D2). The D2 is a re-configurable UAV, with the ability to shift the center of gravity by adjusting arm angles, arm placement, battery, sensor payload, and on-board processor location. It is composed of a combination of 3D printed polyethylene terephthalate glycol (PETG) and carbon fiber infused nylon, and traditional carbon fiber parts. It can carry various types of sensor payloads and on-board processors. It has an approximate payload capacity of 1.5 lbs and an approximate flight time of 8 minutes. For this data collection, the sensor payload consisted of a UI-3250ML-C-HQ EO camera

with an Edmund Optics 6mm/F1.4 lens and a FLIR Boson 640 8.7 mm IR camera. The cameras were co-located on the front of the D2 and the EO camera's image was cropped to match the FOV of the FLIR Boson (50° HFOV). The on-board processor was an NVIDIA Xavier NX with a 1 TB NVMe SSD for additional data storage.

Data Collection. In this dataset, the targets consisted of real people wearing civilian clothing in three different poses: *stand*, *kneel*, and *prone*. The data collection process involved having the D2 fly circles at radii ranging from 20-50 meters, at intervals of 5 meters, and at altitudes ranging from 15-50 meters, at intervals of 5 meters. The camera angle relative to level flight was manually adjusted between flights from -22.5°, -45°, and -67.5° to ensure the targets remained within the FOV of the cameras. In total, 52 circles were flown around the targets. To fly the circles, custom Robot Operating System (ROS) based autonomy code was used, along with a custom Python-based graphical user interface (GUI), which communicated with the UAV. From the ground control station (GCS), the target GPS location, circle radius, altitude, maximum velocity, and file name were entered into the GUI. Once the circle parameters were entered, the D2 was manually armed, launched, and switched over to "offboard mode" which passed control of the UAV to the GCS. The GCS then commanded the UAV to perform the autonomous circle. Once complete, the next circle's parameters were entered into the GUI and sent to the UAV while still in the air. This was repeated each flight until the UAV had to be brought back down to replace the battery. In addition to stationary targets, a few circles were also flown where the people walked, jogged, crawled, and waved. Note that *Archangel-Real* involves human subjects as UAV-based detection instances. However, an Institutional Review Board (IRB) approval was exempted since one cannot identify individuals in the dataset.

2.3.4 Importance of Camera Parameters

Before moving on to the data analysis section, we want to highlight the importance of revealing the camera parameters used in the data collection. Note that the scale of human instances in UAV-based object detection datasets, such as *Archangel*, is strongly influenced by the camera parameters used in the data collection, including FOV, pixel-array size, and pitch angles, in addition to the UAV altitude and radius of rotation circle. Hence, the detection results can vary greatly when using different camera parameters. However, all the conclusions derived from the following data analysis of *Archangel* can still be applied to other UAV-based datasets using different camera parameters through extrapolation. That is, the performance gap can be easily calibrated by adjusting the scale of human instances if the camera parameters and the original object size are known *a priori*.

2.4 Experimental Setup

2.4.1 Overview

In this chapter, we designed a series of experiments based on the flow shown in Fig. 2.4. In brief, for each experiment, we selected one of the three pre-trained YOLOv5 models (Table 2.2) and fine-tuned the model on varying amounts of UAV-based real (i.e., *Archangel-Mannequin*) and synthetic (i.e, *Archangel-Synthetic*) data. We then evaluated the model on a sequestered UAV-based dataset (i.e., *Archangel-Real*). Based on the results, the designed experimental flow can provide valuable insights into optimizing UAV-based object detectors with hybrid sets of real and synthetic data.

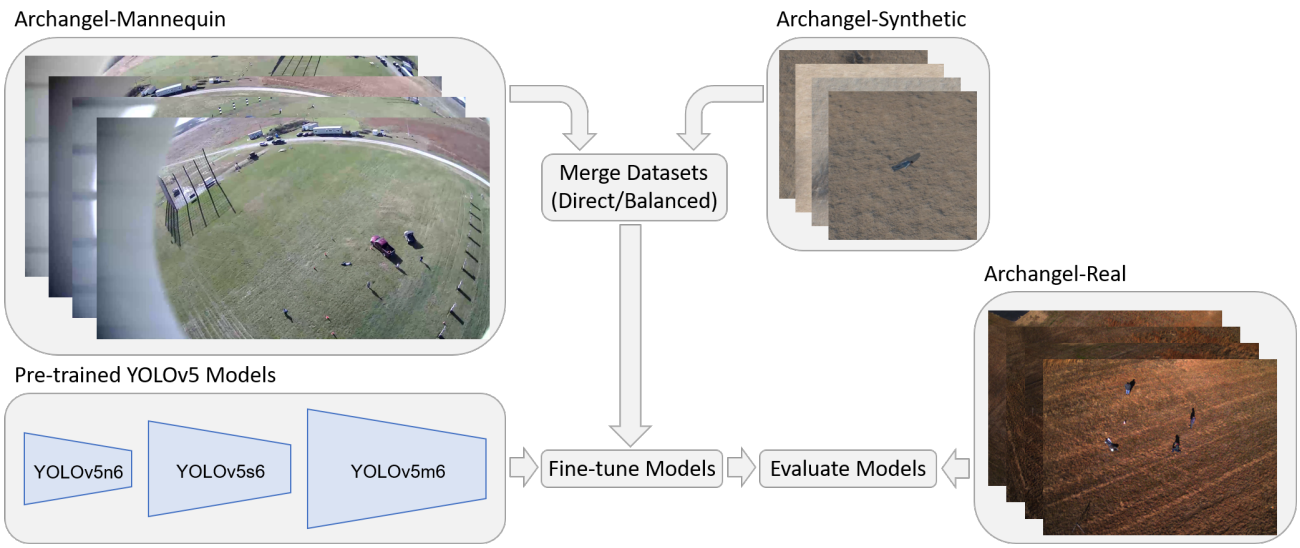


Figure 2.4: Overview of the experimental workflow. Pre-trained YOLOv5 models of varying complexity were fine-tuned on different hybrid sets of UAV-based real and synthetic data, and evaluated on a held-out UAV-based dataset.

2.4.2 Datasets

Note that acquiring the best performance for an UAV-based object detector is not the primary purpose of this study. Thus, we subsampled each of the three sub-datasets of *Archangel* to explore optimal strategies for fine-tuning or evaluating models:

1. *Archangel-Mannequin*: The dataset consists of video clips collected in 11 UAV flight trials. In this chapter, we carefully split the dataset into two subsets so that each covered the whole range of the UAV positions covered during the entire data collection. The video clips collected in 6 of the 11 trials (i.e., Trial-5, 6, 8, 9, 10, 11) were used for evaluating models. The rest (i.e., Trial-1, 2, 3, 4, 7) were used for fine-tuning models. All the video clips were uniformly subsampled at 3 fps. This resulted in two sets of frames, *Arch-Mann-FT37*, containing 6.7k frames for fine-tuning models, and *Arch-Mann-Eval*, containing 11.2k frames for evaluating models. *Arch-Mann-FT37* is named based on the amount of data it has compared to the entire

Archangel-Mannequin in terms of percentage (i.e., 37%).

2. *Archangel-Real*: Similarly, we uniformly subsampled the video clips in *Archangel-Real* at 1 fps. This resulted in a set of frames, named as *Arch-Real-Eval*, containing 4.1k frames for evaluating models.

3. *Archangel-Synthetic*: Only one virtual character in all of the three poses was used. For each UAV position, only one of the four sun angles was randomly selected. Additionally, instead of using all the UAV positions, we uniformly sampled images across each rotation circle in 60° increments. This resulted in a set of images, named as *Arch-Syn-FT*, containing 4.6k images for fine-tuning models.

Each of the three sub-datasets has its own unique usage in this chapter. In general, *Archangel-Real* serves as the primary UAV-based benchmark for measuring detection accuracy. *Archangel-Mannequin* can be viewed as the real UAV-based fine-tuning dataset for adapting the detection models to the target UAV-based domain. Although it includes mannequins instead of real humans, fine-tuning on this dataset is shown to be effective in the following sections. *Archangel-Synthetic*, on the other hand, is used as the synthetic version of the UAV-based fine-tuning dataset, which can be combined with *Archangel-Mannequin* to further optimize the models.

2.4.3 Evaluation

We utilized standard AP50, the average precision with an IoU (Intersection of Union) threshold of 0.5, as the metric to measure the performance of each object detector. Moreover, AP50 was computed for each pose respectively. As more than one pose may exist in a single image, to obtain the performance for only one certain pose, the other two poses were ignored during the evaluation process.

2.4.4 Implementation Details

The official repository of YOLOv5 [25] was used for both fine-tuning and evaluating models. If not specified otherwise, the default hyperparameters were adopted. The input images were rescaled (i.e., *imgsz*=1280) first before being fed into all the models. During fine-tuning, the backbone for each model was frozen (i.e., *freeze*=10) to prevent the model from easily over-fitting. We fine-tuned each model for 20 epochs with a batch size of 16 on a server with 4 NVIDIA GeForce RTX 2080 TI GPUs. During the evaluation, we set the confidence threshold to be 0.05.

2.5 Results

2.5.1 Performance of Pre-trained Models

To begin with, we evaluated the three pre-trained YOLOv5 models on *Arch-Mann-Eval* and *Arch-Real-Eval*. The models were pre-trained on MS-COCO, a representative ground-based dataset. The results are shown in Fig. 2.5. From the results, we can gain several useful insights on UAV-based object detection. First, larger pre-trained models achieved better accuracy across all the evaluation datasets and poses. One possible reason for this is that larger models, compared with smaller ones, can explore better and find more powerful features for classification and detection [70]. Although it implies that we can get higher accuracy by using larger pre-trained models, such larger models may not fit well on small UAV platforms with computational constraints.

Another trend we can observe is that the pre-trained models had much better accuracy on *stand*. It is mainly because that the dataset used for pre-training models (i.e., MS-COCO) contains significantly more human instances in *stand* (i.e., 84.53%) [71]. In other words, it is impractical to directly

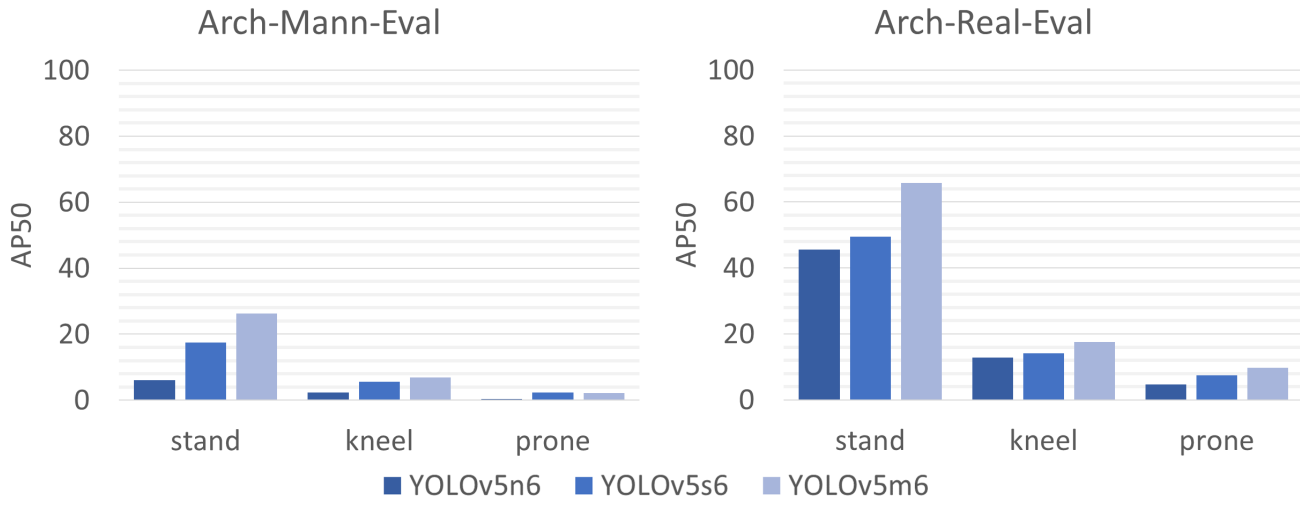


Figure 2.5: AP50 of pre-trained YOLOv5 models evaluated on *Arch-Mann-Eval* and *Arch-Real-Eval*.

use detectors pre-trained on standard datasets, especially in unusual scenarios where we need to detect people in uncommon positions, such as search and rescue in disaster relief. It is worth mentioning that the pre-trained YOLOv5 models performed much worse on *Arch-Mann-Eval* than on *Arch-Real-Eval*. That is mainly because *Arch-Mann-Eval* contains some other objects, such as traffic cones and fiducials, which are easily misclassified as humans when captured by the pre-trained detectors at high altitudes [39].

2.5.2 Fine-tuning on a Real UAV-based Dataset

As we have discussed, most ground-based datasets used to fine-tune models usually lack UAV-specific samples for the models to learn from, such as human instances in non-standing positions captured from various camera viewing angles and altitudes. Therefore, we allowed the pre-trained YOLOv5 models to acquire such knowledge by fine-tuning the models on *Arch-Mann-FT37*. Moreover, given the significant challenges of collecting and annotating UAV-based datasets [9] and the lack of existing large-scale UAV-based object detection benchmarks, we explored the idea of fine-

tuning models in the small-data regime [72]. More precisely, we subsampled the original fine-tuning dataset and created several smaller subsets for fine-tuning, which contained much less data (i.e., *Arch-Mann-FT20*, *Arch-Mann-FT10*, *Arch-Mann-FT5* and *Arch-Mann-FT2*). We followed the same naming strategy as *Arch-Mann-FT37* for the extra fine-tuning datasets.

The results are presented in Fig. 2.6. We would like to highlight the importance of having an evaluation dataset with different characteristics from the fine-tuning dataset. As we fine-tuned the models on data from *Arch-Mann-FT37*, we could improve their detection accuracy on a similar evaluation dataset such as *Arch-Mann-Eval*. However, the models fine-tuned on too much data from *Arch-Mann-FT37* tended to perform worse on *Arch-Real-Eval*. We argue that this was because the models started to learn certain dataset-specific features from the fine-tuning dataset, adversely affecting the models' generalization capability to unseen datasets, such as the evaluation dataset in our case. In practice, ML models embedded into UAVs are usually deployed to new environments unseen during training. Thus, in the following experiments, we chose to fine-tune models on data from *Arch-Mann-FT37* and *Arch-Syn-FT* but evaluate them on *Arch-Real-Eval*.

2.5.3 Fine-tuning on Real and Synthetic Datasets

One of the significant advantages of *Archangel* is that it contains both real and synthetic subsets acquired from similar imaging conditions in data collections and synthetic rendering, respectively. Hence, investigating the effect of augmenting the original UAV-based fine-tuning datasets with UAV-based synthetic data is another major topic for this chapter. To do so, we fine-tuned the pre-trained YOLOv5 models on *Arch-Syn-FT*. Additionally, a well-known concern about learning from synthetic data is that, compared with real data, synthetic data usually contains much less variations in appear-

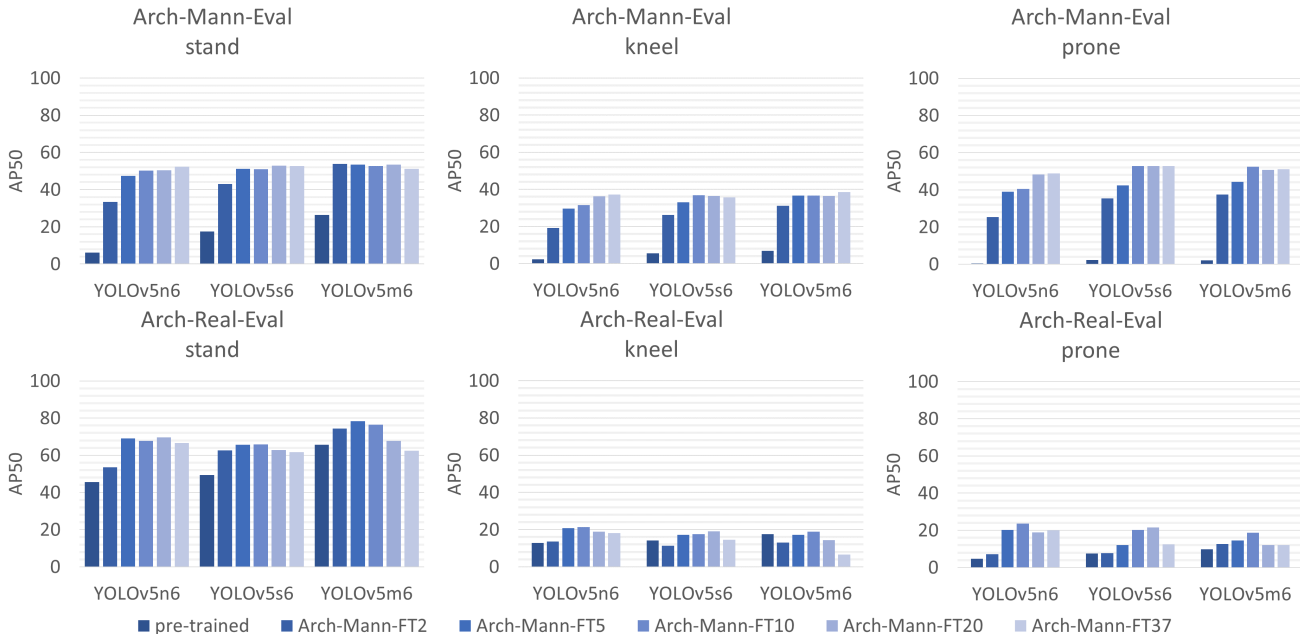


Figure 2.6: AP50 of YOLOv5 models fine-tuned on different subsets of *Arch-Mann-FT37*. Each model was evaluated on *Arch-Mann-Eval* (top) and *Arch-Real-Eval* (bottom).

ances/attributes of objects or structures of scenes [73]. Therefore, instead of fine-tuning models only on *Arch-Syn-FT*, we also explored the idea of multi-source learning [74], constructing multiple hybrid fine-tuning datasets by directly merging *Arch-Syn-FT* with all the subsets of *Arch-Mann-FT37* used in the previous experiment.

The results are shown in Fig. 2.7. For *prone*, a rarely seen pose in the pre-training dataset, the detection accuracy of the pre-trained models was very low. After fine-tuning the models on the various

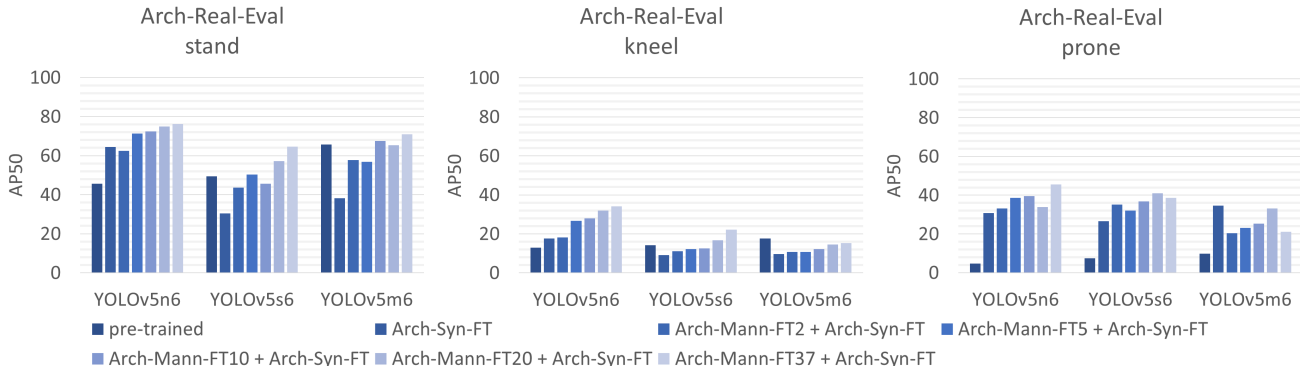


Figure 2.7: AP50 of YOLOv5 models fine-tuned on hybrid sets constructed from *Arch-Syn-FT* and different subsets of *Arch-Mann-FT37*.

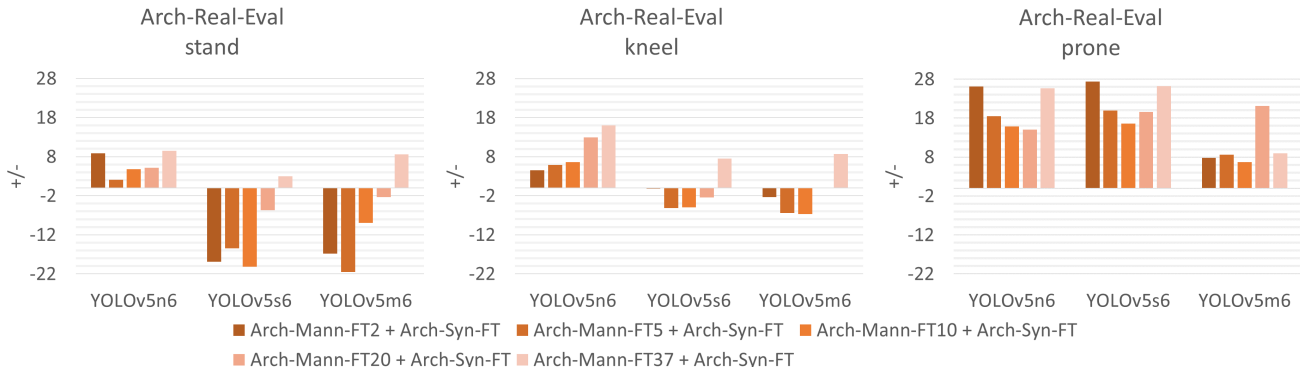


Figure 2.8: Improvement in AP50 of YOLOv5 models after adding *Arch-Syn-FT* into the original fine-tuning datasets based on different subsets of *Arch-Mann-FT37*.

hybrid subsets of *Arch-Syn-FT* and *Arch-Mann-FT37*, the detection accuracy continually increased with few exceptions. For *stand*, the pre-trained models performed much better than they did for *prone* as expected, but fine-tuning on the hybrid sets of the real and synthetic data still provided much improvement over the pre-trained models, as clearly observed from the detection accuracy of *YOLOv5n6*. Similar observations can be made for *kneel*.

We now compare the results shown in Fig. 2.6 with the ones shown in Fig. 2.7 to highlight the effect of adding the synthetic data to the fine-tuning dataset based only on the subsets of *Arch-Mann-FT37*. The results are shown in Fig. 2.8. For *prone*, we can observe significant performance improvement across all the models and hybrid subsets for fine-tuning after introducing the synthetic data into the fine-tuning datasets, compared with the results of fine-tuning on data from *Arch-Mann-FT37* only. One explanation for the above finding is that most of the human instances in the pre-training dataset are in certain upright positions, including *stand* and *kneel*. Therefore, adding synthetic characters in *prone* captured from various camera viewing angles and UAV altitudes to the fine-tuning dataset can effectively aid the models to learn how to detect humans in *prone*. More in-depth analysis of this issue will be discussed in the ablation study (Sec. 2.6).

Additionally, *Arch-Syn-FT*, which includes only one virtual character and one type of back-

ground, might be too simple for larger models, such as *YOLOv5s6* and *YOLOv5m6* in our case, to learn from, causing them to overfit to the fine-tuning dataset. A proof of this assumption was that using *Arch-Syn-FT* along with the subsets of *Arch-Mann-FT37* to fine-tune *YOLOv5s6* and *YOLOv5m6* significantly decreased their performance on *stand* and *kneel*, especially when we included fewer data from *Archangel-Mann-FT37* (Fig. 2.8). On the other hand, fine-tuning *YOLOv5n6* on *Arch-Syn-FT* along with the subsets of *Arch-Mann-FT37* did not have such a negative effect. As a result, in the following ablation study, we focused on analyzing the results of *YOLOv5n6* once we included *Arch-Syn-FT* in the fine-tuning dataset.

2.6 Ablation Studies

2.6.1 Effect of Synthetic Dataset Size

We have demonstrated the effects of directly combining the various subsets of *Arch-Mann-FT37* with the same set of synthetic data (i.e., *Arch-Syn-FT*) for fine-tuning models (Fig. 2.7 and 2.8). In this section, we are interested in further exploring the outcome of using different amounts of synthetic data to fine-tune models. Notably, we aim to investigate whether a "balanced" fine-tuning dataset, which contains the same amount of real and synthetic data, is better than its "unbalanced" counterpart, in which the ratio of the synthetic data to the real data varies due to the use of a fixed set of the synthetic data across all the hybrid fine-tuning datasets.

To achieve this goal, for each real fine-tuning dataset (i.e., *Arch-Mann-FT37*, *Arch-Mann-FT20*, *Arch-Mann-FT10*, *Arch-Mann-FT5*, and *Arch-Mann-FT2*), the corresponding amount of data was randomly sampled from *Arch-Syn-FT* to match the real fine-tuning dataset. Namely, if the real fine-tuning dataset was smaller than *Arch-Syn-FT*, a subset of *Arch-Syn-FT* was randomly selected and combined

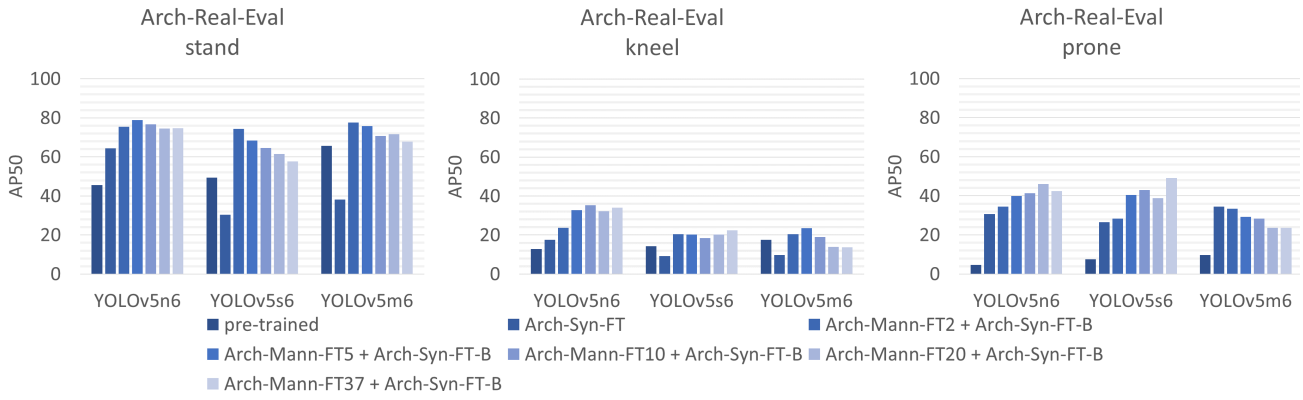


Figure 2.9: AP50 of YOLOv5 models fine-tuned on balanced hybrid sets constructed from *Arch-Syn-FT-B* and different subsets of *Arch-Mann-FT37*.

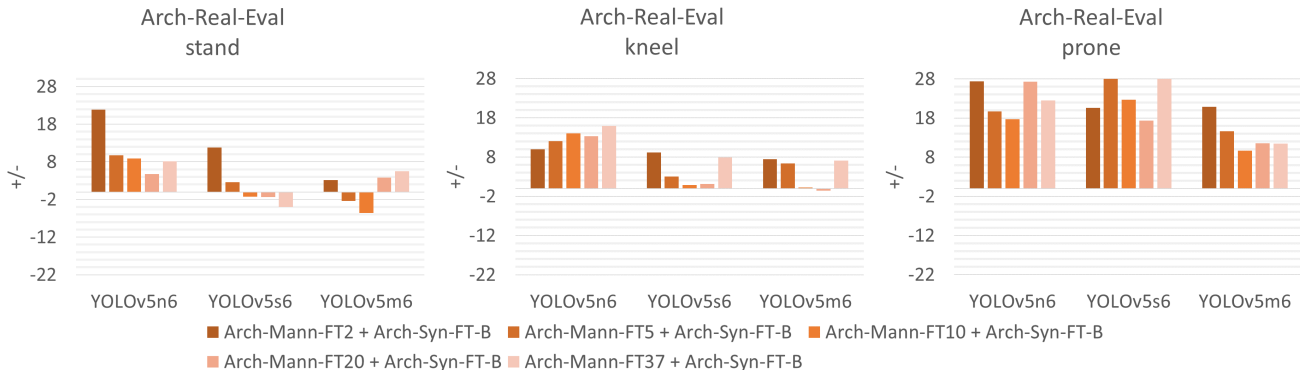


Figure 2.10: Improvement in AP50 of YOLOv5 models after adding *Arch-Syn-FT-B* into the original fine-tuning datasets based on different subsets of *Arch-Mann-FT37*.

with the real fine-tuning dataset to form a balanced fine-tuning dataset. Similarly, if the real fine-tuning dataset was larger, a random subset of *Arch-Syn-FT* was duplicated before the combination. We denote the synthetic fine-tuning dataset as *Arch-Syn-FT-B* if the aforementioned data balancing procedure has been conducted.

The results are shown in Fig. 2.9, 2.10 and 2.11. Comparing Fig. 2.8 with Fig. 2.10, we can observe that the negative effect of fine-tuning larger models (i.e., *YOLOv5s6* and *YOLOv5m6*) on hybrid fine-tuning sets largely decreases or even diminishes. Moreover, the improvement becomes more significant, especially for *YOLOv5n6*. From Fig. 2.11, it is shown that fine-tuning *YOLOv5n6* on the balanced hybrid sets of the real and synthetic data can always be at least on par with the best

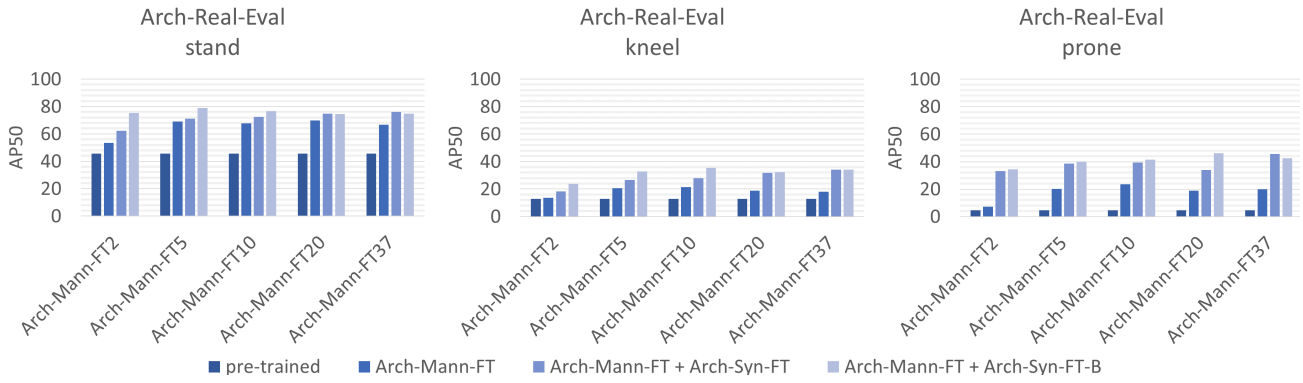


Figure 2.11: AP50 comparison of *YOLOv5n6* fine-tuned on hybrid sets constructed from *Arch-Syn-FT*, *Arch-Syn-FT-B*, and different subsets of *Arch-Mann-FT37*.

performance setting for fine-tuning. These findings indicate that the proportion of each data to the whole fine-tuning dataset significantly impacts the model's fine-tuning performance.

2.6.2 Leave-One-Pose-Out Analysis

We have claimed that fine-tuning models with synthetic human instances are particularly important for *prone*, a pose rarely seen in the original training data. In this section, we would like to provide more evidence to support this statement. Specifically, we used *YOLOv5n6* for this set of experiments since it showed less sign of overfitting when fine-tuning on *Arch-Syn-FT* (Fig. 2.7). Additionally, the technique of dataset balancing was adopted due to its positive effect on fine-tuning models (Fig. 2.9 and 2.10). We followed a similar procedure to generate each hybrid fine-tuning dataset, except that each time one of the poses, *stand*, *kneel*, or *prone*, was excluded in advance from *Arch-Syn-FT*, resulting in a "leave-one-pose-out" fine-tuning dataset, *Arch-Syn-FT-NoSt*, *Arch-Syn-FT-NoKn*, or *Arch-Syn-FT-NoPr*, respectively.

The results are presented in Fig. 2.12. In general, the detection accuracy of *stand* and *kneel* did not obviously change when we removed any one of the poses from the synthetic fine-tuning dataset. However, the performance of *prone* degraded drastically when we excluded the virtual characters in

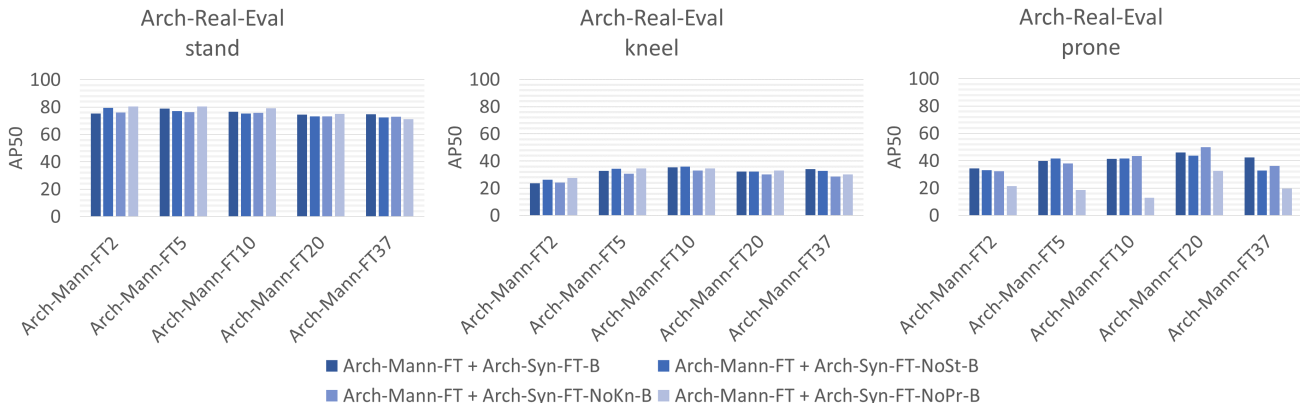


Figure 2.12: AP50 of *YOLOv5n6* fine-tuned on balanced hybrid sets constructed from three leave-one-pose-out synthetic fine-tuning datasets (*Arch-Syn-FT-NoSt-B*, *Arch-Syn-FT-NoKn-B*, and *Arch-Syn-FT-NoPr-B*) and different subsets of *Arch-Mann-FT37*.

prone from the fine-tuning dataset, which strongly supports our earlier claim.

2.6.3 Precise Model Diagnosis: Performance on the Altitude-Radius Grid

So far, we have demonstrated that the detection accuracy of the pre-trained YOLOv5 models on the UAV-based evaluation dataset can be considerably boosted by fine-tuning the models on the hybrid sets of UAV-based real and synthetic data. For example, we have shown that the AP50 of the pre-trained *YOLOv5n6* can be increased by about 30 in AP value by fine-tuning it on a joint set of *Arch-Mann-FT2* and *Arch-Syn-FT-B* (Fig. 2.11). In this section, we further analyze this particular example with the complete information about the UAV positions over the altitude-radius grid as shown in Fig. 2.13. Our goal is to give an idea of how to utilize the metadata provided by *Archangel* to diagnose problems with a UAV-based object detection model.

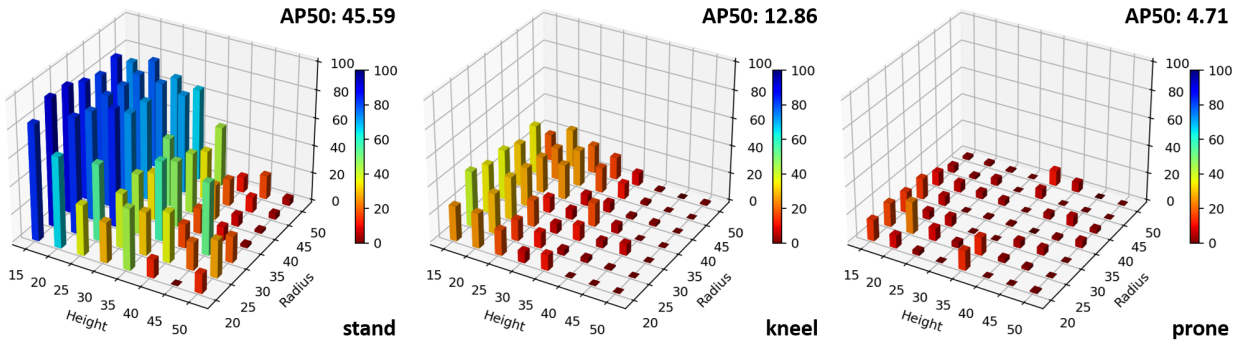
The results are presented in Fig. 2.13. In this figure, we can clearly observe how the pre-trained *YOLOv5n6*'s performance gradually progresses with the different fine-tuning datasets, from the real-data-only dataset to the unbalanced hybrid dataset to the balanced hybrid dataset. Initially, the pre-trained *YOLOv5n6* performs fairly well at the low altitudes but fails at the high altitudes due to the

curse of the pre-training dataset, which is composed mostly of ground-based human instances. After being fine-tuned on *Arch-Mann-FT2*, the fine-tuned *YOLOv5n6* gets much better at detecting human instances from a relatively higher altitude or larger circle radii. However, the detection accuracy of the human instances at close range decreases considerably. We argue that this is mainly because the image resolution of *Arch-Mann-FT2* (i.e., 1920x1080) is much larger than that of the evaluation dataset (i.e., 1304x978) and the bounding boxes contained in *Arch-Mann-FT2* are generally small. In other words, fine-tuning *YOLOv5n6* on *Arch-Mann-FT2* inevitably causes a bias on the model toward detecting tiny objects. In contrast, this negative effect does not occur when we fine-tune the model on the hybrid set of *Arch-Mann-FT2* and *Arch-Syn-FT* (or *Arch-Syn-FT-B*). We believe that this is because our synthetic dataset covers an extended range of UAV positions over the grid so as to cause less bias on the fine-tuning process.

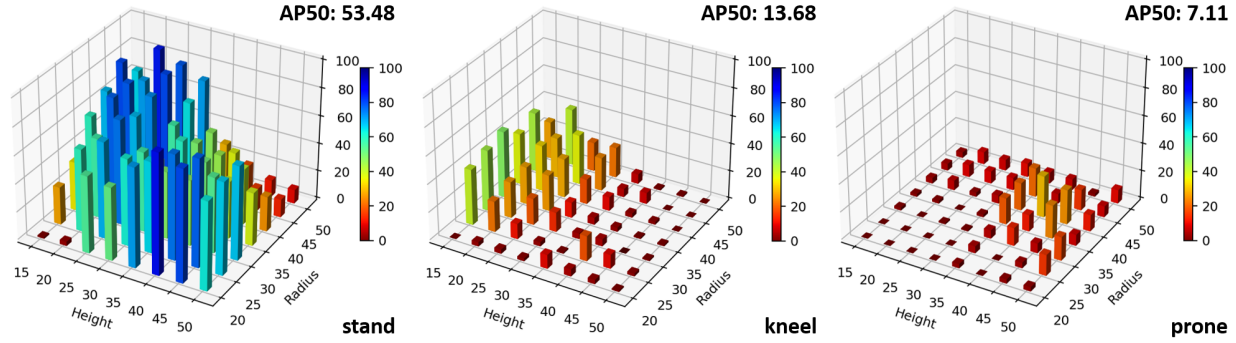
Based on the results shown in Fig. 2.13, to further improve the model's detection accuracy, we can focus on improving the model's performance at high altitudes and large circle radii, or the performance of non-standing positions, such as *kneel* and *prone*. Notably, we find that the performance improvement of *kneel* is surprisingly small to high altitudes and large circle radii, which requires a deeper investigation in future work.

2.7 Discussion and Future Directions

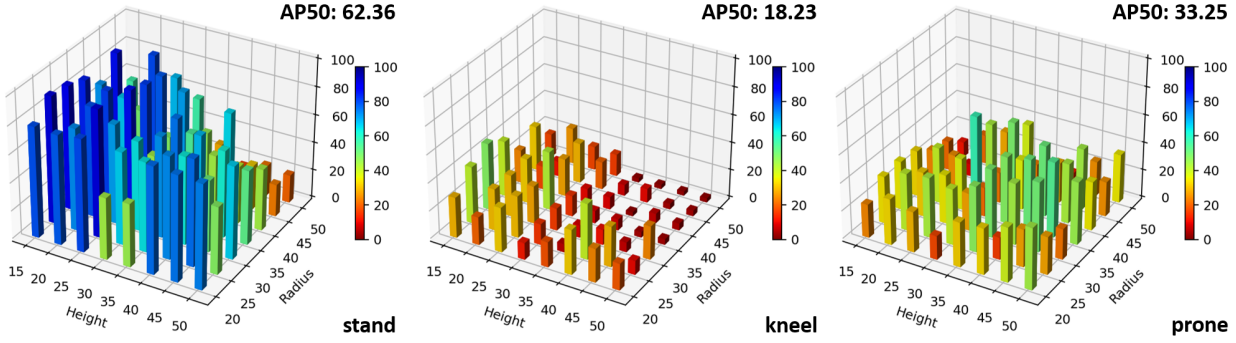
With the series of experiments presented in Sec. 2.5 and 2.6, we have demonstrated the distinctive value of the *Archangel* dataset. Particularly, we have clearly illustrated how to utilize the dataset's metadata to evaluate and diagnose the UAV-based object detectors on the altitude/radius grid. Moreover, we have systematically analyzed how to involve both real and synthetic data within the UAV-



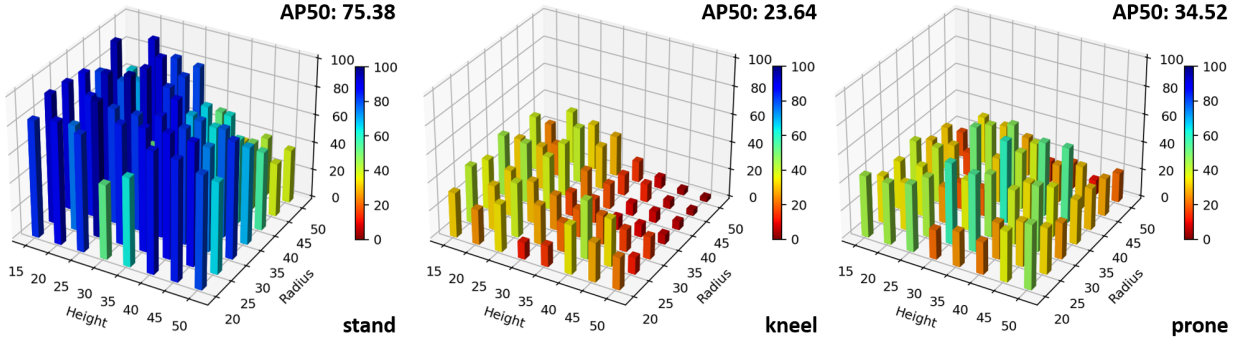
(a) pre-trained



(b) Arch-Mann-FT2



(c) Arch-Mann-FT2 + Arch-Syn-FT



(d) Arch-Mann-FT2 + Arch-Syn-FT-B

Figure 2.13: AP50 comparison across altitude-radius grids for *YOLOv5n6* fine-tuned on hybrid sets constructed from *Arch-Mann-FT2*, *Arch-Syn-FT*, and *Arch-Syn-FT-B*.

based fine-tuning process. Such fundamental studies of UAV-based perception had not been achieved until we curated *Archangel*.

Despite having all these merits, *Archangel* is still in its early stage and has much room for improvement and development. In the following, we suggest a few possible future directions regarding *Archangel*:

2.7.1 Direct Extensions of This Study

Many results shown in Sec. 2.5 and 2.6 imply that increasing the data diversity of *Archangel* is one of the most promising future research directions. For instance, since we have demonstrated that fine-tuning models with virtual characters in unusual poses is particularly effective, we can include more atypical poses into *Archangel* to further explore this phenomenon, which is crucial especially in search and rescue scenarios where finding people in severe physiological states is the priority. Additionally, we can diversify the appearances/attributes of either the real or synthetic human instances in the dataset, investigating whether this will resolve the issue of overfitting as we have discussed earlier. For the same purpose, we can increase the diversity of components beyond the foreground objects, such as the real and synthetic backgrounds included in *Archangel*. Finally, we can extend *Archangel* to include more object categories, such as various types of vehicles, which frequently exist in other UAV-based object detection datasets (Table 2.1) so that *Archangel* can be used in conjunction with those datasets.

Next, exploring more sophisticated fine-tuning strategies is another potential direction to extend this chapter. In this chapter, we have demonstrated that the performance of the pre-trained SoTA object detector can be boosted considerably by fine-tuning the model on a balanced UAV-based fine-

tuning dataset constructed by directly merging a real subset and a synthetic subset. Nevertheless, it is worth exploring if there is a better strategy for sampling each subset or merging the two subsets. For instance, to build up a balanced fine-tuning dataset, instead of randomly selected samples from the synthetic fine-tuning dataset, we may do the sampling based on certain distance measurements.

2.7.2 UAV-based Visual Representation Learning with Metadata

In this chapter, we used the position and pose metadata provided by *Archangel* only for accurate model evaluation and diagnosis. However, we believe that there is a huge potential to utilize such metadata during training for better visual representation learning. A notable example for this is NDFT [12], where the authors exploited adversarial training with the coarse metadata labeled by themselves to enhance the robustness of the learned features for UAV-based object detection. We expect that such a framework will benefit greatly from the extensive metadata provided by *Archangel*. Beyond UAV-based perception, in medical [75] and underwater [76] imaging, it has also been shown that dataset metadata is useful for learning visual representation with self-supervised learning or contrastive learning.

2.7.3 UAV-based Synthetic Data Generation and Augmentation

We have found that fine-tuning larger models with the synthetic data that we generated often causes the issue of overfitting. This issue might be mitigated by directly synthesizing more diverse images. However, as we have discussed, there is usually a huge domain gap between the synthetic data and real data in terms of object appearances/attributes and scene structures, which may not be solved by simply increasing the number of synthetic images. Hence, addressing this domain gap issue within the

scope of UAV-based perception is another important future research direction for *Archangel*. Possible solutions include: (1) jointly training an object detector with a generative model which transforms synthetic images to be more visually realistic [26, 77], and (2) formulating the process of synthetic data generation as a learning problem to synthesize scene structures better matching real-world scene distributions [78].

2.8 Conclusion

In this chapter, we introduce a unique UAV-based object detection dataset, *Archangel*, to encourage the community to continue developing more effective UAV-based object detection approaches with dataset metadata and synthetic data. A comprehensive study is carefully designed to show how to utilize *Archangel* to fully optimize a state-of-the-art object detector with a hybrid fine-tuning dataset comprising both real and synthetic data. Additionally, we also demonstrate the huge benefit of leveraging the dataset metadata during model evaluation by comparing the performance of the model across the different object poses and UAV positions. As we have discussed, although there is still much room for improvement, we hope that *Archangel* is useful for the broader machine learning community and can lead to future advances in the area of UAV-based perception.

Chapter 3: Progressive Transformation Learning for Leveraging Virtual Images in Training

3.1 Introduction

Training an object detector usually requires a large-scale training image set so that the detector can acquire the ability to detect objects' diverse appearances. This desire for a large-scale training set is bound to be greater for object categories with more diverse appearances, such as the human category whose appearances vary greatly depending on its pose or viewing angles. Moreover, a person's appearance becomes more varied in images captured by an unmanned aerial vehicle (UAV), leading to a wide variety of camera viewing angles compared to ground-based cameras, making the desire for a large-scale training set even greater. In this chapter, we aim to satisfy this desire, especially when the availability of UAV-based images to train a human detector is scarce, where this desire is more pressing.

As an intuitive way to expand the training set, one might consider synthesizing virtual images to imitate real-world images by controlling the optical and physical conditions in a virtual environment. Virtual images are particularly useful for UAV-based object detection since abundant object instances can be rendered with varying UAV locations and camera viewing angles along with ground-truth information (e.g., bounding boxes, segmentation masks) that comes free of charge. Therefore, a large-scale virtual image set covering diverse appearances of human subjects that are rarely shown in existing UAV-based object detection benchmarks [9, 10, 11] can be conveniently acquired by controlling entities and parameters in a virtual environment, such as poses, camera viewing angles, and illumination conditions.

To make virtual images usable for training real-world object detection models, recent works [19,

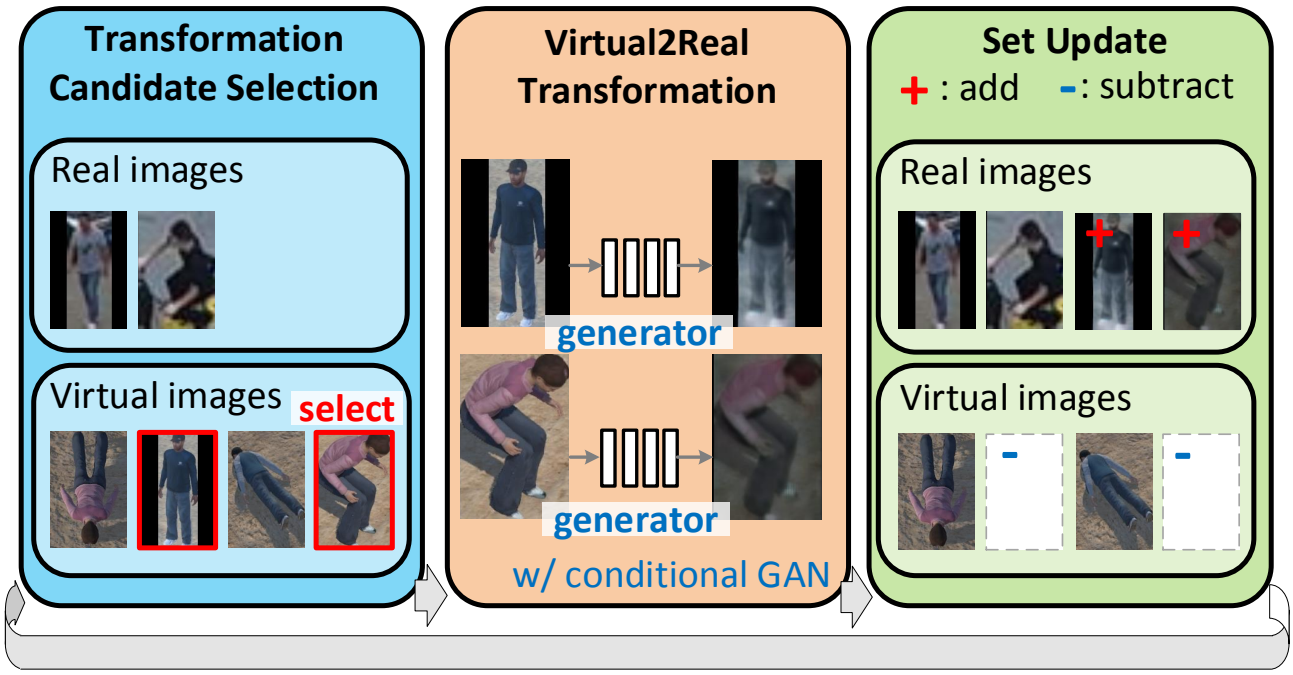


Figure 3.1: Overview of Progressive Transformation Learning.

79, 80, 81] transform virtual images to look realistic. They commonly use the virtual2real generator [82] trained with the conditional GAN framework to transform images in the source domain to have the visual properties of images in the target domain. Here, virtual and real images are treated as the source and target domains, respectively. However, the large discrepancy in visual appearance between the two domains, referred to as the "domain gap", result in the degraded transformation quality of the generator. In fact, the aforementioned works using virtual images validate their methods where the domain gap is not large (e.g., digit detection [19]) or when additional information is available (e.g., animal pose estimation with additional keypoint information [79, 80, 81]). In our case, real and virtual humans in UAV-based images inevitably have a large domain gap due to the wide variety of human appearances.

To address the large domain gap, one critical question inherent in our task is *how to measure*

accurately the domain gap. Consequently, we estimate the domain gap in the representation space of a human detector trained on the real images. The representation space of the detector is learned such that test samples, which have significantly different properties than training samples from the perspective of the detector, are located away from the training samples. In this chapter, we show that the feature distribution of object entities belonging to a certain category, such as the human category, in the representation space can be modeled with a multivariate Gaussian distribution if the following two conditions are met: i) the detector uses the sigmoid function to normalize the final output and ii) the representation space is constructed using the output of the penultimate layer of the detector. This idea is inspired by [83], which shows that softmax-based classifiers can be modeled as multivariate Gaussian distributions. In this chapter, we show that the proposition is also applicable to sigmoid-based classifiers, which are widely used by object detectors. Based on this modeling, when the two aforementioned conditions are met, the human category in the representation space can be represented by two parameters (i.e., mean and covariance) of a multivariate Gaussian distribution that can be computed on the training images. With the empirically calculated mean and covariance, the domain gap from a single virtual human image to real human images (i.e., the training set) can be measured using the Mahalanobis distance [84].

To add virtual images to the training set to include more diverse appearances of objects while preventing the transformation quality degradation caused by large domain gaps, we introduce *Progressive Transformation Learning (PTL)* (Figure 3.1). PTL progressively expands the training set by adding virtual images through iterating the three steps: 1) transformation candidate selection, 2) virtual2real transformation, and 3) set update. When selecting transformation candidates from a virtual image pool, we use weighted random sampling, which gives higher weights to images with smaller domain gaps. The weight takes an exponential term with one hyperparameter controlling the ratio

between images with smaller domain gaps and images with more diverse appearances. Then, the virtual2real transformation generator is trained via the conditional GAN, taking the selected transformation candidates as the "source" and the images in the training set as the "target". After transforming the transformation candidates by applying the virtual2real transformation generator, the training set is expanded with the transformed candidates while the original candidates are excluded from the pool of virtual images.

Material in this chapter was published in [26]. The main contribution of this chapter is that we have validated the utility of virtual images in augmenting training data via PTL coupled with carefully designed comprehensive experiments. We first use the task of low-shot learning, where adequately expanding datasets has notable effects. Specifically, PTL provides better accuracy on three UAV-view human detection benchmarks than other previous works that leverage virtual images in training, as well as methods that only use real images. Then, we validate PTL on the cross-domain detection task where training and test sets are from distinct domains and virtual images can serve as a bridge between these two sets. The experimental results indicate that a high-performance human detection model can be effectively learned via PTL, even with the significant lack of real-world training data.

3.2 Related Work

3.2.1 Leveraging Virtual Images in Training

In this section, we have listed previous works that demonstrate how virtual images can play a role in a variety of real-world applications when used for training. In fact, virtual images are desirable for model training as large-scale labeled datasets can be built virtually almost free of charge. Unfortunately, when virtual images are used without proper care, it is shown that the performance improvement

is limited due to the domain gap between the virtual images and the real test images. Generally, previous works leveraging virtual images during model training can be summarized into three approaches according to how they exploit the advantages of virtual images and address the challenges of using virtual images.

The most intuitive and widely used approach of using virtual images is to pre-train a model on virtual images and fine-tune the pre-trained model on real images acquired in the same domain as the test images [79, 85, 86, 87, 88, 89, 90, 91, 92, 93]. This approach aims to avoid the domain gap by fine-tuning the model on real images acquired under the same conditions and environments of test images. While the first approach seeks to use the representative capability learned from *large-scale* virtual datasets, the second approach seeks to exploit additional information that can be easily *labeled* on virtual images. For example, [94] annotates part segmentation maps when acquiring virtual vehicle images, and uses the part segmentation results from the pre-trained model on the virtual images during the fine-tuning process. Similarly, [95] uses depth and semantic information labeled when acquiring virtual images.

The third approach directly builds training batches consisting of both virtual images and real images. [96] and [97] adopt the most naive approach to build a batch by randomly selecting a fixed number of images from each of the real and virtual image sets. However, even if the number of virtual images is several orders of magnitude greater than the number of real training images, this approach does not provide remarkably better accuracy and may even provide worse accuracy than its counterparts using only real images for training. In this case, the effect of using virtual images during model training does not appear as expected because the large domain gap between the real (test) images and the virtual images is not adequately addressed. In this chapter, we also use virtual images directly for model training while considering appropriately reducing this domain gap.

3.2.2 Progressive Learning

Progressive learning is a machine learning strategy that continuously trains a model from easy to hard tasks, primarily for the purpose of training stabilization or fast optimization. One of the most common applications for progressive learning is incrementally increasing the network capacity to improve network capability. The most intuitive approach in this category is to gradually increase the network size (e.g., depth or width) to ease the training difficulty of very deep networks [98, 99, 100, 101, 102, 103, 104]. Conversely, [105] uses progressive learning in the direction of reducing the network size for fast training. Progressive learning is also used in GAN frameworks to enhance the generator's ability to transform input images of larger resolution [106]. Curriculum learning [107, 108, 109, 110], which continuously raises the level of training from easy to difficult samples, also falls into this category.

Progressive learning is also used to deal with the scalability of datasets that are incompletely labeled. In a semi-supervised learning task, [111, 112, 113] adopt progressive learning by gradually increasing the number of unlabeled data used for training. Self-learning in [114, 115] uses progressive learning by repeating the two steps, assigning labels depending on the current detector and updating the current detector with these labels.

Our method can be seen as similar to the second approach in that it also intend to expand the training dataset. However, our method uses progressive learning to reliably add realistically transformed virtual images to the training dataset by avoiding quality degradation of the transformation, which has never been attempted before using the progressive learning strategy.

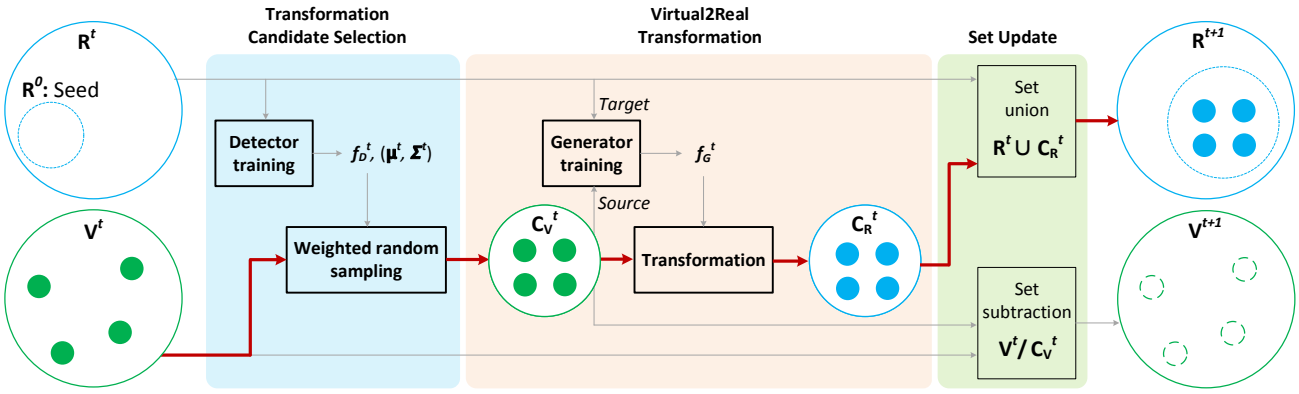


Figure 3.2: Progressive Transformation Learning (PTL) pipeline. The red arrow indicates the processing flow of virtual images selected for inclusion in the training set.

3.3 Method

3.3.1 Measuring Domain Gap between Real and Virtual Images

Modeling Training Set with Multivariate Gaussian Distribution. The purpose of adopting progressive transformation learning, which progressively expands the training set with a subset of the realistically transformed virtual images instead of expanding it with the full set at once, is to avoid the significant domain gap between the real and virtual images when training the transformation generator. Here, the domain gap is measured in the representation space of the detector, which is learned so that two samples with different properties from the perspective of the detector are separated far from each other.

In general, the representation space of the detector refers to the space formed by the output of the penultimate layer of the detector since all layers of the detector except for the last layer can be transferred for different downstream tasks [116, 117, 118]. In [83], it is shown that for the softmax-based classifier, the distribution of each category in the representation space can be modeled as a multivariate Gaussian distribution. Object detector generally uses the sigmoid function (i.e.,

$f_{sigmoid}(x) = 1/(1+\exp(-w_c^T x - b_c))$ for the category c), which does not consider outputs for other categories, instead of the softmax function (i.e., $f_{softmax}(x) = \exp(w_c^T x + b_c)/\sum_{c'}\exp(w_{c'}^T x + b_{c'})$ for the category c) that competes for outputs for all categories to normalize the model output to [0 1]. This is because, unlike classification, the detection task must take into account that two or more co-located objects may be active on a single output. In Sec. 3.6.1, we show that even for the sigmoid-based detector, the distribution of each category in the representation space can also be modeled as a multivariate Gaussian distribution.

Specifically, let $x \in \mathcal{X}$ and $y = \{y_c\}_{c=1,\cdots,C} \in \mathcal{Y}, y_c \in \{0,1\}$ be an input and its label, respectively. Then, the representation space of the sigmoid-based detector can be expressed as follows:

$$P(f(x)|y_c = 1) \sim \mathcal{N}(f(x)|\mu_c, \Sigma_c), \tag{3.1}$$

where $f(\cdot)$ denotes the output of the penultimate layer of the detector. μ_c and Σ_c are the mean and the covariance of the multivariate Gaussian distribution for the category c. μ_c and Σ_c can be calculated over the entire set of training images as follows:

$$\begin{aligned} \mu_c &= \frac{1}{|\mathrm{D}_c|}\sum_{x\in\mathrm{D}_c} f(x), \\ \Sigma_c &= \frac{1}{|\mathrm{D}_c|}\sum_{x\in\mathrm{D}_c} (f(x)-\mu_c)\,(f(x)-\mu_c)^\top, \end{aligned} \tag{3.2}$$

where D_c is the set of instances for the category c. Practically, any detection whose IoU with the groundtruth of the category c is greater than 0.5 belongs to D_c.

Measuring Domain Gap. After μ_c and Σ_c are empirically calculated to represent D_c, the domain gap

between a new image x_{new} and D_c can be measured using the Mahalanobis distance, as follows:

$$d(x_{new}) = (f(x_{new}) - \mu_c)^\top \Sigma_c^{-1} (f(x_{new}) - \mu_c) . \tag{3.3}$$

This measurement of the domain gap is highly dependent on the detector's ability to detect objects in the image. It is commonly known that the detection capability of a detector is greatly affected by the image size as well as the object appearance in the image. To mitigate the effect of image size on measuring the domain gap, the Mahalanobis distance for $\mathbf{x}_{new}$ is calculated at multiple image scales, and the minimum distance is used as the domain gap, as follows:

$$d(x_{new}) = \min_{s \in S}(\{d(x_{new}^s)\}), \tag{3.4}$$

where x_{new}^s is the resized image of x_{new} to be $s \times s$. S is the set of resizing factors. In our experiments, we use $S = \{128, 256, 384, 512\}$.

3.3.2 Progressive Transformation Learning

Our objective is to expand the training set consisting of real images by adding virtual images which are transformed to intimate real images. The virtual2real transformation can be performed by a generator trained by treating virtual images and real images as "source" and "target", respectively, in the conditional GAN framework. Inevitably, the transformation quality of the trained generator is degraded when the domain gap between the source domain and the target domain is large. To prevent the degraded transformation quality due to the large domain gap, we introduce *Progressive Transformation Learning (PTL)*, which progressively and iteratively expands the training set with a

subset of virtual images carefully selected to avoid the large domain gap. PTL goes through three steps for each iteration (Fig 3.2): i) sampling a subset of virtual images from a virtual image pool by giving heavier weights to images close to the current training set (transformation candidate selection), ii) transforming the selected images to be realistic (virtual2real transformation), and iii) adding the transformed images to the training set while excluding the selected images from the virtual image pool (set update). The details of each step are described next.

Transformation Candidate Selection. When selecting transformation candidates, we must consider two conflicting claims simultaneously: i) to suit the purpose of PTL, virtual images with a small domain gap should be selected, but ii) to suit the purpose of expanding the training set, virtual images with appearances that rarely appear in the training set, which usually implies a large domain gap, should also be selected.

To jointly consider these two claims, we use weighted random sampling. The sampling weight takes the exponential term which gives higher weights to virtual images with smaller domain gaps, while introducing one hyper-parameter τ to control the amplitude of the weights, as follows:

$$w(x) = \exp\left(-\frac{d(x)}{\tau}\right), \tag{3.5}$$

where $d(x)$ is the Mahalanobis distance, which is used to measure the domain gap of x from the current training set (Eq. 3.4). Intuitively, using a small τ allows a more frequent selection of images with smaller domain gaps by giving them larger weights than using a large τ. (We use τ=5.0 throughout all experiments.)

In practice, transformation candidates are selected from virtual image pool through the following four steps: i) training the human detector f_D^t on the current training set of real images R^t, ii) calculating

μ^t and Σ^t on R^t as in Eq. 3.2, iii) calculating weights for each image in the current set of virtual images V^t as in Eq. 3.5, and iv) applying weighted random sampling to V^t to select a pre-defined number n of transformation candidates. (We use n=100 throughout all experiments.)

Virtual2Real Transformation. In line with the goal of this chapter to obtain a human detector that can identify humans with diverse appearances captured by a UAV, we design the virtual2real transformation to focus on the person region rather than the background of the selected virtual images. To do so, we crop the person region in the virtual image, apply the transformation only to this region, and segment the transformed person back to the original background. For accurate segmentation, the pixel-wise segmentation mask is required. Obtaining such pixel-wise segmentation mask at no cost is another benefit of using virtual images.

The conditional GAN framework [82], in which the generator is trained to transform a given input image from source styles into target styles, is widely used to transform virtual images to look like real images [80, 119]. Among many variants of conditional GANs, we use CycleGAN [120] in which the generator is trained to minimize the reconstruction error between the input image and the reconstructed image transformed back to the original style of the input image after the initial transformation to the target style. It is shown in [82] that the transformation with CycleGAN is likely to maintain the original object pose while changing detailed styles such as patterns (e.g., transforming a white horse into a zebra in the same pose). We intend to borrow this characteristic of CycleGAN to transform virtual images in the direction that makes the detailed styles realistic while maintaining the overall human appearances, which depend on various viewing angles or human poses.

In practice, the virtual2real transformation generator f_G^t is trained using the CycleGAN framework by treating the selected transformation candidates $\mathrm{C_V}^t$ and the current set of real images R^t as

"source" and "target", respectively. Then, $C_V{}^t$ are transformed to realistic transformed images $C_R{}^t$ by applying the virtual2real transformation generator.

Set Update. After the transformed images $C_R{}^t$ are acquired from the selected transformation candidates $C_V{}^t$, PTL updates the current real image set R^t and the current virtual image set V^t as follows:

$$R^{t+1} = R^t \cup C_R{}^t \qquad V^{t+1} = V^t / C_V{}^t \tag{3.6}$$

When this progressive learning is terminated, the final human detection model can be acquired by training on the final set of real images.

In practice, the first two steps of PTL are applied to the tightly cropped image region of human region, but in the 'set update' step, the entire image including the human region and the background is added to the training set for training the human detector. More precisely, when training the virtual2real transformation generator, the tightly cropped image region around each human from the training images are used as the "target".

3.4 Experiments

3.4.1 Experimental Setup

Datasets and Evaluation Metrics. We perform experiments on three real UAV-based datasets, VisDrone [9], Okutama-Action [10], and ICG [11], and a virtual UAV-based dataset, *Archangel-Synthetic* (Sec. 2.3.2), all of which include human instances. We use a subset of *Archangel-Synthetic* consisting of various virtual characters with different poses across a range of altitudes and circle radii, along with different camera pitch angles (i.e., 17.3K images with eight characters, three poses, ten altitudes,

six circle radii, and twelve camera pitch angles). Each image in *Archangel-Synthetic* accompanies metadata about the above imaging conditions, allowing us to analyze how the feature distributions of virtual characters evolve with respect to these imaging conditions when PTL progresses. The details of the datasets used in the experiments are described in Sec. 3.6.2. We use AP@.5 and AP@[.5:.95] as evaluation metrics for all experiments.

Detector. For the detector, we use RetinaNet [65] that uses the feature pyramid network (FPN) to provide a rich multi-scale feature pyramid and processes features at all scale levels with the same subnetwork responsible for the final classification and bounding box regression. It is important to use the same subnetwork across all scale levels since the domain gap for each virtual image should be measured in a shared representation space regardless of the image size. Most other object detectors using FPN (e.g., SSD [121] and v4 or later versions of YOLO [25, 64, 122, 123]) use different subnetworks at each scale level. However, PTL is not structurally limited to RetinaNet as it can be used with any detector with minor modifications, such as adding one shared layer across all scale levels. The architectural details of the generator and discriminator of CycleGAN, along with the detector, are described in Sec. 3.6.2.

3.4.2 Properties of Progressive Learning

Which Virtual Images Are Selected for Each PTL Iteration? The top row of Figure 3.3 shows the change in the accumulated distribution of virtual images added to the training set via each PTL iteration with respect to camera locations. By examining the distribution, we can identify which camera locations of the virtual images contribute more to the training set at each PTL iteration.

It is observed that after the 1st PTL iteration, most virtual images included in the training set

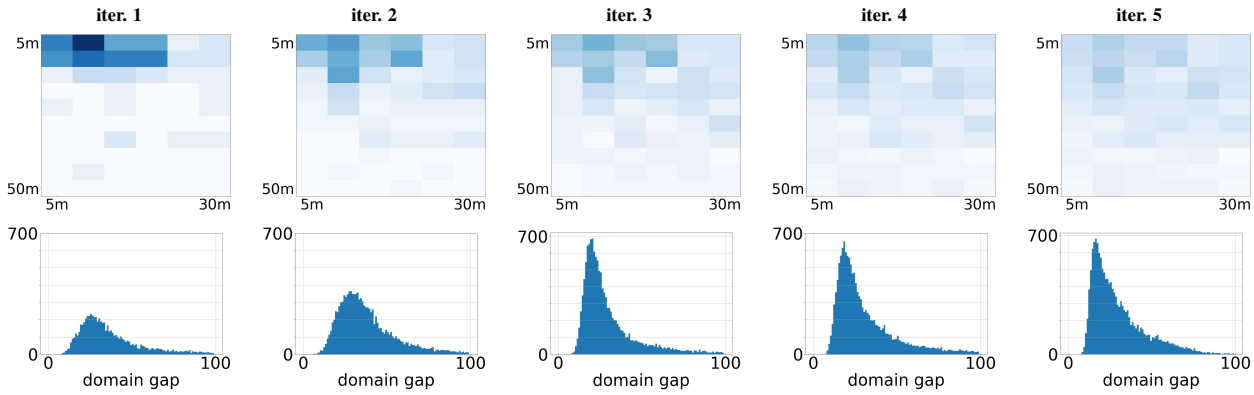


Figure 3.3: Analysis of the use of virtual images during PTL. The top row shows the accumulated distribution of transformation candidates with respect to camera locations (rotation circle radius and altitude relative to the target human along x- and y-axes, respectively) for each PTL iteration. Darker bins indicate a higher number of selected virtual images. The bottom row (x-axis: domain gap; y-axis: number of virtual images) shows the distribution of virtual images across domain gap values, where the domain gap is measured by Eq. 3.4. Results are based on experiments using 50 real images from the VisDrone dataset for training.

are taken from the camera locations close to the human subjects. As PTL progresses, the camera locations of the virtual images included in the training set gradually spread across the UAV altitudes and rotation circle radii. Consequently, after the 5th PTL iteration, transformed virtual images with diverse appearances from much broader camera locations are included in the final training set. This demonstrates that the proposed transformation candidate selection process is adequately designed to consider the two conflicting claims together.

How Close Does the Domain Gap Get as PTL Progresses? The bottom row of Figure 3.3 shows the domain gap between the virtual images and the training set at each PTL iteration. We can observe the domain gap distribution of virtual images to the training set gradually becomes narrower and smaller. Additionally, some virtual images which have not been included in the training set also appear in the long tail of the distribution.

Accuracy Variation as PTL Evolves. Figure 3.4 shows how the accuracy changes as PTL progresses

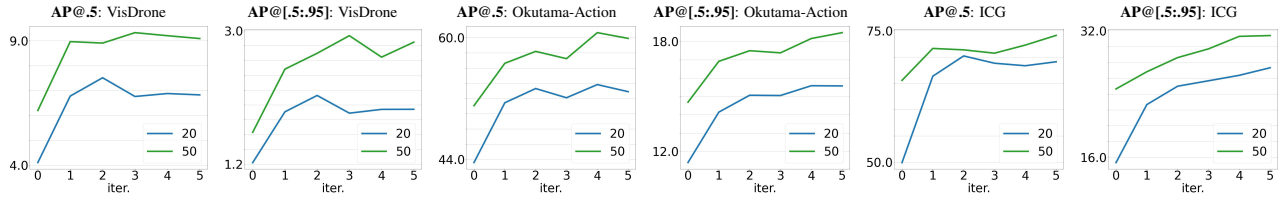


Figure 3.4: Learning curves of AP@.5 and AP@[.5:.95] on the three datasets.

on the three datasets. Overall, for AP@.5, accuracy increases rapidly until the 3rd iteration and does not change significantly thereafter. On the other hand, for AP@[.5:.95], accuracy continues to increase even after the 3rd iteration on the Okutama-Action and ICG datasets. This can be interpreted such that human bounding boxes are estimated more accurately as PTL progresses on these two datasets.

3.4.3 Results on Low-Shot Learning

Baselines. We compare PTL with the method utilizing only real images for training (i.e., 'baseline') and other three methods also leveraging virtual images in conjunction with real images for training (i.e., 'pretrain-finetune', 'naive merge', and 'naive merge w/ transform') in terms of human detection accuracy. 'Pretrain-finetune' is the strategy to train a model by pre-training on virtual images and then fine-tuning on real images, which is the most widely used approach leveraging virtual images in previous work [79, 85, 86, 87, 88, 89, 90, 91, 92, 93]. 'Naive merge' is the strategy that uses a training set naively merging from real and virtual images for model training, which has also been used in previous works [96, 97]. 'Naive merge w/ transform' naively adds transformed virtual images to the training set, where the transformation generator is trained with the CycleGAN framework by considering all virtual images and all real images as "source" and "target".

Main Results. In Table 3.1, we compare PTL to the baselines in terms of human detection accuracy in two low-shot detection regimes (i.e., 20 and 50 real images are used for training) on the three real-

world UAV-based datasets. Low-shot detection is a suitable task to validate the proper use of virtual images as notable effects can be expected from adequately expanded datasets.

In all cases, the previous methods leveraging virtual images do not present significantly better, or even worse, accuracy than their counterpart (i.e., 'baseline') using real images only for training. 'Pretrain-finetune' is the only method, except for PTL, that presents better accuracy than the baseline in some cases. This two-step method is effective in avoiding adverse effects due to the large domain gap while indirectly taking advantage of the large-scale dataset. However, the increase in accuracy is marginal as the task-specific properties (i.e., human detection) of the large-scale virtual dataset used for pre-training are not fully exploited due to the catastrophic forgetting issue inherent in this indirect method. In addition, 'naive merge w/ transformation', which is the only previous method that uses transformed virtual images, presents significantly reduced accuracy, which can be regarded as the adverse effects (e.g., transformation quality degradation) when the large domain gap is not properly addressed. The final model obtained after the 5th PTL iteration or the best model achieved when PTL progresses consistently presents significantly better performance than any compared method. This demonstrates *the effectiveness of PTL such that accuracy is substantially improved by expanding the training set using virtual images while the large domain gap is appropriately addressed*.

Ablation: The Effect of τ. In Table 3.2a, we compare five different τ values to investigate the effect of τ used to control the sampling weights when selecting transformation candidates (Eq. 3.5). It is found that as large τ values (≥ 100) are used, the accuracy on the VisDrone dataset decreases while the accuracy on the ICG dataset increases. The finding indicates that adding more virtual images with a large domain gap to the training set using a large τ has no significant effect or even adverse effect in terms of accuracy when the training set and the test set are in the same domain. However, a large

Table 3.1: Low-shot learning accuracy using 20 and 50 real images. AP@.5 and AP@[.5:.95] are reported in each bin. For PTL, the margin relative to the baseline accuracy is shown below the reported accuracy. The best accuracy for each setting is highlighted in bold. R and V denote real and virtual image sets, respectively.

method	train set	VisDrone		Okutama-Action		ICG	
		20	50	20	50	20	50
baseline	R	3.74/ 1.09	6.42/ 1.86	41.61/ 11.23	49.84/ 13.76	49.35/ 14.69	66.75/ 23.91
pretrain-finetune	R+V	4.99/ 1.46	6.25/ 1.99	44.57/ 12.78	49.06/ 15.08	66.92/ 26.67	68.41/ 29.73
naive merge	R+V	3.41/ 1.02	5.18/ 1.65	34.26/ 9.21	48.33/ 14.61	55.95/ 20.76	65.68/ 26.73
w/ transform	R+V	1.26/ 0.49	4.02/ 1.37	27.37/ 7.84	41.36/ 12.64	48.02/ 17.62	65.03/ 27.21
PTL (5th itr.)	R+V	6.83/ 1.94	9.09/ 2.85	52.89/ 15.57	59.90/ **18.48**	69.11/ **27.33**	**74.14/ 31.41**
		+3.09/+0.85	+2.67/+0.99	+11.28/ +4.34	+10.06/ +4.72	+19.76/+12.64	+7.39/ +7.50
PTL (best)	R+V	**7.52/ 2.13**	**9.33/ 2.94**	**53.82/ 15.59**	**60.65/ 18.48**	**70.23/ 27.33**	**74.14/ 31.41**
		+3.78/+1.04	+2.91/+1.08	+12.21/ +4.36	+10.81/ +4.72	+20.88/+12.64	+7.39/ +7.50

Table 3.2: PTL ablation experiments. Models are trained on the VisDrone dataset using the 50-shot setup.

(a) Varying τ in PTL (after 5th iteration).

τ	VisDrone	Okutama-Action	ICG
1	9.48/ 3.01	39.37/ 10.45	27.87/ 7.75
5	9.09/ 2.85	**42.39/ 11.41**	29.26/ 7.27
10	**9.68/ 2.87**	37.48/ 9.51	33.06/ 7.66
100	8.97/ 2.54	38.25/ 10.18	33.78/ 9.29
1000	8.90/ 2.63	39.15/ 10.00	**43.90/ 11.97**

(b) Varying # of virtual images added to the training set per PTL iteration. The reported accuracies are obtained by using the best PTL models.

# img per itr	VisDrone	Okutama-Action	ICG
50	**9.59/** 2.90	41.62/ 11.19	25.94/ 6.04
100	9.33/ **2.94**	**42.39/ 11.46**	30.01/ 7.36
200	9.04/ 2.91	41.29/ 11.18	**35.50/ 10.20**

τ shows a remarkable effect in the cross-domain setting, especially when the training set and the test set have very different characteristics, such as the VisDrone train set and the ICG test set. We use τ=5 throughout the experiment for the same-domain setup, but we can also consider using a large τ for the cross-domain setup.

Ablation: The Effect of n. n, which is the number of virtual images added to the training set per PTL iteration, can affect not only the optimality of training but also the scalability of models trained by PTL toward other datasets different from the real dataset used for training. Accordingly, to find the optimal value of n by taking these effects into account, we carry out ablation experiments under the cross-domain setup as shown in Table 3.2b.

We compare three cases when n is 50, 100, and 200. When the training set and the test set comes from the same domain (i.e., training and testing on the VisDrone dataset), the human detection

accuracy is not very sensitive to n. The best accuracy is obtained with the fewest virtual images (i.e., n=50). The insensitivity to the hyperparameter n is also observed in the Okutama-Action dataset, having relatively similar properties to the VisDrone dataset under the cross-domain setup. However, for ICG, which is considered to have noticeably different properties from the VisDrone dataset under the cross-domain setup, the accuracy is significantly improved as more virtual images (i.e., n=200) are used per PTL iteration. This implies that using more virtual images through more PTL iterations tends to further reduce the cross-domain gap when the discrepancy between the training set and the test set is considerably large, such as VisDrone vs. ICG, resulting in improved PTL scalability. Considering the trade-off between the training optimality and the PTL scalability, we use n=100 throughout all other experiments in this chapter.

Qualitative Analysis of Transformation. Figures 3.5 show several samples of transformed virtual images included in the training set using methods with virtual2real transformation (i.e., PTL and 'naive merge w/ transformation') in the 'VisDrone and 50-shot' setup. First, the quality of the transformed image by 'naive merge w/transformation' generally deteriorates a lot, and there are samples, in some severe cases, in which it is difficult to recognize the human appearance anymore. On the other hand, in the case of the virtual image transformed by PTL considering the domain gap when training the transformation generator, it can be seen that only the pattern is changed to be similar to the image of the real dataset while the human pose is maintained well. This qualitative analysis supports the validity of our claim that *the domain gap between virtual images and real images should be considered when training the virtual2real transformation generator*.



Naive merge w/ transform PTL

Figure 3.5: Sample Virtual2Real transformation results (VisDrone, 50-shot). Each set consists of three images: original virtual image (left), transformed image (middle), and transformed image with background (right).

3.4.4 Results on Cross-Domain Detection

In Table 3.3, we show the accuracy for the cross-domain setups on the three datasets. The 'cross-domain detection' setups are used to validate the impact of using virtual images for training when training and test images have distinct characteristics, such as when the distributions of human appearances with respect to human poses and viewing angles are different.

First, it is observed that using PTL yields much better accuracies than the baseline when using the same real training dataset. Despite the inherent difficulty of cross-domain learning, it is also observed that leveraging virtual images through PTL produces results not far behind the accuracy of the baseline (shown in the first row of Table 3.3) which uses the training images from the same domain as the test images. However, the improvement in the cross-domain detection accuracy when using the ICG dataset for training is relatively low compared to other cases. This is because the ICG dataset has very different characteristics from other datasets, and virtual images added to the training set do not reduce this difference. Note that if the real images in the initial training set have very distinct appearances, a very different set of virtual images may be selected in the first PTL iteration. Thus, the disparity may not be overcome even if the PTL progresses further. Nevertheless, in general, we can confirm that *human detectors trained using virtual images through PTL can improve substantially*

Table 3.3: Cross-domain detection accuracy under 3×3 cross-domain setups. For each setup, the datasets on the left and right of the arrow indicate training and test sets, respectively. PTL and baseline accuracies (without virtual images) are reported. The top row corresponds to same-domain setups and serves as reference performance for cross-domain results. All setups within each column use the same test dataset and low-shot regime.

method	VisDrone				Okutama-Action				ICG			
	20		50		20		50		20		50	
	Vis → *Vis*				*Oku* → *Oku*				*ICG* → *ICG*			
baseline	3.74/	1.09	6.42/	1.86	41.61/	11.23	49.84/	13.76	49.35/	14.69	66.75/	23.91
PTL (5th itr.)	6.83/	1.94	9.09/	2.85	52.89/	15.57	59.90/	18.48	69.11/	27.33	74.14/	31.41
PTL (best)	7.52/	2.13	9.33/	2.94	53.82/	15.59	60.65/	18.48	70.23/	27.33	74.14/	31.41
	Oku → *Vis*				*Vis* → *Oku*				*Vis* → *ICG*			
baseline	1.62/	0.47	2.04/	0.57	17.13/	4.53	36.82/	9.87	2.92/	0.56	7.46/	1.83
PTL (5th itr.)	2.72/	0.94	3.05/	1.07	30.72/	7.45	42.39/	11.41	26.86/	7.22	29.26/	7.27
PTL (best)	3.00/	1.22	3.56/	1.17	33.25/	8.59	42.39/	11.46	29.60/	7.69	30.01/	7.36
	ICG → *Vis*				*ICG* → *Oku*				*Oku* → *ICG*			
baseline	0.54/	0.13	0.99/	0.26	3.56/	0.75	10.27/	2.49	5.37/	1.25	5.23/	1.20
PTL (5th itr.)	1.09/	0.33	1.61/	0.50	11.19/	2.58	14.20/	3.56	28.98/	8.14	25.39/	6.53
PTL (best)	1.58/	1.02	1.70/	0.63	12.82/	2.96	14.20/	3.71	28.98/	8.14	26.62/	6.53

detection accuracy, regardless of which real dataset is used during training.

3.5 Discussion

Our method has been proved effective in leveraging virtual images during training as it presents much better accuracy than any other previous methods for low-shot learning tasks, where scaling up the training dataset can significantly impact. In addition, compared to the baseline which does not use virtual images, our method also presents remarkable accuracy on cross-domain detection, where the real training and test datasets are from two distinct domains with very different characteristics.

Despite the merits of the proposed method, there is still room to improve PTL further. Specifically, the current version of PTL can only leverage a subset of the entire virtual images within a limited number of iterations, beyond which the accuracy may decrease. We hope that more advanced methods can be developed to address this issue so that PTL can progress for more iterations with a continuous accuracy increase until all virtual images are leveraged for training.

3.6 Additional Details

3.6.1 Modeling Representation Space of a Detector

Mathematical Derivation. In this section, we derive the earlier formulation that models the representation space of a general object detector with a multivariate Gaussian distribution. Let the input random variable and its corresponding label for a linear classifier be denoted by $x \in \mathcal{X}$ and $y = \{y_c\}_{c=1,\cdots,C} \in \mathcal{Y}$, respectively, where $y_c \in \{0, 1\}$. The posterior distribution defined by the linear classifier, whose output is given by the sigmoid function, can be expressed as:

$$\begin{aligned} P(y_c = 1|x) &= \frac{1}{1 + \exp\left(-w_c x - b_c\right)} \\ &= \frac{\exp\left(w_c x + b_c\right)}{\exp\left(w_c x + b_c\right) + 1}, \end{aligned} \tag{3.7}$$

where w_c and b_c denote the weights and bias of the linear classifier for category c, respectively.

Gaussian Discriminant Analysis (GDA) models the posterior distribution of the classifier by assuming that the class-conditional distribution $P(x|y)$ and the class prior distribution $P(y)$ follow multivariate Gaussian and Bernoulli distributions, respectively:

$$\begin{aligned} P(x|y_c = 0) &= \mathcal{N}(\mu_0, \Sigma_0), \\ P(x|y_c = 1) &= \mathcal{N}(\mu_1, \Sigma_1), \\ P(y_c = 0) &= \beta_0 / \left(\beta_0 + \beta_1\right), \\ P(y_c = 1) &= \beta_1 / \left(\beta_0 + \beta_1\right), \end{aligned} \tag{3.8}$$

where $\mu_{\{0,1\}}$ and $\Sigma_{\{0,1\}}$ denote the mean and covariance of the multivariate Gaussian distributions, respectively, and $\beta_{\{0,1\}}$ represent the unnormalized priors for the background and category c.

For the special case of GDA where all categories share the same covariance matrix (i.e., $\Sigma_0 = \Sigma_1 = \Sigma_c$), known as Linear Discriminant Analysis (LDA), the posterior distribution $P(y_c|x)$ can be written in terms of $P(x|y_c)$ and $P(y_c)$ as:

$$
\begin{aligned}
&P(y_c = 1|x) \\
&= \frac{P(y_c = 1)P(x|y_c = 1)}{P(y_c = 0)P(x|y_c = 0) + P(y_c = 1)P(x|y_c = 1)} \\
&= \frac{\begin{array}{l}\exp\Big((\mu_1 - \mu_0)^\top \Sigma_c^{-1} x \ldots \\ \quad -\frac{1}{2}\mu_1^\top \Sigma_c^{-1}\mu_1 + \frac{1}{2}\mu_0^\top \Sigma_c^{-1}\mu_0 + \log \beta_1/\beta_0\Big)\end{array}}{\begin{array}{l}\exp\Big((\mu_1 - \mu_0)^\top \Sigma_c^{-1} x \ldots \\ \quad -\frac{1}{2}\mu_1^\top \Sigma_c^{-1}\mu_1 + \frac{1}{2}\mu_0^\top \Sigma_c^{-1}\mu_0 + \log \beta_1/\beta_0\Big) + 1\end{array}}.
\end{aligned}
\tag{3.9}
$$

The quadratic terms cancel under the shared covariance matrix assumption. Consequently, the posterior derived from GDA (Eq. 3.9) is equivalent to the sigmoid-based posterior of the linear classifier in Eq. 3.7, with $w_c = (\mu_1 - \mu_0)^\top \Sigma_c^{-1}$ and $b_c = -\frac{1}{2}\mu_1^\top \Sigma_c^{-1}\mu_1 + \frac{1}{2}\mu_0^\top \Sigma_c^{-1}\mu_0 + \log \beta_1/\beta_0$. This equivalence implies that the representation space induced by x can be modeled by a multivariate Gaussian distribution.

Based on the above derivation, let x denote the output of the penultimate layer of an object detector for a region proposal, and let a linear classifier parameterized by w_c and b_c serve as the final layer. Then, the representation space of the object detector corresponding to category c can be modeled

by a multivariate Gaussian distribution. In other words, it can be characterized by the parameters μ_1 (i.e., μ_c) and Σ_c.

Relation to Prior Work. The sigmoid function can be viewed as a special case of the softmax function for a single category, as both take the form of an exponential term for the category of interest normalized by the sum of exponential terms over all categories. Therefore, the modeling of a sigmoid-based detector can be derived from prior work [83], which shows that a softmax-based classifier can be modeled with a multivariate Gaussian distribution in the representation space. However, our derivation remains meaningful, as it extends this modeling framework, previously limited to softmax-based classifiers, to general object detectors based on sigmoid outputs. This distinction is important because most object detectors, especially one-stage detectors, employ the sigmoid function, which treat each category independently and allow multiple categories to be active simultaneously.

3.6.2 Implementation Details

Multi-Scale Training. We adopt a multi-scale training strategy when preparing input images, in addition to the 5-level feature scaling provided by the detector's FPN module. This aims to improve robustness to variations in human scale across images. Due to the significant differences in human sizes between the real and virtual datasets, we apply dataset-specific scaling factors to normalize object scales after resizing. For the real dataset, the short side of each input image is randomly resized to one of $\{768, 800, 832, 864\}$, while the long side is constrained to be at most 1440 pixels. For the virtual dataset, images are resized to fixed square resolutions in $\{128, 256, 384, 512\}^2$.

Network Architecture. We adopt CycleGAN to obtain the generator, which involves two generators and two discriminators during training. Each generator follows a 24-layer UNet-like architecture (i.e.,

the `resnet_9blocks` generator in the official CycleGAN repository [124]), consisting of nine 2-layer residual modules that connect the encoder and decoder. Each discriminator is implemented as a 5-layer fully convolutional network (i.e., the `basic` discriminator in [124]).

For the detector, we adopt the official RetinaNet architecture implemented in Detectron2 [125], with several modifications. First, the feature dimension of both the classification and box regression subnets is reduced from 256 to 32, as we consider a single category (human) instead of multiple categories (e.g., 80 categories in the MS COCO dataset [36]). Second, we replace ReLU with LeakyReLU in both subnets to mitigate the singular covariance issue that may arise when computing the Mahalanobis distance, which can be occasionally triggered by the dying ReLU problem. Third, the kernel size of the final convolutional layer in the classification subnet (i.e., the layer preceding the sigmoid activation) is reduced from 3×3 to 1×1, ensuring that each prediction corresponds to a single feature vector in the representation space. We use ResNet-50 as the backbone.

Training Details. The training settings for the human detector in our method and the baseline approaches are summarized in Table 3.4a. Without further specification, we follow the training settings and initialization strategies of the original RetinaNet [65]. In the pretrain–finetune paradigm, fine-tuning is typically performed with fewer iterations and a smaller learning rate, as convergence is faster. However, we use the same learning rate (0.001) for both stages, since the ImageNet-pretrained backbone is also fine-tuned on virtual images during the pre-training stage.

The training settings for the generator, used in both the baseline with transformation (i.e., 'naive merge w/ transformation') and our method, are listed in Table 3.4b. These settings and initialization strategies follow the original CycleGAN training procedure [120]. The 'naive merge w/ transform' approach uses fewer training epochs than PTL (80 vs. 100), as all virtual images are utilized during

Table 3.4: Training configurations used in the experiments.

(a) Detector Training

config	baseline	pretrain-finetune		naive merge	PTL
		pretrain	finetune		
optimizer	SGD				
momentum	0.9				
weight decay	0.0001				
base lr	0.001				
lr schedule	multi-step lr				
gamma	0.1				
warmup iter.	1000				
total iter.	6000	6000	600	6000	6000
steps	5000	5000	500	5000	5000
batch size	16				
filter empty annot.	False				
box threshold	10				
aspect ratio grouping	False				

(b) Generator Training

config	naive merge w/ transform	PTL
optimizer	Adam	
momentum	β_1, β_2 = 0.5, 0.999	
lr	0.0002	
total epochs	80	100
batch size	8	
load size	256	
preprocess	None	

training. In practice, as dataset size increases, fewer epochs are typically sufficient for convergence.

3.6.3 Dataset Details

In this section, we provide the details of the three real UAV-based datasets—VisDrone [9], Okutama-Action [10], and ICG [11]—as well as the virtual UAV-based dataset, *Archangel-Synthetic*, used in Chapter 3. Sample images from each dataset are shown in Figures 3.6 and 3.7.

VisDrone. The VisDrone dataset consists of four tracks: object detection (DET), video object detection (VID), single object tracking (SOT), and multi-object tracking (MOT), each with a dedicated dataset. We use the DET track and focus on detecting the person and pedestrian categories among the ten object categories provided. The DET track contains 10,209 images, split into 6,471 `training` images, 548 `validation` images, 1,580 `test-challenge` images, and 1,610 `test-dev` images. We use the `training` set for model training and the `test-dev` set for evaluation. The maximal image resolution in the VisDrone dataset is 2000×1500.

Okutama-Action. The Okutama-Action dataset was originally created for human action detection



VisDrone



Okutama-Action



ICG

Figure 3.6: Sample images from each real dataset.

from aerial views. In addition to action detection, it also supports pedestrian detection as a sub-task. The dataset was collected using a drone equipped with a camera flying at altitudes between $10m$ and $45m$, with camera angles set to 45 or 90 degrees. It contains 43 video sequences in 4K resolution (3840×2160), where 33 sequences are used for the `train-val` set, and the remaining 10 sequences are used for the `test` set. Due to the high similarity between consecutive frames, we sample every tenth frame from both the `train-val` set and the `test` set. The model is trained on the `train-val` set and evaluated on the `test` set.

ICG. The ICG dataset is designed for studying semantic understanding of urban scenes and includes ground-truth human bounding box annotations for detection tasks. Images were captured from a camera positioned at altitudes from $5m$ to $50m$, with a resolution of 6000×4000. The dataset provides a

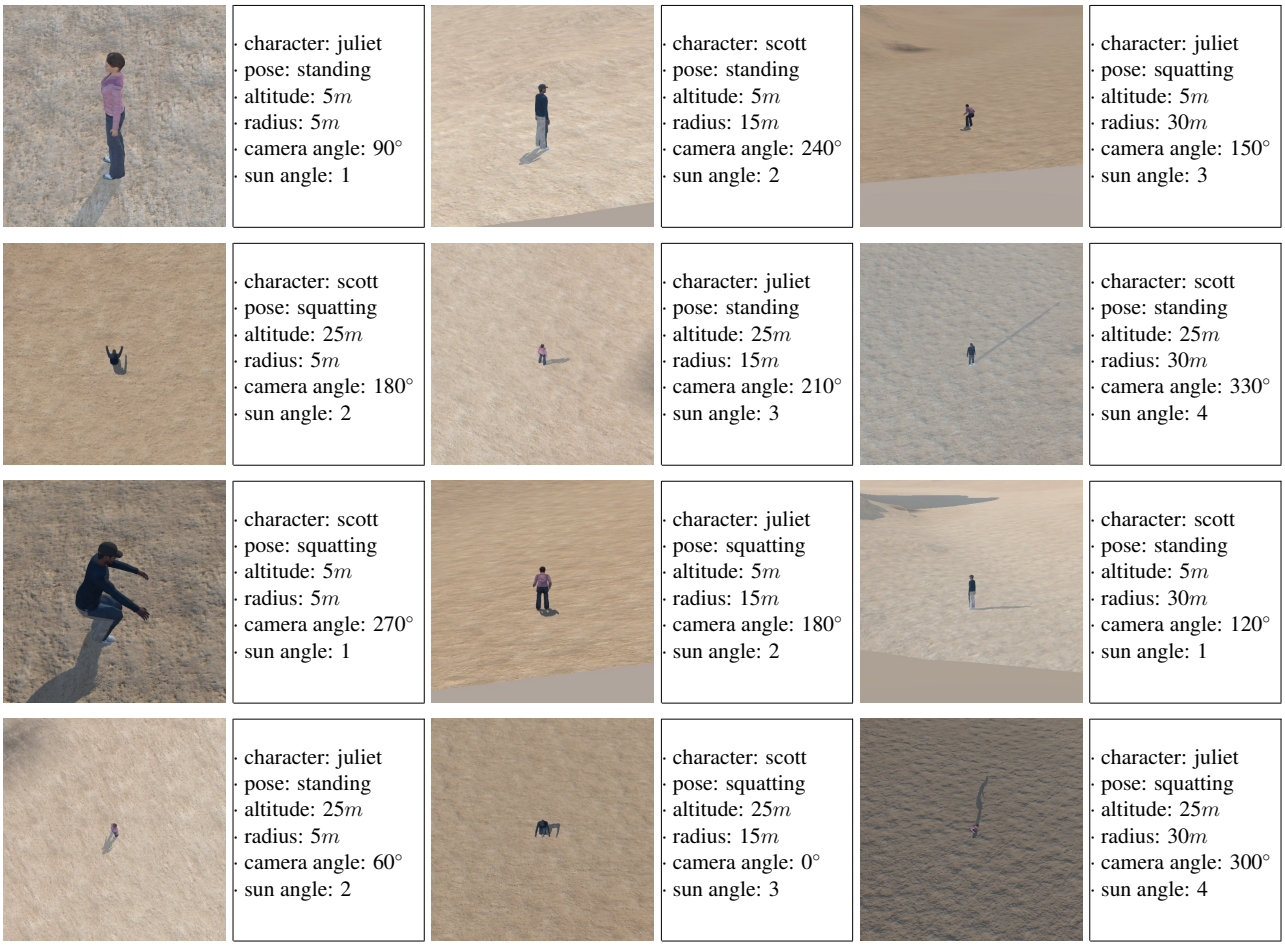


Figure 3.7: Sample images from the *Archangel-Synthetic* dataset. Metadata for each image is shown on the right.

`training` set of 400 images for human detection. We use the first 320 images for training and the remaining 80 images for testing.

Archangel-Synthetic. This dataset is one of the three sub-datasets of the *Archangel* dataset (see Sec. 2.3.2), alongside *Archangel-Real* and *Archangel-Mannequin*. The *Archangel* dataset is a hybrid UAV-based dataset captured under similar imaging conditions across real and synthetic domains. An important property that sets it apart from other datasets is the availability of metadata describing camera positions, including UAV altitudes and rotation radii for each image. The *Archangel-Synthetic* dataset contains 4.4M images. Due to its significantly larger size compared to the utilized real datasets,

Table 3.5: Wall-clock training time (minutes) for the VisDrone 20-shot setting. Experiments are conducted using GeForce RTX 2080 Ti GPUs.

method	total	detail
baseline	40	
pretrain-finetune	21	4/17 (pretrain / finetune)
naive merge	17	
w/ transform	2,777	2,760/17 (GAN train / Dtr train)
PTL	600	20/iter. (domain gap calc. on virtual set)
		40/36/32/28/25/22 (Dtr train, $\sim 6^{th}$ iter.)
		28/41/56/69/83 (GAN train, $\sim 5^{th}$ iter.)

a subset of 17.3K images is used as the virtual image set in our experiments.

3.6.4 Additional Analyses

Training Time Comparison. The training times for PTL and the baseline methods are reported in Table 3.5. For PTL, the detector training time (denoted as 'Dtr train' in the table) decreases as training progresses. This trend arises because the number of virtual images (i.e., *Archangel-Synthetic*), which typically have smaller resolutions than real images (i.e., *VisDrone*), increases over iterations. Notably, detector training in PTL uses a fixed number of iterations regardless of the PTL iteration. In contrast, the CycleGAN training time (denoted as 'GAN train') gradually increases due to the growing number of virtual images.

Although PTL is relatively slow due to CycleGAN training, it remains significantly faster than the 'naive merge w/ transform' baseline. This is because PTL uses only a subset of virtual images at each PTL iteration, rather than the entire dataset. Nevertheless, PTL may encounter scalability issues in terms of training time when a large number of virtual images are used over many iterations. To mitigate this issue, the CycleGAN training time at each PTL iteration can be reduced by fine-tuning the model from the previous iteration [126].

Table 3.6: Comparison of distance metrics (VisDrone, 20-shot).

Metric	Vis	Oku	ICG
Euclidean	5.28 /1.52	28.80 /6.83	26.00 /7.06
Mahalanobis	**6.83 /1.94**	**30.72 /7.45**	**26.86 /7.22**

Table 3.7: Comparison of transformation candidate selection strategies (VisDrone, 50-shot).

test	close 100	mid 100	far 100	our
Vis	9.11/ 2.71	9.11/ 2.89	8.64/ 2.53	**9.38/ 2.94**
Oku	39.57/11.05	**43.31/11.55**	41.98/11.32	42.39/11.47
ICG	29.66/ 7.40	**34.98/ 8.97**	31.24/ 8.19	30.01/ 7.36

Analysis of Domain Gap Measurement. To investigate the effect of using the Mahalanobis distance to measure domain gaps, we compare it with alternative metrics under the assumption that feature representations of a given category in the real dataset follow a multivariate Gaussian distribution. Specifically, we consider Euclidean distance, which depends only on the mean and ignores the covariance of the distribution. As shown in Table 3.6, the Mahalanobis distance consistently yields higher accuracy in both in-domain and cross-domain setups.

Analysis of Transformation Candidate Selection. To assess the impact of weighted random sampling for selecting transformation candidates, we compare it with several alternative selection strategies. Particularly, we conduct experiments by selecting 100 virtual images from three different domain-gap ranges (i.e., {close, mid, far}) instead of using weighted random sampling ('our' in Table 3.7) at each PTL iteration. The results show that our selection strategy achieves the best performance in the in-domain setup while maintaining competitive accuracy in the cross-domain setup.

Further Analysis of Progressive Learning Properties. In this section, we analyze the distributions of transformation candidates with respect to camera positions (Figure 3.8) and domain gaps (Figure 3.9) for five additional cases not shown in Figure 3.3. These analyses demonstrate that the observations

described in Section 3.4 generalize to these scenarios.

For the distributions of transformation candidates with respect to camera locations (Figure 3.8), the trend observed in Section 3.4 remains consistent across all cases. As PTL progresses, the camera locations of virtual images included in the training set gradually expand to cover the entire spatial range. This suggests that the transformation candidate selection strategy in PTL is robust across different experimental setups. Furthermore, the distribution of human instances in the real training set with respect to camera locations is likely similar to that of transformation candidates in the first PTL iteration, where virtual images with smaller domain gaps are more frequently selected. This observation suggests that, in the Okutama-Action dataset, human instances are captured across a wide range of camera positions. In contrast, the other two datasets exhibit more concentrated camera ranges. In particular, for the ICG dataset, most images are captured within a close range, which may explain why increasing camera diversity via PTL does not significantly improve cross-domain performance.

For the distribution of virtual images with respect to domain gaps (Figure 3.9), the narrowing trend described in Section 3.4 is also observed. However, the rate of this change is noticeably slower for the ICG dataset compared to the others, as shown in Figure 3.9. This finding further indicates that the ICG dataset has very distinct characteristics compared to the other datasets.

Qualitative Analysis of Transformation. Figures 3.10–3.15 present examples of transformed virtual images used for training in methods employing virtual2real transformation (i.e., PTL and the 'naive merge w/ transformation' baseline) across six experimental setups. The observed trends are consistent with those shown in Figure 3.5, reinforcing the qualitative superiority of PTL over the baseline approach. Furthermore, these results support our claim that explicitly accounting for the domain gap between virtual and real images is crucial when training virtual2real transformation models.

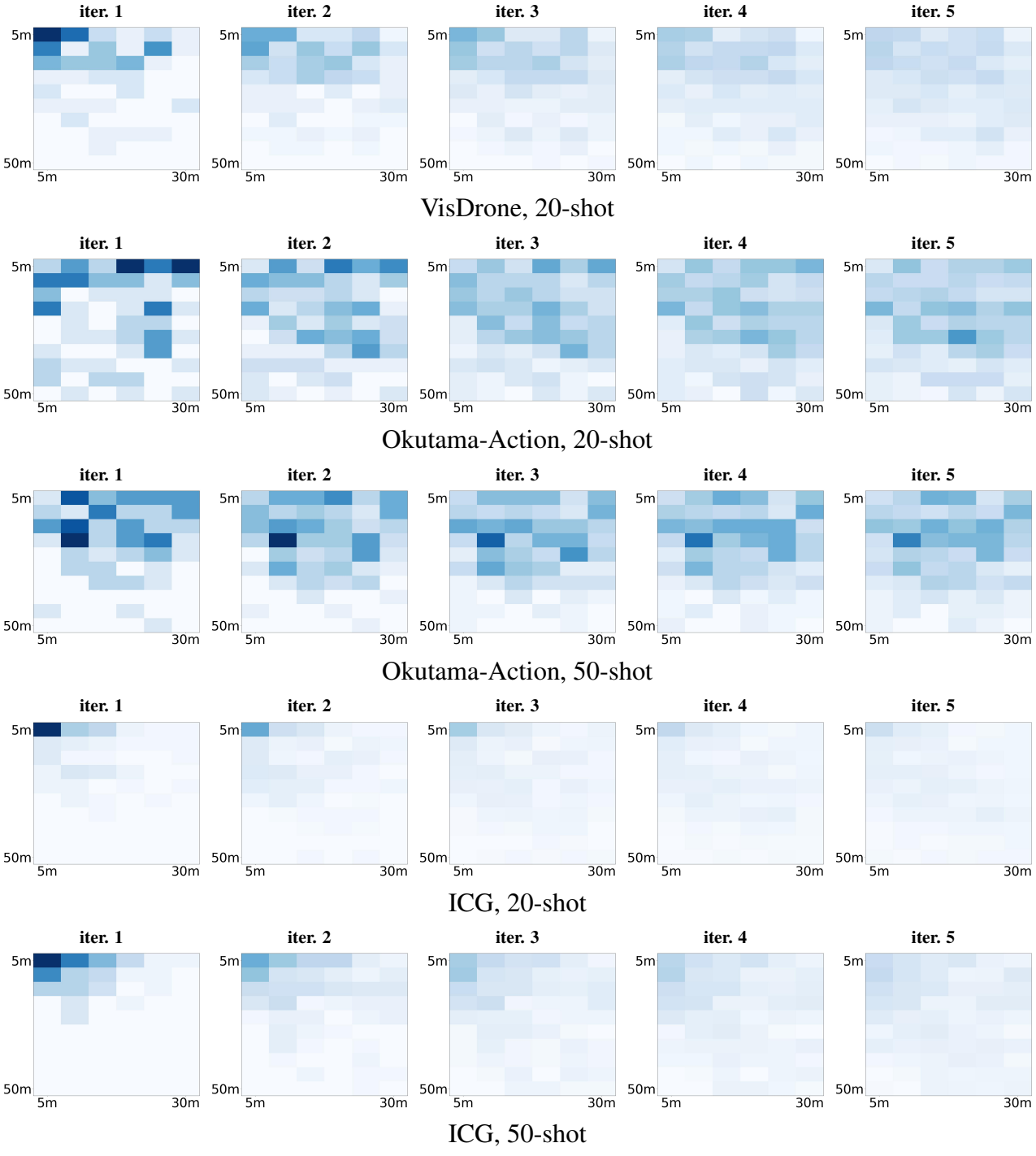


Figure 3.8: Accumulated distributions of transformation candidates with respect to camera locations in additional setups. The x- and y-axes denote rotation circle radius and altitude relative to the target human, respectively. Results are shown for all experimental setups except for the setup (VisDrone, 50-shot) presented in Figure 3.3.

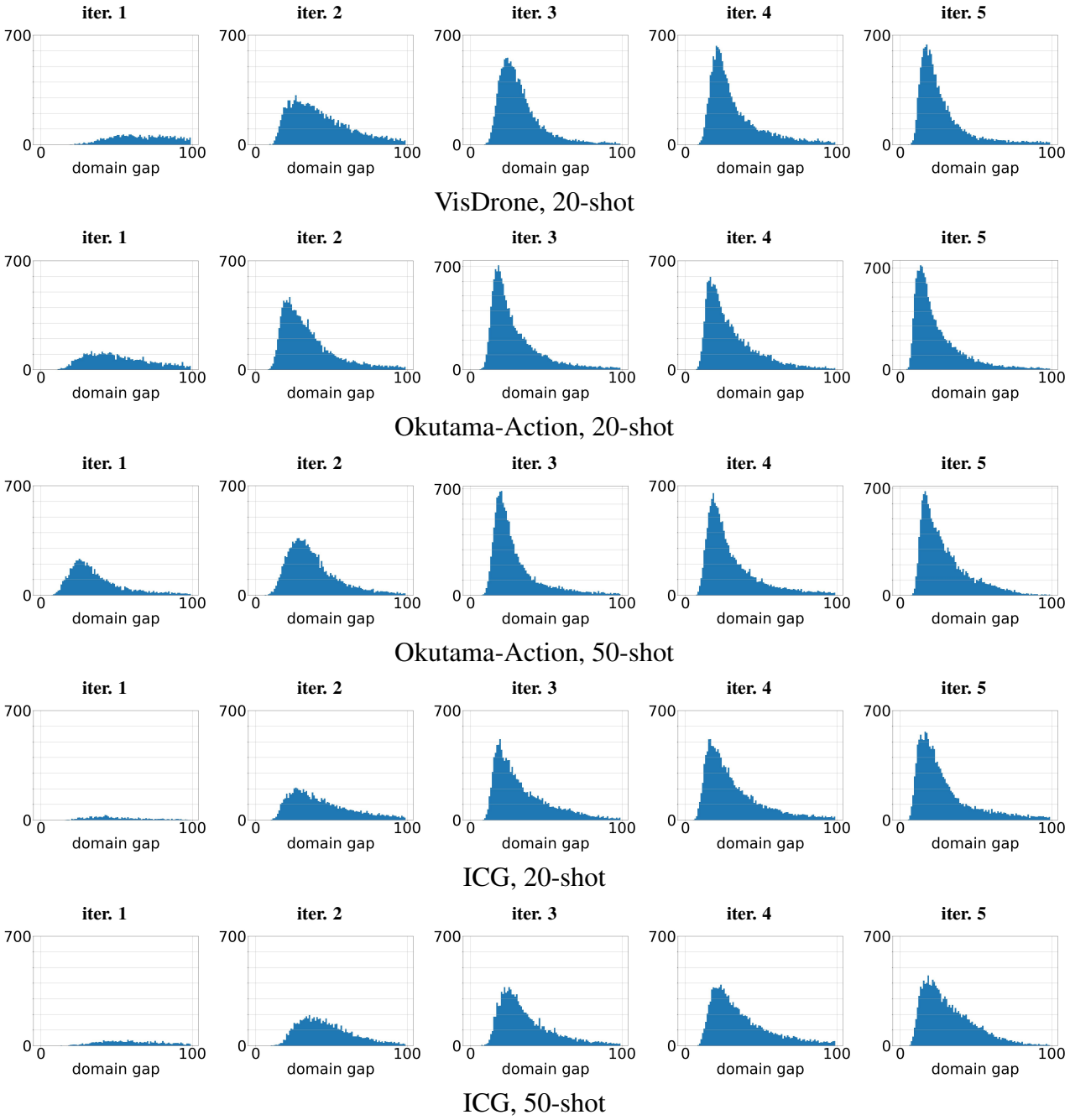


Figure 3.9: Distributions of virtual images with respect to domain gap in additional setups. The x-axis represents the domain gap and the y-axis represents the corresponding number of virtual images. Results are shown for all experimental setups except for the setup (VisDrone, 50-shot) presented in Figure 3.3.



Naive merge w/ transform PTL

Figure 3.10: Sample Virtual2Real transformation results (VisDrone, 20-shot). Each set consists of three images: original virtual image (left), transformed image (middle), and transformed image with background (right).



Naive merge w/ transform PTL

Figure 3.11: Sample Virtual2Real transformation results (VisDrone, 50-shot). Each set consists of three images: original virtual image (left), transformed image (middle), and transformed image with background (right).



Naive merge w/ transform PTL

Figure 3.12: Sample Virtual2Real transformation results (Okutama-Action, 20-shot). Each set consists of three images: original virtual image (left), transformed image (middle), and transformed image with background (right).



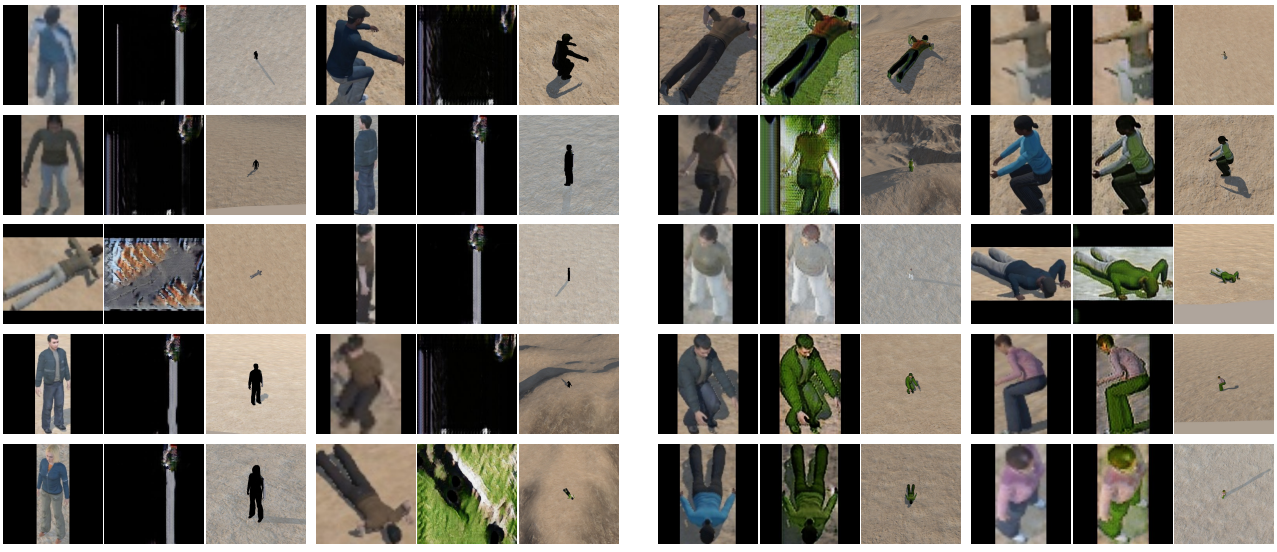
Naive merge w/ transform PTL

Figure 3.13: Sample Virtual2Real transformation results (Okutama-Action, 50-shot). Each set consists of three images: original virtual image (left), transformed image (middle), and transformed image with background (right).



Naive merge w/ transform PTL

Figure 3.14: Sample Virtual2Real transformation results (ICG, 20-shot). Each set consists of three images: original virtual image (left), transformed image (middle), and transformed image with background (right).



Naive merge w/ transform PTL

Figure 3.15: Sample Virtual2Real transformation results (ICG, 50-shot). Each set consists of three images: original virtual image (left), transformed image (middle), and transformed image with background (right).

Chapter 4: Diversifying Human Pose in Synthetic Data for Aerial-view Human Detection

4.1 Introduction

Despite the recent surge in the use of unmanned aerial vehicles (UAVs), aerial scene perception, particularly object detection in aerial images, has not progressed as much as general object detection [127]. This gap may be attributed to the lack of real-world datasets that sufficiently capture diverse object appearances, especially humans, from aerial perspectives.

Leveraging synthetic data—generated by manipulating factors related to human appearances—for training detectors has recently emerged as a promising solution [26]. Efforts have also been made to synthesize images of human postures using multiple virtual characters with varied outfits and physical characteristics [13], or by capturing them from different angles and distances [24]. However, constructing a synthetic dataset with diverse human appearances that match real-world characteristics typically requires significant efforts from experts to model characters and design motions. Human poses, in particular, are a key factor contributing to distinctive human appearances; they are highly diverse and only partially defined, making them exceptionally difficult to control. This challenge becomes even more pronounced when everything must be built from scratch, a common scenario due to the fact that existing synthetic datasets often do not share the complete assets (e.g., characters, motions, or backgrounds) used to generate them.

In this chapter, we introduce an alternative approach called SynPoseDiv—a novel framework that diversifies human poses in an existing synthetic dataset (Fig. 4.1). Specifically, SynPoseDiv transforms images in the existing dataset so that the characters assume poses not found in the original set, while maintaining the original style. To achieve this, SynPoseDiv addresses two key challenges: i)

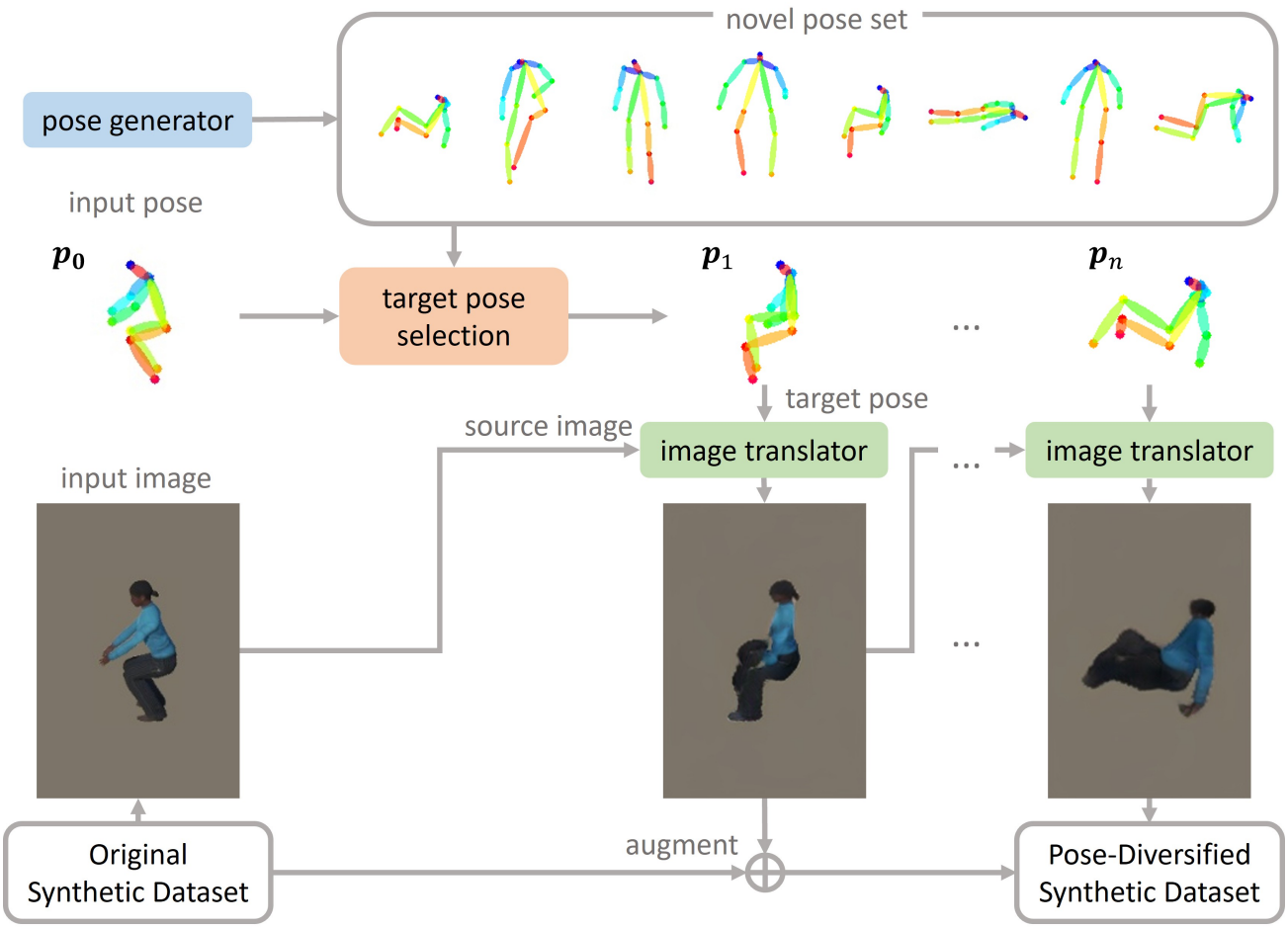


Figure 4.1: The SynPoseDiv framework generates a set of novel poses using a pose generator. For each synthetic image in the original dataset, a sequence of target poses is selected from the novel pose set through target pose selection. The image translator then generates images corresponding to each target pose in the sequence. A pose-diversified synthetic dataset is constructed by adding the generated images to the original synthetic dataset after post-processing.

how to identify a variety of novel yet realistic poses, and ii) *how to generate human images in these novel poses*.

To tackle the first challenge, SynPoseDiv incorporates a pose generator based on the diffusion model, an emerging generative model known for its remarkable performance [128]. The pose generator produces a wide range of realistic human poses in the form of 3D keypoint locations. Using this generator, SynPoseDiv constructs a large set of diverse novel poses ('novel pose set') while ensuring

diversity by filtering out highly similar poses.

To address the second challenge, SynPoseDiv employs a source-to-target image translator to create human images where virtual characters assume various poses from the novel pose set. Since target images corresponding to these novel poses are unavailable for training—which could limit the image translator's effectiveness—SynPoseDiv applies the translator to sequences of target poses ('target pose sequences') that demonstrate *incremental* body movements, gradually transforming an input pose into the desired final target pose. These target pose sequences are selected from the novel pose set through a target pose selection process, which identifies the optimal path from a starting pose to a final target pose within a graph built from the novel pose set. This process is solved using Dijkstra's algorithm [129]. For each target pose in the sequence, a corresponding target image is generated by the image translator, using the previously generated target image as the source.

We demonstrate that leveraging SynPoseDiv to diversify a dataset remarkably enhances detection performance, especially in low-shot scenarios where additional data is crucial. Moreover, regardless of the training approach (either 'pre-training and fine-tuning' or 'Progressive Transformation Learning (PTL) [26]'), the pose-diversified synthetic dataset consistently yields significantly better detection accuracy than the original synthetic dataset across three aerial-view human detection benchmarks: VisDrone [9], Okutama-Action [10], and ICG [11]. The positive impact of using the pose-diversified synthetic dataset persists regardless of the dataset size, as supported by a series of scaling behavior analyses from various perspectives. Additionally, SynPoseDiv is designed to be general and not limited by the network architecture of the pose generator or image translator, allowing it to benefit from any future advancements in these areas. This chapter is largely based on material published in [27], with additional details presented in Sec. 4.7.

4.2 Related Work

SynPoseDiv offers a novel framework unifying human image synthesis and pose generation to enhance detector performance. We briefly outline related areas:

4.2.1 Synthesizing Human Appearances

Prior work has manipulated clothing [13] and viewing angles [24] to synthesize data using parameterized sampling. However, the vast diversity of human poses makes selecting representative samples challenging. To address this, some methods [130] use MoCap in controlled settings to capture real poses, while others synthesize poses from 2D images using 3D data like depth [131]. Similar to our goal, prior work has modified existing images to depict varied poses. For instance, HUSC [132] used an image translator to create unseen poses. In this chapter, we introduce an image translator that tackles the same challenge, generating novel poses without ground truth images.

4.2.2 Pose Generation

Prior work has generated novel poses by completing missing keypoints using visible joints (e.g., transformers [133]) or sampling from a pose prior (e.g., multi-task representations [134]). Additionally, adversarial learning [135] has been used to guide pose generators to produce plausible poses. In this chapter, building on the second approach, we developed a diffusion model-based pose generator.

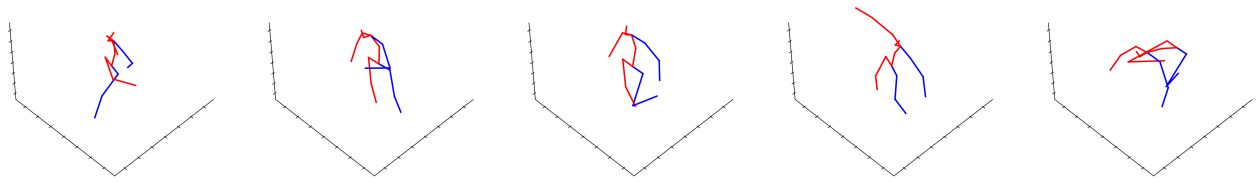


Figure 4.2: Sample poses in $\mathcal{P}_n$. Poses are represented by 3D locations of 17 keypoints.

4.2.3 Generating Images of Novel Poses

Many efforts have focused on transforming human images to match target poses using generators like conditional GANs [136] and diffusion models [137], consistently improving generation quality. To address the challenge of training without paired source and target pose images, DPE [138] used a bidirectional cyclic training strategy that minimized differences between images of the same character after aligning pose and expression. However, this required optimal face reconstruction, which is difficult in our case. To overcome this, in this chapter, we train an image translator with limited transformation capability, applying it only when source and target poses are similar. We construct the target pose sequence with sufficiently similar adjacent poses and generate target images in sequence.

4.3 The SynPoseDiv Framework

4.3.1 Pose Generator

The pose generator is designed to create novel yet realistic poses absent from existing datasets. Each generated pose is represented as 3D coordinates of K keypoints, denoted $\mathbf{p} \in \mathbb{R}^{K\times 3}$ (where $K = 17$ in our case). To achieve this, the pose generator employs a diffusion model, an emerging generative model known for its impressive performance in various domains [128]. Using this generator, we



Figure 4.3: Sample generated images before applying post-processing are shown on the right. Images of the three poses from the original synthetic dataset are shown on the left.

construct a novel pose set $\mathcal{P}_n$. To ensure diversity within the set, we exclude pose pairs that exhibit high similarity. Specifically, a newly generated pose is added to the set only if it is dissimilar to existing poses, defined by a pose distance (Eq. 4.4) less than a specified threshold T_{sim} (in our case, $T_{\text{sim}} = 0.24$). This process continues until N_{pos} poses are included in the set. Fig. 4.2 presents examples of poses in $\mathcal{P}_n$.

4.3.2 Target Pose Selection

Due to the lack of target images corresponding to novel poses for training, the image translator may struggle to generate images with poses significantly different from the input. To address this limitation, we introduce *progressive pose search* that evolves sequential target poses from the input pose. The pseudo-code is provided in Sec. 4.7.1. This method leverages Dijkstra's algorithm [129]

to search for the optimal path from the start node (input pose) to the destination node (target pose) within a graph representation of the novel pose set $\mathcal{P}_n$. The input pose is estimated as 2D human keypoints using ViTPose [2] on the input image, while the target pose is randomly selected from $\mathcal{P}_n$. Intermediate target poses $\mathbf{p}_1,\ \mathbf{p}_2,\ \cdots,\ \mathbf{p}_{n-1}$ between the input pose ($\mathbf{p}_0$) and the final target pose ($\mathbf{p}_n$) are optimally determined from $\mathcal{P}_n$ by minimizing the objective:

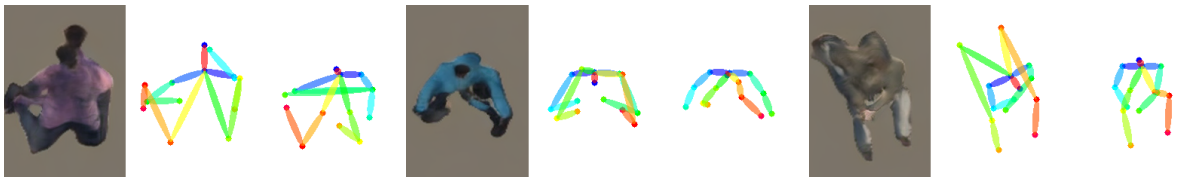
$$d(\mathbf{p}_0, \mathbf{p}_n) = \sum_{i=1}^{n} \left(d_{\text{pose}}\left(\mathbf{p}_{i-1}, \mathbf{p}_i\right)\right)^k \quad \text{s.t.,} |\mathcal{P}_t| \leq N_{\text{max}}, \tag{4.1}$$

where d_{pose} denotes the pose distance (Eq. 4.4) between the 2D keypoint locations of two poses, and $\mathcal{P}_t = \{\mathbf{p}_1,\ \mathbf{p}_2,\ \cdots,\ \mathbf{p}_n\}$ represents the target pose set. To reduce cumulative generation errors, we limit the maximum number of target poses to N_{max}. In our setup, $k = 2$ ensures that shorter pose distances have a greater impact on the objective. Dijkstra's algorithm is solved using *dynamic programming*, with Eq. 4.1 expressed as recurrence relations:

$$d\left(\mathbf{p}_{i-1}, \mathbf{p}_n\right) = \min_{\mathbf{p} \in \mathcal{P}_n} \Big(\left(d_{\text{pose}}\left(\mathbf{p}_{i-1}, \mathbf{p}\right)\right)^k + d\left(\mathbf{p}, \mathbf{p}_n\right) \Big), \tag{4.2}$$

$$\mathbf{p}_i = \arg\min_{\mathbf{p} \in \mathcal{P}_n} \Big(\left(d_{\text{pose}}\left(\mathbf{p}_{i-1}, \mathbf{p}\right)\right)^k + d\left(\mathbf{p}, \mathbf{p}_n\right) \Big). \tag{4.3}$$

The dynamic programming approach decomposes the optimization into sub-problems from $\mathbf{p}_1$ to $\mathbf{p}_{n-1}$, which are solved recursively in reverse order (i.e., optimally finding $\mathbf{p}_{n-1}$ to $\mathbf{p}_1$). This recursive formulation mirrors the shortest-path computation in Dijkstra's algorithm, where each intermediate pose is chosen to minimize the cumulative pose distance to the final target pose. Once a target pose sequence is constructed via progressive pose search, the image translator is applied to each target pose sequentially, using the previously generated image as the source for the next image generation.



$d_{\text{pose}} = 0.32$ $d_{\text{pose}} = 0.16$ $d_{\text{pose}} = 0.22$

Figure 4.4: Filtered noisy images. Each example shows a generated image (left), a target pose (middle), and an estimated pose (right). The pose distance d_{pose} is displayed beneath.



Figure 4.5: Sample generated images after occlusion insertion.

4.3.3 Image Translator

Training the image translator requires source poses, target poses, and their corresponding images. Since target images corresponding to novel poses are unavailable for training, we fine-tune a pre-trained image translator [139] using pose pairs from the original synthetic dataset [24], rather than relying on the novel poses in $\mathcal{P}_n$. After fine-tuning, the image translator generates human images in significantly different poses from the input by iterative application. The translator is applied to human regions segmented from the input synthetic image to prevent non-human regions from adversely affecting generation (Sec. 4.7.3). Segmentation masks, typically available with synthetic data, are used for this purpose. The segmented human region in the input image is then filled using background inpainting [140]. The generated human images are resized at random ratios and positioned at random locations within background images, following the human-size distribution of the original synthetic dataset. Fig. 4.3 illustrates examples of generated images.

4.3.4 Post-Processing: Noisy Image Filtering and Occlusion Insertion

We begin by filtering out noisy images that could negatively impact performance. We conduct a sanity check based on pose estimation to identify and remove low-quality images from all generated images. If the estimated poses differ significantly from the target poses used during generation, these images are considered noisy. Specifically, images are filtered out if the pose distances (Eq. 4.4) between the target and estimated poses exceeds a threshold T_{filt}. In our setup, $T_{\text{filt}} = 0.1$. Examples of noisy generated images are shown in Fig. 4.4.

Next, we address occlusions [141], which significantly contribute to variations in human appearances in real-world aerial-view images. We focus on a common scenario where one human occludes another, often observed in aerial-view images. To simulate this, two generated humans with the same camera position are placed within a single image. Their locations are randomly determined to ensure overlap, with the occluding human positioned lower than the occluded one to mimic natural occlusions in aerial views. Examples of generated images after this step are shown in Fig. 4.5.

4.4 Implementation Details

4.4.1 Measuring Pose Similarity

The pose distance d_{pose}, used in several aspects of our approach (Sec. 4.7.2), is calculated by comparing the keypoint locations (2D or 3D) of two poses $\mathbf{p}_i$ and $\mathbf{p}_j$ as follows:

$$d_{\text{pose}}(\mathbf{p}_i, \mathbf{p}_j) = \sqrt{2\left(1 - \bar{\mathbf{p}}_i^\top \bar{\mathbf{p}}_j^{\text{tr}}\right)}, \tag{4.4}$$

where $\bar{\mathbf{p}}_i$ and $\bar{\mathbf{p}}_j^{\text{tr}}$ are L_2 normalized pose vectors. $\mathbf{p}_j^{\text{tr}}$ represents a transformed $\mathbf{p}_j$ adjusted such that both input poses share the same facing direction and are captured from the same camera position. To determine the facing direction of each pose, we use the direction of the *neck-to-nose* vector. This alignment is generally effective in our case, where head and neck movements are not dominant. The camera position for each pose is either provided by the original synthetic dataset as metadata or is easily obtainable. When comparing a 2D pose and a 3D pose (as used in 'target pose selection'), the pose similarity is calculated after transforming the 3D pose and projecting it into the 2D space.

4.4.2 Pose Generator

Inspired by [128], the pose generator utilizes a diffusion model with two reverse, iterative processes: i) *diffusion* and ii) *denoising*. In the diffusion process at step t, the model diffuses the denoised pose $\mathbf{p}^{(0)}$ into a noisy pose $\mathbf{p}^{(t)}$ by repeatedly adding Gaussian noise $z_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ to the input pose for t times. During the denoising process at step t, a model g_{pos} estimates a clean pose $\hat{\mathbf{p}}^{(0)}$ from the noisy pose $\mathbf{p}^{(t)}$. The initial input $\mathbf{p}^{(t)}$ is drawn from $\mathcal{N}(\mathbf{0}, \mathbf{I})$. The model g_{pos} employs an 8-layer transformer encoder. Each keypoint is treated as an input token and embedded into a 512-dimensional space via a feed-forward network. The noise time step t is similarly projected using a separate network. Each embedding is combined with a positional embedding and fed to the encoder. Excluding the first output token (representing the noise time step), the remaining tokens are processed through another feed-forward network to predict clean keypoints. The training objective is formulated as:

$$\mathcal{L}(\mathbf{p}^{(t)}) = \mathbb{E}_{\mathbf{p}^{(0)} \sim q(\mathbf{p}^{(0)}), t \sim [1,T]} ||\mathbf{p}^{(0)} - g_{\text{pos}}(\mathbf{p}^{(t)}, t)||_2^2. \tag{4.5}$$

4.4.3 Image Translator

We employ the pre-trained NTED [139] as the image translator, but SynPoseDiv is easily adaptable to other image translators. The image translator is fine-tuned on Archangel-Synthetic (Sec. 2.3.2), the original synthetic dataset used in chapter, which contains human image pairs with limited types of similar poses. Image pairs with similar camera positions—information provided as metadata by Archangel-Synthetic—are defined as having similar poses.

4.5 Experiments

4.5.1 Experimental Setup

Tasks and Datasets. Following PTL (Chapter 3), we evaluate aerial-view human detection in two data-scarce scenarios: *low-shot* and *cross-domain* setups, where synthetic data is crucial. Experiments are conducted using the official implementation of PTL, training a RetinaNet [142] variant. We use VisDrone [9], Okutama-Action [10], and ICG [11] as real aerial-view datasets, while Archangel-Synthetic [24] serves as the original synthetic dataset. We adopt the same subset of Archangel-Synthetic (17.3K images) defined in Sec. 3.4. For low-shot setups, we train on 20 or 50 randomly selected VisDrone images (Vis-$\{20, 50\}$) and evaluate on the VisDrone test set. For cross-domain setups, we test on Okutama-Action and ICG. Results are averaged over three runs to reduce randomness. We use COCO-style AP (i.e., $AP_{50:5:95}$) and AP_{50} as the evaluation metrics.

Details of the Pose-Diversified Synthetic Dataset. The pose generator is trained on Human3.6M [130], utilizing the 17 defined in this dataset, of which 14 are used in the image generator. The novel pose set $\mathcal{P}_n$ contains 1000 poses ($N_{\text{pos}} = 1000$). For each synthetic image, 5 final target poses are

Table 4.1: Detection accuracy of the pose-diversified synthetic dataset ('p-div') and the original synthetic dataset ('orig').

method	set	Vis-20 VisDrone	Vis-20 Okutama	Vis-20 ICG	Vis-50 VisDrone	Vis-50 Okutama	Vis-50 ICG
	real	0.58/ 2.27	3.64/ 14.54	0.62/ 1.89	0.76/ 3.30	7.82/ 28.66	1.30/ 5.65
PT-FT	+ orig	0.76/ 2.48	4.24/ 17.17	6.53/ 23.67	1.29/ 3.76	5.32/ 20.96	7.10/ 27.95
	+ **p-div**	**1.21/ 4.02**	**9.14/ 34.70**	**8.20/ 28.80**	**1.84/ 5.37**	**10.39/ 36.83**	**8.63/ 30.09**
		(+0.45/+1.54)	(+4.90/+17.53)	(+1.67/+5.13)	(+0.55/+1.61)	(+5.07/+15.87)	(+1.53/+2.14)
PTL	+ orig	2.07/ 6.72	7.90/ 31.53	**8.81/ 33.71**	2.92/ 9.26	11.49/ 42.51	**8.98/ 33.21**
	+ **p-div**	**2.26/ 7.39**	**8.95/ 36.97**	6.45/ 26.13	**2.99/ 9.42**	**12.89/ 47.24**	6.29/ 25.50
		(+0.19/+0.67)	(+1.05/ +5.44)	(-2.36/-7.58)	(+0.07/+0.16)	(+1.40/ +4.73)	(-2.69/-7.71)

Table 4.2: Effect of post-processing on image quality and detection accuracy (Vis-20).

post-proc. filt.	post-proc. occ.	# img	FID↓	acc.↑ PT-FT	acc.↑ PTL
✗	✗	34,403	124.53	0.89/ 2.83	1.90/ 6.14
✓	✗	17,714	**116.17**	0.94/ 3.03	2.12/ 6.46
✗	✓	34,403	121.71	1.11/ 3.98	1.95/ 6.61
✓	✓	17,714	116.41	**1.21/ 4.02**	**2.26/ 7.39**

selected from $\mathcal{P}_n$, with the maximum number of poses in a target pose sequence (N_{max}) set to 3. Based on these specifications, the total number of generated novel images is 17,714. The entire generation process took approximately 3 hours.

Leveraging Synthetic Data in Model Training. We adopt two approaches: pre-training and fine-tuning ('PT-FT'), and progressive transformation learning (PTL) (Chapter 3). In Sec. 3.4.3 and 3.4.4, these two methods were found to be the most effective among various approaches for leveraging synthetic data in training. Unless specified otherwise, PTL accuracies reported in this chapter are based on results after the 5$^{\text{th}}$ PTL iteration. Additional training details are provided in Sec. 4.7.4.

4.5.2 Main Results

In Table 4.1, we compare detection accuracy using three training setups: the original synthetic dataset ('real + orig'), the pose-diversified synthetic dataset ('real + p-div'), and only real data ('real'). In nearly all cases—except one (Vis-50 tested on the Okutama-Action dataset)—training with synthetic data improves accuracy over using only real images. Even in the exception, the pose-diversified dataset still enhances accuracy, highlighting the benefit of synthetic data in improving detection performance.

The pose-diversified dataset generally outperforms the original synthetic dataset in boosting detection accuracy, regardless of the training approach. This improvement is more noticeable in cross-domain setups (Okutama-Action and ICG), where the VisDrone training set lacks representative features of the test set. However, in rare cases (PTL on the ICG test set), accuracy declines despite using pose-diversified data, which is discussed later.

In addition, PTL typically achieves better accuracy than PT-FT, proving more effective at leveraging synthetic data. However, this advantage disappears on the ICG test set. PTL selects synthetic images that are closer in domain to the training set, but due to the significant perspective differences between the VisDrone (oblique) and ICG (nadir) datasets, these images fail to represent ICG data accurately, reducing performance. Conversely, PT-FT, which uses the entire pose-diversified dataset, provides better representation and performance in both same-domain and cross-domain tasks.

4.5.3 Ablation Study on Post-Processing

Table 4.2 examines the impact of post-processing. Additionally, we report FID (Fréchet Inception Distance) [143], which quantifies the fidelity and diversity of generated images, using the VisDrone training set as the reference real dataset. The results show that filtering noisy images low-

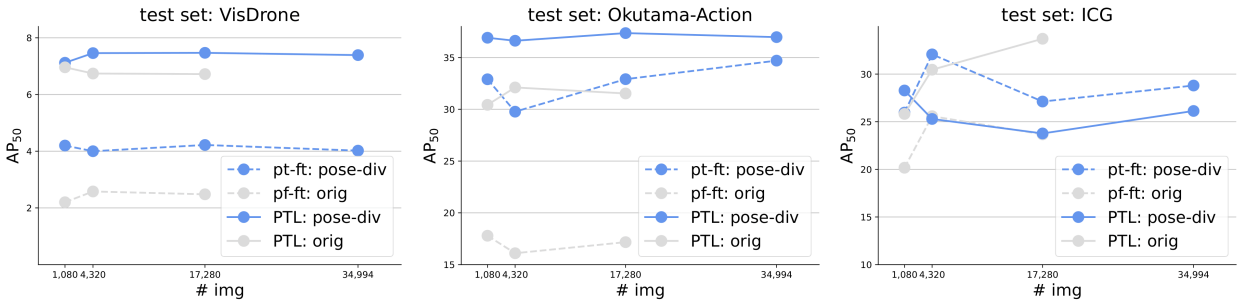


Figure 4.6: Scaling behavior of the synthetic dataset under the Vis-20 setup. The scaling behavior of each dataset is compared by using randomly sampled subsets of 1,080 (1/16th), 4,320 (1/4th), and 17,280 (full) images.

ers FID, improving the quality of pose-diversified synthetic data. This improvement in image quality also boosts detection accuracy. Conversely, accounting for occlusions does not reduce FID but still contributes to higher detection accuracy.

4.5.4 Scaling Behavior of Synthetic Data

Scaling Behavior with Data Size. One might assume that the improved performance of the pose-diversified synthetic dataset is due to its larger size. However, Fig. 4.6 shows it consistently achieves better accuracy on VisDrone and Okutama-Action, even when matched in size with the original synthetic dataset. On ICG, PTL accuracy remains lower with the pose-diversified dataset, consistent with earlier results. This behavior is discussed in later sections.

Scaling Behavior as PTL Evolves. Fig. 4.7 illustrates how PTL gradually increases the synthetic data used during training. Once over 200 synthetic images are selected, a noticeable gap emerges, with the pose-diversified dataset outperforming the original on VisDrone and Okutama-Action. While accuracy for the original dataset plateaus or declines as more images are added, the pose-diversified set continues to improve in cross-domain setups (Okutama-Action and ICG). This highlights *the importance of data diversity in enhancing performance, particularly in cross-domain settings*.

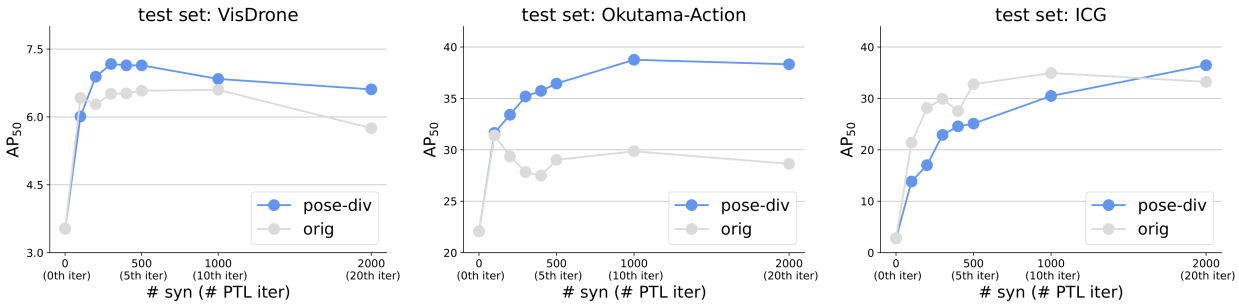


Figure 4.7: Accuracy changes as PTL progresses under the Vis-20 setup. The scaling behavior is evaluated up to the 20th PTL iteration (four times longer than the standard PTL).

Scaling Behavior of PTL on ICG. As shown in Fig. 4.7, when 2000 images are selected, the pose-diversified dataset surpasses the original in accuracy on ICG. Initially, performance is lower due to the limited nadir-view images in the pose-diversified set, while ICG mainly contains nadir-view images. However, *increasing data diversity at scale eventually mitigates this limitation and improves accuracy*.

4.6 Conclusion

In this chapter, we present SynPoseDiv, a framework that modifies synthetic images to adopt novel poses not in the original set. Incorporating these pose-diversified images into training data enables models to better capture varied human appearances, improving aerial-view human detection. We also examined synthetic dataset scaling, showing that representing diverse human poses enhances performance. For example, with PTL, detection accuracy keeps improving using our pose-diversified set, while it plateaus with the original set.

4.7 Additional Details

4.7.1 Progressive Pose Search

The pseudo-code for the 'progressive pose search' algorithm, which optimally searches for target poses through dynamic programming, is shown in Alg. 1. In this algorithm, the function `progsearch` takes the previous step's optimal pose (`p_prev`) and the final pose (`p_final`) as inputs. It then searches for the optimal path from `p_prev` to `p_final`. The function returns the optimal distance (`d_optim`) and a sequence of target poses placed along the optimal path (`seq_tp`), as determined by Eq. 4.2 and 4.3. Designed to operate recursively, the code aims to find a sequence of poses that minimize Eq. 4.1, while ensuring that the number of poses does not exceed N_{max}.

Deriving from Eq. 4.2 and 4.3, we can formulate the following expression defining the optimal pose at the i^{th} step (i.e., $\mathbf{p}_i$):

$$\begin{aligned} d(\mathbf{p}_{i-1}, \mathbf{p}_n) &= d_{\text{pose}}\left(\mathbf{p}_{i-1}, \mathbf{p}_i\right)^k + d\left(\mathbf{p}_i, \mathbf{p}_n\right) \\ &= \min_{\mathbf{p}\in\mathcal{P}_n} \left(\left(d_{\text{pose}}\left(\mathbf{p}_{i-1}, \mathbf{p}\right)\right)^k + d\left(\mathbf{p}, \mathbf{p}_n\right) \right) \\ &\leq d_{\text{pose}}\left(\mathbf{p}_{i-1}, \mathbf{p}\right)^k + d\left(\mathbf{p}, \mathbf{p}_n\right), \end{aligned} \tag{4.6}$$

where $\mathbf{p}$ can be any pose in the novel pose set, i.e., $\mathbf{p} \in \mathcal{P}_n$. Since $d\left(\mathbf{p}_i, \mathbf{p}_n\right) \geq 0$, the above equation can be re-expressed as:

$$d_{\text{pos}}\left(\mathbf{p}_{i-1}, \mathbf{p}_i\right)^k \leq d_{\text{pos}}\left(\mathbf{p}_{i-1}, \mathbf{p}\right)^k + d\left(\mathbf{p}, \mathbf{p}_n\right). \tag{4.7}$$

Since $\mathbf{p}_n$ also belongs to $\mathcal{P}_n$, it can be considered as $\mathbf{p}$ in Eq. 4.7. Therefore, by substituting $\mathbf{p}$ by $\mathbf{p}_n$

and using $d(\mathbf{p}_n, \mathbf{p}_n) = 0$, the above equation can be simplified as follows:

$$d_{\text{pose}}(\mathbf{p}_{i-1}, \mathbf{p}_i)^k \leq d_{\text{pos}}(\mathbf{p}_{i-1}, \mathbf{p}_n)^k . \tag{4.8}$$

In summary, any pose $\mathbf{p} \in \mathcal{P}_n$ *s.t.* $d_{\text{pos}}(\mathbf{p}_{i-1}, \mathbf{p}) \leq d_{\text{pos}}(\mathbf{p}_{i-1}, \mathbf{p}_n)$ can be considered as a candidate for the optimal pose $\mathbf{p}_i$. Note that this condition can be evaluated without the need to compute any sub-optimal objective $d()$, which would entail additional steps in the recursive implementation. In our implementation, we use this condition to reduce the number of candidate poses during the search for the optimal pose, thereby improving efficiency.

The recursion terminates when the pose selected in the previous step is the final target pose (i.e., `p_prev == p_final`) or when the number of target poses reaches N_{max} (i.e., `max_p == 1`).

4.7.2 Pose Transformation

The pose distance measurement, which uses pose transformation to align the facing direction and camera position of two input poses, is employed in several aspects within our method. Here, we provide specific details regarding the pose transformation implemented in each scenario. The pose distance measurement is used in the following scenarios:

- **Building the novel pose set (3D pose vs. 3D pose).** To ensure pose diversity within the novel pose set, any generated poses are excluded if they closely resemble existing poses in the set in terms of pose distance. Here, the poses within the novel pose set are represented by the 3D locations of human keypoints. When comparing a generated pose to each pose in the set, the generated pose undergoes a transformation to align its facing direction with that of the pose being compared, where the facing direction is defined by a *neck-to-nose* vector. Since the pose genera-

Algorithm 1: Python-Style Pseudo-Code for Progressive Pose Search

```
# Input:
#   p_prev:  the pose selected in the previous step
#   p_final:  the final pose
#   max_p:  the maximum number of optimal poses in the current
 sub-optimization
# Output:
#   d_optim:  the optimal distance in the current sub-optimization
#   seq_p:  a sequence of optimal poses in the current sub-optimization

def progsearch(p_prev, p_final, max_p):
   d_optim = 0
   seq_p = []
   # Recursion termination condition 1
   if p_prev == p_final:  return d_optim, seq_p

   d_optim = d_pose(p_prev, p_final)**k # k=2 in our setting
   seq_p = [p_final]
   # Recursion termination condition 2
   if max_p == 1:  return d_optim, seq_p

   for p in set_np:  # set_np:  the novel pose set
      d_p = d_pose(p_prev, p)

      # Reduce the number of candidates for target poses
      if not d_p <= d_pose(p_prev, p_final):
         continue

      # Call the sub-optimization (recursion point)
      d_next, seq_p_next = progsearch(p, p_final, max_p - 1)

      # Check if p is more optimal than the previous optimal pose
      if d_p**k + d_next < d_optim:
         d_optim = d_p**k + d_next
         seq_p = [p] + seq_p_next
    return d_optim, seq_p

# Input:
#   p_0:  the input pose
#   N_max:  the maximum number of target poses
# Ouput:
#   seq_tp:  the optimal target pose sequence

p_n = randselect(set_np) # Randomly select a p_n from set_np
~, seq_tp = progsearch(p_0, p_n, N_max)
```

tor is trained using poses captured from a ground view, two generated poses being compared are assumed to have the same camera position on the ground. Therefore, no further transformation is required.

- **Selecting target poses (2D pose vs. 3D pose).** In the process of target pose selection, we calculate the pose distance between each input pose in the original synthetic data, represented in 2D keypoint positions, and every pose in the novel pose set, represented in 3D keypoint positions. The camera position corresponding to the input pose can be extracted from the metadata of the original synthetic data. Therefore, we first transform poses in the novel pose set to align with the camera positions of the input pose. We then adjust these poses to match the facing direction of the input pose using the *neck-to-nose* vector. After that, the transformed poses are projected onto a 2D plane for the computation of pose distance.

- **Filtering noisy images (2D pose vs. 2D pose).** In the process of noisy image filtering, we compute the pose distance for each generated image, measuring the disparity between the target pose used during generation and the estimated pose. As these two poses correspond to the same image, no transformation is needed.

4.7.3 Image Generation

The overview of our image generation process is shown in Fig. 4.8. To minimize potential noise coming from the background during the generation of novel images, we adopt a strategy of processing the human and background regions independently. Segmentation of the human region can be achieved through the utilization of the segmentation mask provided within the Archangel-synthetic dataset [24], serving as the original synthetic dataset in our scenario.

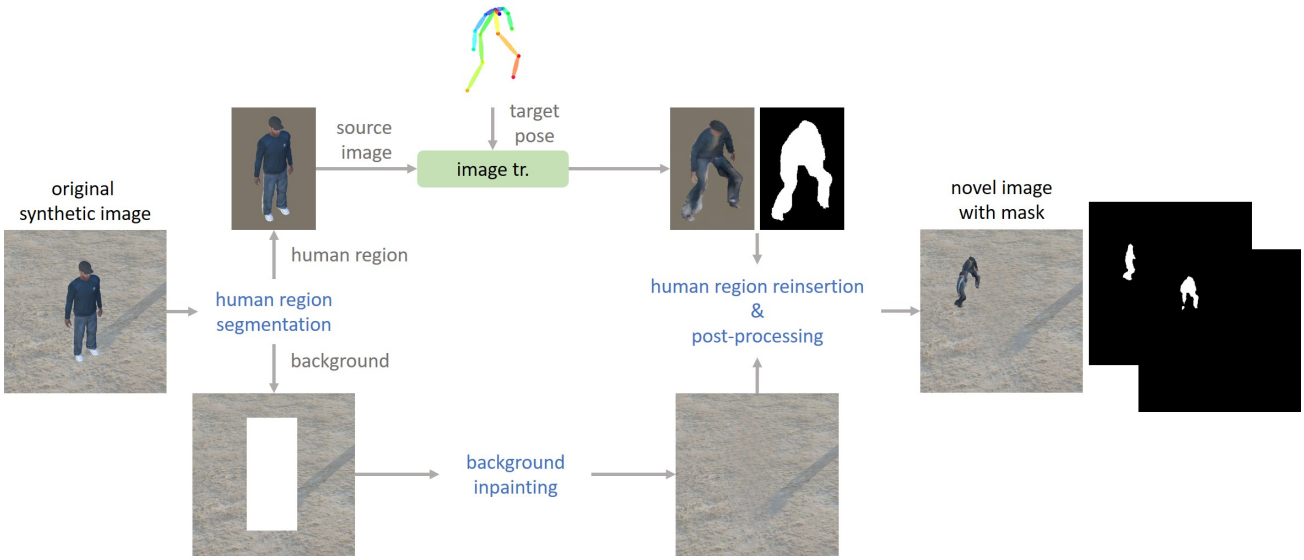


Figure 4.8: Overview of the image generation process.

The human region where the background is excluded is filled with a mono color (in our case, gray) to facilitate the creation of the segmentation mask after image generation. With a given target pose, this human image undergoes alterations to ensure that the human in the image assumes the target pose, using the *source-to-target* image translator ('image tr.'). The segmentation mask for the human within the generated image is then constructed by capturing pixels that deviate from the pre-defined mono color. To mitigate artifacts in the segmentation mask, we apply a series of morphological operations, including four iterations of dilation followed by two iterations of erosion, using a kernel size of 3×3.

The human region, segmented from the generated image using the segmentation mask, is then resized using a randomly selected resizing factor, while maintaining a fixed aspect ratio. To elaborate, we randomly select a human image from the Archangel-synthetic dataset and determine the resizing factor such that the longer side of the human region in the generated image matches that of the human region in the randomly selected image. Subsequently, the generated and resized human region is reinserted into the background at a random location, and the original human region is inpainted. To

introduce occlusion in the generated image, we add two generated human regions onto the image during post-processing.

4.7.4 Training Details

The proposed framework comprises three types of model training: pose generator training, image translator training, and detector training (i.e., PT-FT and PTL). In this section, we delve into additional training details. All the models were trained on a server with 4 NVIDIA GeForce RTX 2080 Ti GPUs.

- **Pose generator training.** We built our pose generator based on the official implementation of MDM [128]. Without further specifications, we maintained all the settings and training strategies as defined in the original implementation. The pose generator was trained on the Human3.6M dataset [130], which was preprocessed according to the procedure outlined in VideoPose3D [144]. We trained the pose generator for 600k iterations with a batch size of 256, a learning rate of 0.0001, and 1k diffusion steps.

- **Image translator training.** We utilized the official NTED model [139] as our image translator, which had been pre-trained on the DeepFashion dataset [145], with an input human image size of 256×176. We fine-tuned the pre-trained image translator on the Archangel-synthetic dataset for 200 epochs with a batch size of 4. During fine-tuning, we adjusted the learning rate to be 10 times smaller (i.e., 0.0002) than the default setting used for pre-training. Additionally, the generative adversarial loss was incorporated after 20k training iterations. Default settings were applied if no further specifications were provided.

- **Detector training.** We utilized two different methods that leverage synthetic data in detector training: PTL and PT-FT. For PTL, we adhered to the official implementation [26], which trained a variant of RetinaNet [142] in a progressive manner, using default hyper-parameters. It is important to note that the original PTL framework only considers the case when there is only a single virtual character present in each synthetic image. However, in our pose-diversified synthetic dataset, there might be two virtual characters in a single image. To address this issue, when calculating the domain gap (i.e., Mahalanobis distance) for each synthetic image containing two characters, we computed the domain gaps separately for each character and represented the synthetic image based on the character with the smaller domain gap. Moreover, if a synthetic image with two characters was selected during a PTL iteration, both characters were involved in the syn2real transformation. For PT-FT, we adopted the same detector architecture and training setups as specified in PTL, with the exception of omitting the progressive aspect. The detector was pre-trained on the synthetic dataset for 6k iterations, followed by fine-tuning on the real dataset for 600 iterations.

4.7.5 Additional Experiments

Quantitative Results. To ensure the general applicability of our primary analysis in Sec. 4.5, we perform additional experiments within the same setup, using different training datasets. Here, we replace the training domain from VisDrone with Okutama-Action or ICG (Table 4.3). Consistent with our primary analysis, using the pose-diversified synthetic dataset generally results in higher accuracy than using the original synthetic dataset, with one exception: when leveraging synthetic data for training using PTL and testing trained detectors on the ICG test set.

Table 4.3: Detection accuracy using Okutama-Action and ICG for training.

method	set	Oku-20 VisDrone	Oku-20 Okutama	Oku-20 ICG	Oku-50 VisDrone	Oku-50 Okutama	Oku-50 ICG
	real	0.42/ 1.52	10.81/ 41.08	1.23/ 5.10	0.59/ 2.27	14.33/ 50.42	1.55/ 7.22
PT-FT	+ orig	0.68/ 2.23	12.21/ 43.35	5.24/ 18.72	0.82/ 2.56	14.46/ 49.07	**7.11/25.67**
	+ **p-div**	**0.85/ 2.79**	**16.13/ 53.54**	**5.95/20.23**	**0.86/ 2.83**	**18.90/ 59.42**	7.04/ 25.39
		(+0.17/+0.56)	(+3.82/+10.19)	(+0.71/+1.51)	(+0.04/+0.27)	(+4.44/+10.35)	(-0.07/ -0.28)
PTL	+ orig	1.09/ 3.20	15.21/ 52.61	**8.00/29.57**	1.29/ 3.56	18.50/ 60.19	**7.47/29.43**
	+ **p-div**	**1.37/ 3.92**	**16.67/ 56.92**	7.61/ 28.16	**1.42/ 3.99**	**19.60/ 62.78**	7.23/ 28.12
		(+0.28/+0.72)	(+1.46/ +4.31)	(-0.39/ -1.41)	(+0.13/+0.43)	(+1.10/ +2.59)	(-0.24/ -1.31)

method	set	ICG-20 VisDrone	ICG-20 Okutama	ICG-20 ICG	ICG-50 VisDrone	ICG-50 Okutama	ICG-50 ICG
	real	0.16/ 0.63	1.13/ 5.01	15.29/ 50.05	0.21/ 0.65	2.26/ 9.67	23.90/ 66.81
PT-FT	+ orig	0.64/ 1.46	3.27/ 14.56	24.61/ 63.67	0.91/ 1.94	5.37/ 21.61	26.80/ 66.20
	+ **p-div**	**0.92/ 1.86**	**6.30/24.32**	**27.49/69.39**	**0.95/ 2.10**	**8.31/ 30.17**	**30.35/73.18**
		(+0.28/+0.40)	(+3.03/+9.76)	(+2.88/+5.72)	(+0.04/+0.16)	(+2.94/+8.56)	(+3.55/+6.98)
PTL	+ orig	0.58/ 1.39	2.42/ 10.03	**25.91**/ 68.19	0.92/ 1.99	4.00/ 16.07	**31.21**/73.74
	+ **p-div**	**0.87/ 2.38**	**4.39/18.95**	24.90/ **71.21**	**1.00/ 2.55**	**6.13/ 24.70**	30.13/**76.82**
		(+0.29/+0.99)	(+1.97/+8.92)	(-1.01/+3.02)	(+0.08/+0.56)	(+2.13/+8.63)	(-1.08/+3.08)

Meanwhile, two cases exhibit different trends from the main results obtained with the VisDrone dataset for training. Both cases arise during testing on the ICG dataset, which primarily consists of nadir-view images. First, when using PT-FT in the Oku-50 setting, utilizing pose-diversified synthetic data presents lower accuracy. In this case, the adverse effects of using pose-diversified synthetic data with relatively fewer nadir-view images become more severe, even though the data is used for pre-training but not for fine-tuning. Second, in the ICG-20 and ICG-50 settings with PTL+ICG, the AP_{50} accuracy improves when using the pose-diversified synthetic dataset, while the AP accuracy remains lower. The lack of nadir-view images during training results in degraded accuracy in precise detection within the ICG test set. However, it appears that this deficiency can be compensated to some extent by ensuring sufficient pose diversity in the data when using a lenient detection criterion (i.e., IoU threshold = 50%).

Table 4.4: Impact of varying the number of final target poses in constructing the pose-diversified synthetic dataset. Detection accuracies ('acc') are reported under the same-domain Vis-20 setup.

			acc.↑	
# tp	# img	FID↓	PT-FT	PTL
3	10,620	**114.90**	**1.25**/ 3.90	**2.38/7.78**
5	17,714	116.41	1.21/ 4.02	2.26/7.39
7	24,912	117.91	1.19/**4.25**	2.22/7.17

Ablation Studies. Table 4.4 investigates the effects of using different number of final target poses for each original synthetic image. In this study, we scale the parameter up to 7 (i.e., 3, 5, and 7), proportionally increasing the number of total generated images. Image quality and detection accuracy do not change significantly even when the data size, controlled by changing the number of final target poses, varies. Therefore, we use 5 final target poses as default, which generates a number of images comparable to the size of the original synthetic dataset.

4.7.6 Pose-Diversified Synthetic Data Samples

In Fig. 4.9, we show more image samples from the pose-diversified synthetic dataset. The proposed method is particularly effective in generating novel poses absent in the original Archange-synthetic dataset, which features virtual characters only in standing, prone, and squatting positions. However, it generally does not provide detailed representations of particular human body parts, such as faces, primarily due to the limited detail originally represented in images captured by UAVs at relatively high altitudes and long ranges. Nonetheless, this reduction in detail about body parts typically does not significantly affect human detection accuracy within the UAV-based perception domain.



Figure 4.9: Sample images from the pose-diversified synthetic dataset. The original synthetic dataset (i.e., Archangel-Synthetic) has only three poses: standing, prone, and squatting.

Chapter 5: AutoComPose — Automatic Generation of Pose Transition Descriptions for Composed Pose Retrieval Using Multimodal LLMs

5.1 Introduction

Understanding and retrieving human body poses remains a fundamental challenge in computer vision, particularly in designing systems that allow users to search for specific poses through intuitive queries. These queries typically fall into two main categories: *visual queries* [146, 147], where users specify their desired pose using an image, and *textual queries* [3, 148, 149], which describe poses in natural language. Visual queries are intuitive but impractical when an exact reference image is unavailable. Textual queries, on the other hand, offer fine-grained control but demand precise and often tedious descriptions. This trade-off highlights a fundamental gap: how can we enable users to retrieve human poses efficiently while balancing ease of use and descriptive flexibility?

A promising solution to this gap is *composed pose retrieval* (CPR) [23], which allows users to refine a reference pose using a textual transition description. As a specialized case of composed image retrieval (CIR) [150, 151, 152, 153, 154], CPR leverages vision-language models (VLMs) [155, 156] trained on triplets of {reference image, target image, transition description}, aligning composed query features with target pose image representations. However, a key bottleneck in this pipeline is the costly and error-prone process of collecting transition descriptions, particularly for highly articulated human poses.

Unlike CIR datasets such as CIRR [151] and FashionIQ [152], which focus on simple transitions like attribute modifications (e.g., color and shape, or background), spatial shifts (e.g., position and viewpoint), CPR requires fine-grained motion descriptions across multiple articulated body parts. This makes human annotation especially challenging: annotators may overlook subtle changes, use

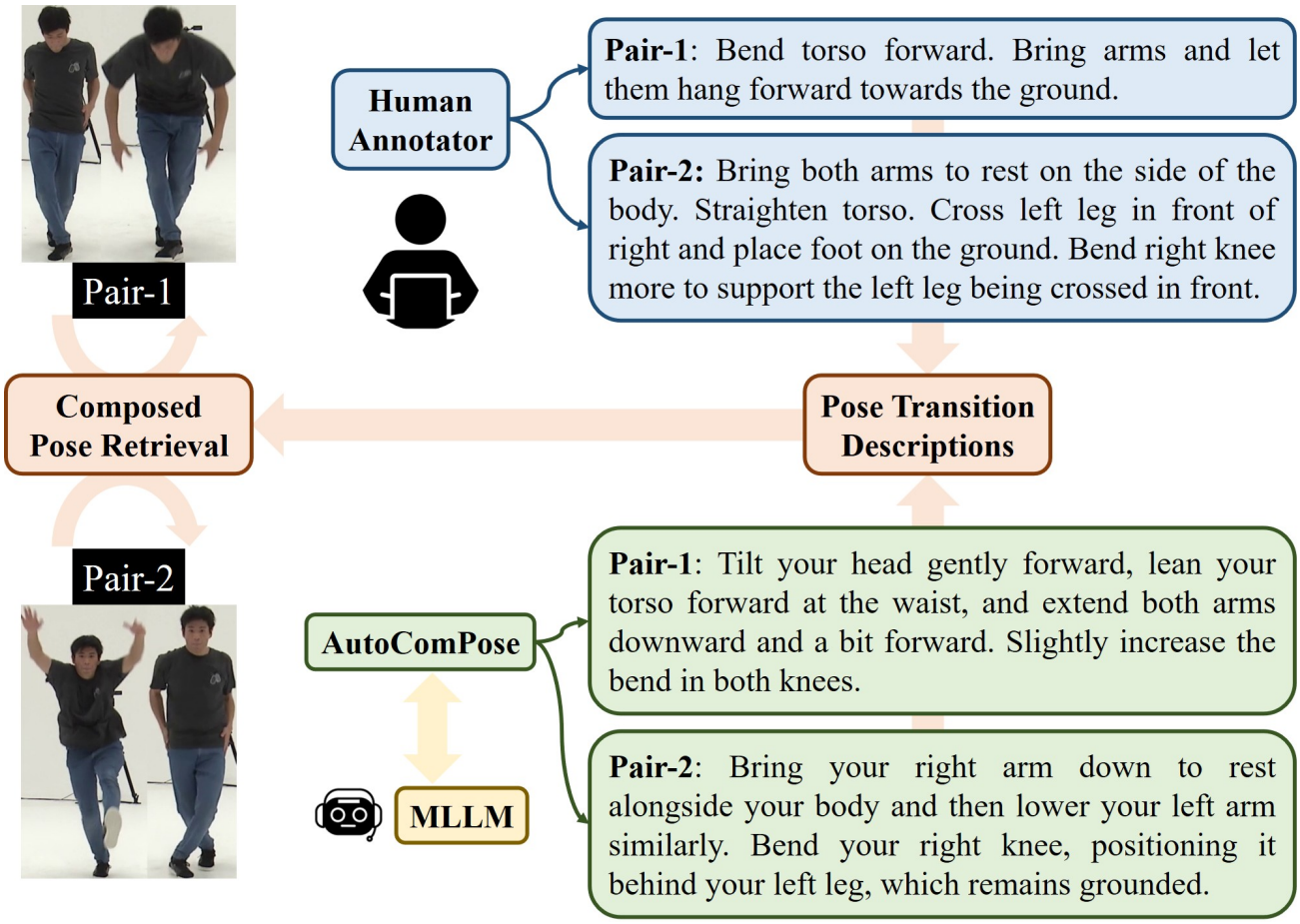


Figure 5.1: Training composed pose retrieval models requires a large and diverse set of human pose transition descriptions, demanding significant human annotation efforts. We propose AutoComPose, a framework that automatically generates pose transition descriptions using multimodal large language models (MLLMs), enhancing pose retrieval performance.

inconsistent phrasing, or introduce subjective language [23, 157]. The result is an expensive, inconsistent, and often insufficiently diverse dataset of pose transitions. Addressing this annotation bottleneck is essential for improving CPR performance. Despite the growing interest in CPR, no prior work has explored *automatically generating pose transition descriptions* tailored for this task. We take the first step in this direction by leveraging multimodal large language models (MLLMs) to construct expressive, structured, and scalable transition annotations.

In this chapter, we present **AutoComPose** (short for **Auto**matic generation of **Com**posed **Pose** descriptions), an MLLM-based framework for automatic human pose transition description generation targeted for CPR, as shown in Fig. 5.1. Prior attempts to pose transition annotation rely on either human labeling [23, 157] or 3D human pose data with hand-crafted generation rules [3, 157], both of which face scalability issues and may require extra information (e.g., 3D human pose) during inference, limiting their applicability. On the other hand, recent advances in multimodal large language models (MLLMs) [158, 159, 160] suggest that these models possess a degree of human pose understanding [148], making them a promising tool for generating rich and diverse pose transition descriptions.

While harnessing MLLMs for generating pose transition descriptions, AutoComPose is designed to ensure expressive, diverse, and robust transition descriptions for CPR. For *expressiveness*, it adopts a two-step approach inspired by human pose descriptions: it first generates detailed body part-based transitions and then integrates and paraphrases them into a complete description. For *diversity*, AutoComPose introduces swapped and mirrored pose variations, enriching the training data.

However, MLLMs are known to generate incorrect or inconsistent descriptions [161], such as misassigning body part movements or fabricating transitions that do not exist in the input images. These errors can propagate into CPR training, potentially degrading retrieval performance. To im-

prove *robustness*, AutoComPose incorporates a *cycle consistency loss* during training, enforcing a constraint that ensures the composed features (i.e., reference image + transition description) can be mapped back to the reference image features via the reverse transition description features. The reverse transition descriptions are obtained by prompting the MLLM with swapped input image pairs. This self-verification mechanism reduces misaligned transitions and improves reliability.

To advance research in CPR, we introduce two new benchmarks constructed from AIST++ [162] and PoseFix [157], namely *AIST-CPR* and *PoseFixCPR*, respectively, to supplement a previous benchmark [23]. These benchmarks establish a foundation for evaluating CPR models under diverse motion scenarios, facilitating future progress in the field. We conduct comprehensive experiments comparing AutoComPose-generated descriptions against human annotations and rule-based methods [3, 157] across all benchmarks. Results show that models trained with AutoComPose consistently yields competitive or superior performance across multiple architectures with respect to the baselines. By significantly reducing annotation costs while maintaining high-quality descriptions, AutoComPose paves the way for more scalable and effective CPR solutions.

Material in this chapter was published in [28]. Our key contributions are as follows:

1. **AutoComPose:** We propose the first MLLM-based, scalable framework for *automatically* generating pose transition descriptions.

2. **New direction and benchmarks:** We initiate the study of auto-annotations for CPR and release two new CPR benchmarks for standardized evaluation.

3. **Expressive and diverse descriptions:** We produced fine-grained body part-based transitions while enhancing diversity through swap/mirror/paraphrase augmentations.

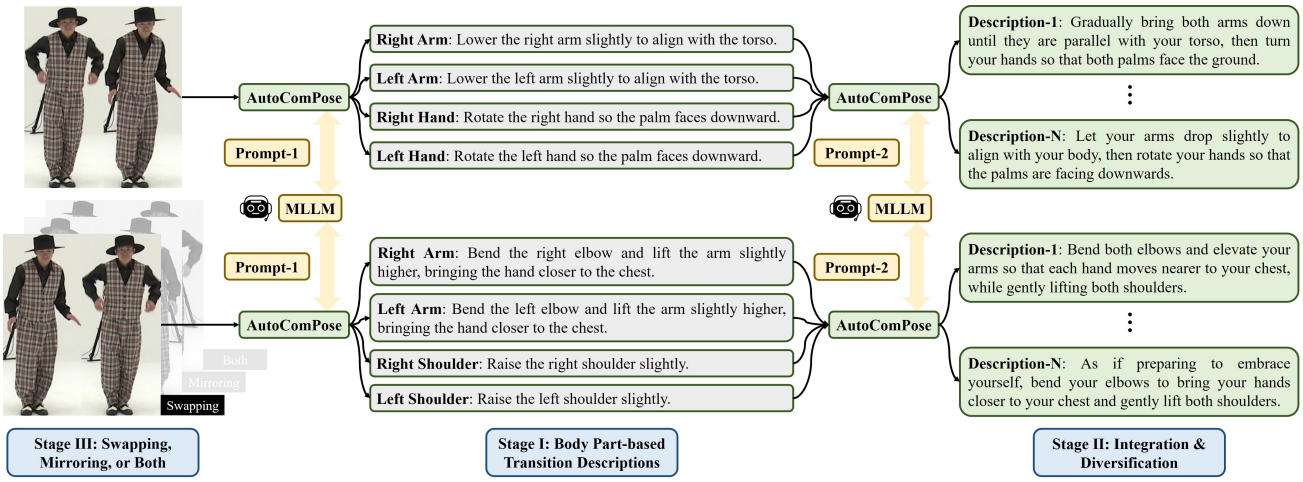


Figure 5.2: AutoComPose generates pose transition descriptions through a three-stage process: In Stage I, it extracts body part-based transition descriptions as an intermediate representation for human body movements to enhance expressiveness. In Stage II, it integrates these body part-based descriptions and diversifies them through paraphrasing. Finally, in Stage III, it generates descriptions for input pose pairs after applying swapping, mirroring, both transformations to further improve diversity.

4. **Cyclic consistency loss:** We introduce a novel cyclic loss to mitigate MLLM errors by enforcing semantic consistency across bi-directional transitions.

5. **State-of-the-art performance:** Evaluations on FixMyPose, AIST-CPR, and PoseFixCPR show that AutoComPose outperforms human-annotations and existing rule-based baselines by a large margin.

5.2 Related Work

5.2.1 Pose Retrieval

Pose retrieval identifies images corresponding to specific human poses. Traditional methods [146, 163] relied on handcrafted features and explicit pose representations. Deep learning has since enabled learning robust feature embeddings for improved retrieval. Wang et al. [164] introduced a

framework for discriminative local descriptors of 3D surface shapes, enhancing pose retrieval accuracy. Pavlakos et al. [165] leveraged ordinal depth supervision to address depth ambiguity in 3D pose estimation. This chapter extends beyond pose retrieval by generating natural language descriptions of pose transitions, enriching retrieval with semantic context.

5.2.2 Composed Image Retrieval

Composed Image Retrieval (CIR) enables retrieving images based on a reference image and a textual modification. Recent advances leverage vision-language models (VLMs) [166, 167, 168, 169] with transformer-based encoders, contrastive learning, and zero-shot methods. Wen et al. [166] proposed a multimodal fusion strategy optimizing linguistic and visual representations. Zhao and Xu introduced NEUCORE [168], a model incorporating multi-modal concept alignment and progressive fusion. Baldrati et al. [167] explored textual inversion for zero-shot CIR, eliminating the need for task-specific training data. A comprehensive survey by Song et al. [169] details CIR models, datasets, and evaluation paradigms.

Despite CIR's advancements, *composed pose retrieval* (CPR) presents unique challenges. Unlike CIR, which modifies object attributes, backgrounds, or styles, our task focuses solely on *human pose*, demanding fine-grained differentiation of body joint movements. Prior work, such as Kim et al. [23], explored pose retrieval but did not isolate pose-specific modifications, as non-pose attributes were concurrently considered. This makes our task significantly more complex, as body movements are continuous and lack strong texture cues commonly used in CIR. While CIR benefits from shape and background context, our model must detect subtle joint displacements under static scene conditions. Additionally, CIR enjoys large-scale labeled datasets such as FashionIQ [152] and CIRR [151],

whereas pose retrieval lacks extensive annotated training data, necessitating automated annotation strategies.

5.2.3 Harnessing MLLMs for Vision Tasks

The integration of multimodal large language models (MLLMs) has revolutionized vision tasks by enabling joint visual-textual processing, enhancing image captioning, visual question answering, and cross-modal retrieval. CLIP's contrastive pretraining facilitates zero-shot classification and retrieval [155], while VisionLLM v2 unifies vision-language tasks within a single framework [170]. Techniques such as Visual Context Compression improve efficiency by reducing visual token redundancy [171], and models like MotionGPT bridge language and motion representation by generating human motion from text [172].

Beyond vision, LLMs are leveraged for automated data annotation, alleviating the cost of manual labeling. Studies highlight their effectiveness in generating high-quality annotations [173], with selective annotation improving few-shot learning efficiency [174]. LLM-driven strategies also enhance event extraction by mitigating data scarcity [175].

Building on these advancements, in this chapter, we propose an MLLM-based approach to automate pose transition annotation, integrating visual-textual processing to improve scalability and consistency. By reducing reliance on manual labeling, our method enhances the efficiency and accuracy of pose retrieval systems while addressing subjectivity and annotation bottlenecks.

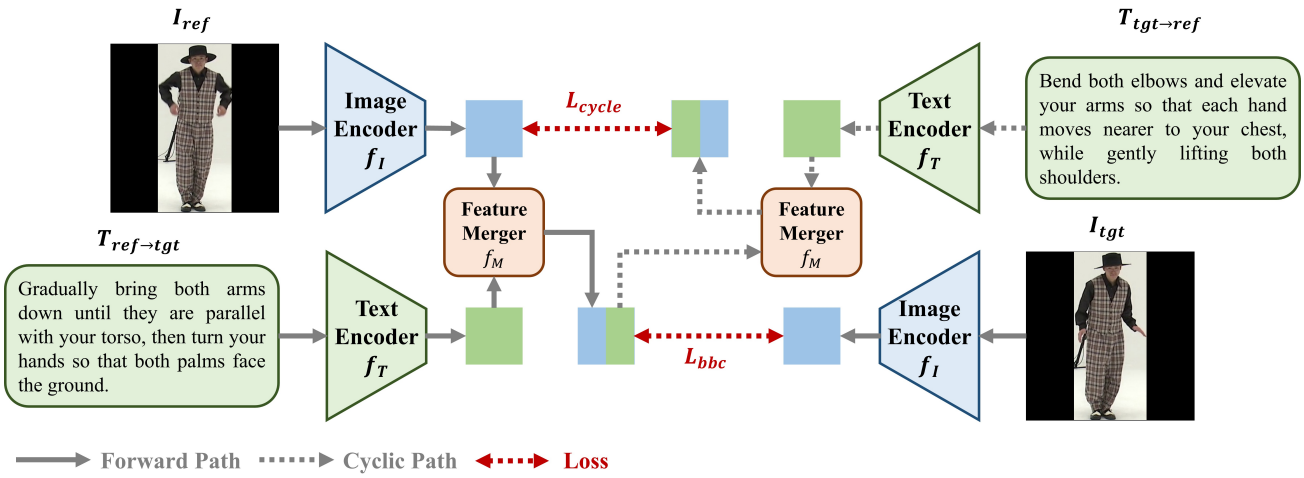


Figure 5.3: During retrieval model training, AutoComPose incorporates a cycle constraint by ensuring that the composed features of input reference images (I_{ref}) and transition descriptions ($T_{ref \to tgt}$) can be transformed back to the reference image features along a cyclic path using the reverse transition descriptions ($T_{tgt \to ref}$), which are obtained during the generation step.

5.3 Method

AutoComPose consists of two primary steps: the *auto*-generation of pose transition descriptions followed by the training of our cyclic model. Each step is elaborated below.

5.3.1 Generating Pose Transition Descriptions

The first step of AutoComPose focuses on generating diverse and expressive pose transition descriptions to enhance CPR effectiveness. An overview of this step is presented in Fig. 5.2. This process consists of three stages:

Stage I: Body Part-based Descriptions. The effectiveness of pose transition descriptions depends on their ability to capture fine-grained body movements with precision. To achieve this, **AutoComPose** leverages an **MLLM** to analyze and compare corresponding body parts between two poses, focusing on key anatomical landmarks: *head, neck, shoulders, arms, elbows, wrists, hands, torso, hips, legs,*

knees, ankles, and feet. For each body part undergoing adjustment, the MLLM generates a concise, targeted description of its motion, ensuring clarity and completeness. The selection of these body parts is grounded in their fundamental role in human pose estimation, as established by state-of-the-art benchmarks in pose tracking [176], dense pose estimation [177], and pose-conditioned motion synthesis [178]. By explicitly structuring pose transitions at the body-part level, AutoComPose not only enhances interpretability but also offers greater adaptability, making it particularly well-suited for complex motion understanding tasks such as CPR.

This structured approach mitigates common limitations in holistic descriptions, where subtle yet critical articulations—such as wrist rotation or knee flexion—are frequently overlooked. By maintaining fine-grained motion analysis, AutoComPose provides a richer, more accurate representation of pose transitions, bridging the gap between human perception and computational models. For a detailed breakdown of the MLLM prompting strategy and an evaluation of Stage I's effectiveness, we refer the reader to Sec. 5.6.3.

Stage II: Integration and Diversification. While body part-based transition descriptions effectively capture fine-grained movement details, they do not naturally align with how users articulate queries. Rather than providing structured, bullet-point descriptions, users typically favor a single, coherent narrative that seamlessly conveys all relevant information. To bridge this gap, AutoComPose prompts the MLLM to synthesize a fluid and concise description that preserves the key motion semantics. Moreover, to account for the inherent variability in human annotations, AutoComPose generates multiple alternative descriptions from the same body part-based inputs, capturing diverse linguistic expressions of the same transition. Additionally, by encouraging the use of analogies when appropriate, AutoComPose enhances expressiveness, making pose transitions more intuitive and relatable. Further details on

the prompting strategy can be found in Sec. 5.6.3.

Stage III: Swapping and Mirroring. Data augmentation is a standard practice in training image-based machine learning models, with techniques like random flipping [179] commonly employed to improve robustness. However, in the case of CPR, applying such augmentations is non-trivial, as the corresponding transition descriptions must precisely align with the transformations applied to the images—an effort that traditionally demands additional manual annotations. AutoComPose addresses this challenge by automatically generating consistent transition descriptions through direct manipulation of input image pairs. By default, it applies swapping (time reversal), mirroring (left-right flipping), or both, effectively increasing dataset diversity without requiring extra annotation efforts. By automating these augmentations, AutoComPose not only enhances scalability but also ensures diverse and balanced datasets, making it a practical and efficient solution for CPR.

5.3.2 Cyclic CPR Model Training

MLLMs occasionally generate unrelated or incorrect responses, which could degrade retrieval performance when these descriptions are used for training. To mitigate this issue and enhance the robustness of pose retrieval, AutoComPose incorporates a cyclic model training approach, as shown in Fig. 5.3.

Generic VLM Training. Assume that we have a reference image I_{ref}, a target image I_{tgt}, and the corresponding pose transition description $T_{ref \to tgt}$ generated by AutoComPose. We can fine-tune any off-the-shelf VLMs by projecting I_{ref} and $T_{ref \to tgt}$ onto a joint manifold using the image encoder f_I and text encoder f_T of the VLMs. The extracted features are then merged using a feature merger f_M, which could be as simple as an element-wise sum. The similarity between the composed features and

the target image features is then measured using specific metric functions, and this similarity score is used as the training loss.

In this chapter, we follow [150, 180, 181] and adopt a batch-based classification loss, which is formulated as:

$$L_{bbc} = \frac{1}{B}\sum_{i=1}^{B} -\log \frac{\exp\{\lambda * \kappa(\psi^i, \phi^i)\}}{\sum_{j=1}^{B}\exp\{\lambda * \kappa(\psi^i, \phi^j)\}}, \tag{5.1}$$

where $\psi^i = f_M(f_I(I^i_{ref}), f_T(T^i_{ref\to tgt}))$ represents the composed features, and $\phi^i = f_I(I^i_{tgt})$ represents the target image features. Following [150], κ is defined as the cosine similarity measure, and the parameter λ is set to 100.

Cycle Constraint. MLLM-generated pose transition descriptions can occasionally introduce errors, such as misidentifying body parts, hallucinating non-existent movements, or omitting crucial motions. As a result, the text features $f_T(T_{ref\to tgt})$ and $f_T(T_{tgt\to ref})$ may fail to capture the true correlations between the reference and target image features. This can lead to unreliable composed features and suboptimal model training. One possible solution is to use a more powerful MLLM or fine-tune it on domain-specific data [182], but this is often resource-intensive or impractical. Another alternative is detecting and correcting incorrect transition descriptions, but this is non-trivial.

AutoComPose takes a different approach based on a simple hypothesis: if the composed features are correct, they should be reversible using the reverse transition description features. Cycle constraints have been widely used in various vision tasks [19, 120], but to our knowledge, they have not been explored in CPR due to the lack of conjugated transition description data (i.e., both forward and backward pose transitions). However, as described in the previous section, AutoComPose can automatically generate the reverse pose transition description $T_{tgt\to ref}$ by swapping the reference and target images.

Similar to L_{bbc}, AutoComPose defines the cycle constraint loss L_{cycle} as:

$$L_{cycle} = \frac{1}{B}\sum_{i=1}^{B} -\log \frac{\exp\{\lambda * \kappa(\hat{\psi^i}, \hat{\phi^i})\}}{\sum_{j=1}^{B}\exp\{\lambda * \kappa(\hat{\psi^i}, \hat{\phi^j})\}}, \tag{5.2}$$

where $\hat{\psi^i} = f_M(\psi^i, f_T(T^i_{tgt\to ref}))$ represents the conjugated composed features, and $\hat{\phi^i} = f_I(I^i_{ref})$ represents the reference image features. The total training loss is then defined as: $L_{total} = \omega * L_{bbc} + (1-\omega) * L_{cycle}$, where ω is set to 0.5 in this chapter.

5.4 Experiments

5.4.1 Datasets and Experimental Setup

We evaluate AutoComPose using three datasets that encompass a diverse range of body movements:

FIXMYPOSE. FIXMYPOSE [23] provides synthetic pose image pairs along with corresponding human-annotated pose transition descriptions, specifically designed for pose correctional captioning and retrieval tasks. The original dataset includes environment-related descriptions (e.g., facing the *stereo system*), which fall outside the scope of this work. To focus specifically on human body poses, we filtered out environment-related descriptions using a large language model, resulting in 1997 training pairs and 510 validation pairs. Since FIXMYPOSE does not publicly release its test set, we used the validation set for our experiments. Please refer to Sec. 5.6.2 for more details on the filtering process.

PoseFixCPR. PoseFix [157] provides human-annotated pose transition descriptions for 3D human meshes sampled from AMASS [183], a large-scale dataset of 3D human motions in SMPL for-

mat [184]. Additionally, the dataset includes automatically generated annotations by a rule-based pipeline [3] that relies low-level properties of 3D body keypoints. Since the original dataset lacks corresponding 2D images, we followed the BEDLAM [13] rendering pipeline to synthesize 2D images from SMPL-X [185] bodies using Unreal Engine [17]. To ensure diversity, we rendered images by randomly sampling the following aspects: 111 body shapes [186], 50 skin tones per gender [187], 1738 clothing textures [13], 115 HDRI background images [188], and four different viewing angles (east, north, south, west). This process resulted in 4596 training pairs and 1188 testing pairs, forming a newly generated dataset tailored for the CPR task, which we refer to as *PoseFixCPR*.

AIST-CPR. AIST++ [162] is a large-scale 3D human dance motion dataset containing a wide range of 3D motion sequences along with music. The dataset comprises 1408 video sequences, capturing 30 different dancers performing dances across different genres, music styles, and choreographies. To ensure reasonable and sufficient gaps between paired poses, we constructed pose pairs by first selecting every 15th frames (0.25 seconds apart). Next, we applied a pose estimator [2] to filter out pose pairs whose pose distance [27] falls outside the predefined range ($[0.1, 2.0]$ in this chapter). This process resulted in 12151 training pairs and 3788 testing pairs, resulting in a new CPR benchmark, namely *AIST-CPR*. Since the dataset does not include transition descriptions for evaluation, we randomly sampled 400 pose pairs from the test set for manual annotation. Each sampled pose pair was annotated by two different annotators (from a pool of four) to ensure sufficient diversity.

Generating Descriptions of Pose Transitions. We leveraged AutoComPose to generate pose transition descriptions for all training pairs across the three datasets mentioned above. For all experiments, we used GPT-4o [189] (*gpt-4o-2024-08-06*) as the MLLM, as it was one of the top-performing MLLMs at the time of this study. We also conducted a study using GPT-4o mini to demonstrate

AutoComPose's applicability to more accessible MLLMs; results from this study are presented in Sec. 5.6.1. By default, AutoCompose generates three descriptions for each pose pair. When swapping and mirroring transformations are applied, this results in a total of 12 descriptions associated with one single pose pair. In rare cases, the MLLM may generate responses that deviate from our guidelines or may even refuse to generate a response (fewer than 2.5% of cases). We ignored these pairs for simplicity, ensuring that all training pairs used in the model training contain three transition descriptions for each transformation (original, swapping, mirroring, and both).

Model Architecture and Training Details. All experiments were conducted based on the implementation from [150], using a single NVIDIA GeForce RTX 2080 Ti GPU. We evaluated AutoComPose on CLIP [155] with four different backbones (RN50, RN101, ViT-B/32, and ViT-B/16) to assess its applicability across different computational budgets. During training, we initialized from the official pre-trained models and only fine-tuned the text encoder to reduce the risk of overfitting the image encoder. When multiple descriptions were associated with a pose pair (i.e., due to paraphrasing), we randomly sampled one for each training iteration. As a baseline, we used element-wise sum as the feature merger. Additionally, we tested AutoComPose with Combiner [150], a dedicated feature merger that has demonstrated superior performance compared to naïve element-wise sum. We followed a the training procedure similar to [150]. First, we fine-tuned the text encoder for 50 epochs with batch size of 128 and a learning rate of $2\mathrm{e}{-6}$. Then, we trained the Combiner while keeping both the image and text encoders fixed for 100 epochs with batch size of 512 and a learning rate of $2\mathrm{e}{-5}$.

Evaluation Metrics. For evaluation, we perform retrieval on the entire test set for each dataset. We evaluate model performance using Recall@k ($k \in 1, 5, 10, 50$), or R@k, a standard metric in image retrieval. Recall@k measures the proportion of queries for which the correct target image appears in

Table 5.1: Comparison of pose transition descriptions produced by human annotators and AutoComPose on FIXMYPOSE and PoseFixCPR datasets. The best and second-best results in each column are highlighted.

	Backbone	Pose Transition Descriptions	FIXMYPOSE (*Size* = 7106)				PoseFixCPR (*Size* = 2376)			
			R@1	R@5	R@10	R@50	R@1	R@5	R@10	R@50
CLIP [155]	RN50	Human Annotators	0.20	1.57	2.94	13.53	48.73	65.15	72.14	86.11
		AutoComPose	**8.24**	27.45	38.63	63.53	61.36	75.84	80.22	90.32
	RN101	Human Annotators	3.92	12.94	19.61	45.29	59.09	72.73	79.21	88.72
		AutoComPose	6.47	**32.35**	43.33	71.18	63.22	76.77	82.91	91.41
	ViT-B/32	Human Annotators	5.10	20.39	29.41	54.31	54.29	66.25	72.14	85.02
		AutoComPose	7.06	24.90	36.08	62.16	58.08	72.31	76.94	86.87
	ViT-B/16	Human Annotators	7.45	30.20	40.98	70.59	59.26	73.74	78.62	88.30
		AutoComPose	**9.41**	**32.75**	**47.45**	75.49	64.14	76.26	80.13	89.56
CLIP w/ Combiner [150]	RN50	Human Annotators	3.14	13.14	20.98	44.12	67.93	82.32	86.36	94.11
		AutoComPose	**9.41**	31.76	43.92	75.49	81.40	92.68	94.95	98.15
	RN101	Human Annotators	5.49	19.22	27.84	56.47	72.14	84.85	89.81	96.04
		AutoComPose	7.64	31.37	44.90	**75.88**	80.72	93.18	95.88	98.73
	ViT-B/32	Human Annotators	4.51	16.67	24.90	52.94	63.13	77.10	82.41	92.93
		AutoComPose	7.45	28.63	44.51	74.12	**82.58**	**94.11**	**95.88**	**98.74**
	ViT-B/16	Human Annotators	7.06	25.10	38.24	68.82	67.85	82.66	86.87	94.28
		AutoComPose	**8.24**	31.57	**47.65**	**81.37**	**84.09**	**93.43**	**96.30**	**99.07**

the top-k retrieved results. Formally, given a query q with its ground-truth target t, R@k is defined as:

$$\text{Recall@}k = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}\{t_i \in \mathcal{R}_i^k\}, \tag{5.3}$$

where N is the total number of queries, $\mathcal{R}_i^k$ is the set of top-k retrieved images for query q_i, and $\mathbb{1}\{\cdot\}$ is an indicator function. Higher R@k signifies better retrieval performance, reflecting the model's ability to rank relevant images near the top.

5.4.2 Main Results

Comparison with Human Annotators. Table 5.1 presents a detailed evaluation comparing human annotations to AutoComPose-generated annotations on the FIXMYPOSE and PoseFixCPR datasets.

Table 5.2: Comparison between the rule-based method and AutoComPose on PoseFix-CPR, highlighting the advantages of AutoComPose in providing less constrained and more generalizable pose transition descriptions.

	Backbone	Pose Transition Descriptions	PoseFixCPR (Size = 2376)			
			R@1	R@5	R@10	R@50
CLIP [155]	RN50	Rule-based	42.00	59.51	67.09	84.34
		AutoComPose	61.36	75.84	80.22	90.32
	RN101	Rule-based	48.32	64.39	69.95	82.92
		AutoComPose	63.22	76.77	82.91	91.41
	ViT-B/32	Rule-based	53.45	67.34	73.48	85.44
		AutoComPose	58.08	72.31	76.94	86.87
	ViT-B/16	Rule-based	58.00	72.98	78.11	88.30
		AutoComPose	64.14	76.26	80.13	89.56
CLIP w/ Combiner [150]	RN50	Rule-based	73.15	86.62	90.07	96.46
		AutoComPose	81.40	92.68	94.95	98.15
	RN101	Rule-based	71.55	86.03	89.90	95.96
		AutoComPose	80.72	93.18	**95.88**	98.73
	ViT-B/32	Rule-based	71.04	86.70	90.40	96.63
		AutoComPose	**82.58**	**94.11**	**95.88**	**98.74**
	ViT-B/16	Rule-based	73.15	87.71	91.41	96.80
		AutoComPose	**84.09**	**93.43**	**96.30**	**99.07**

These results span multiple backbone architectures and different strategies for integrating visual and textual features (i.e., CLIP vs. Combiner). *Notably, across all configurations and datasets, AutoComPose consistently surpasses human annotations, never falling short in any scenario.* This underscores AutoComPose's ability to generalize effectively, eliminating the need for labor-intensive human annotations. We attribute the higher overall performance on PoseFixCPR to fundamental differences in dataset construction. PoseFixCPR features a substantially smaller gallery (2376 vs. 7106) and more distinct one-to-one pose matches, which simplify retrieval. In contrast, FIXMYPOSE's larger gallery, combined with lower pose and character diversity, results in many visually similar candidates, increasing retrieval ambiguity.

Table 5.3: Evaluation on AIST-CPR using descriptions generated via AutoComPose.

	Backbone	Pose Transition Descriptions	AIST-CPR (Size = 8409)			
			R@1	R@5	R@10	R@50
CLIP [155]	RN50	AutoComPose	3.25	13.38	22.62	50.75
	RN101		5.12	16.25	23.00	52.88
	ViT-B/32		3.13	9.62	16.00	42.88
	ViT-B/16		3.38	12.50	22.00	49.13
Combiner [150]	RN50	AutoComPose	5.75	20.75	31.25	65.87
	RN101		**7.00**	**22.37**	**32.62**	**68.12**
	ViT-B/32		5.75	17.87	28.50	63.63
	ViT-B/16		**6.75**	**21.13**	**32.75**	**69.63**

Comparison with Rule-based Generation. Table 5.2 clearly demonstrates the consistent superiority of our AutoComPose over the rule-based pose transition descriptions of PoseFix [157], across various backbone architectures and combiners. While PoseFix aimed to automate pose description generation, its approach ultimately relied on predefined aggregation rules and template-based sentences, making it inherently rule-based. Furthermore, PoseFix depended on a constrained set of 'pair-codes'—descriptors anchored to absolute 3D keypoint positions in image pairs—which fundamentally restricted the expressiveness and generalizability of the generated descriptions. This rigid formulation limited its adaptability to diverse poses and transitions. In contrast, AutoComPose dynamically constructs descriptions without such constraints, enabling more flexible and contextually rich representations. The substantial performance gap observed in Table 5.2 further underscores the importance of allowing greater descriptive freedom, reinforcing the effectiveness of our approach.

Results on AIST-CPR. Table 5.3 presents retrieval performance results across a comprehensive set of feature merger (CLIP vs. Combiner) combined with various backbone architectures on the AIST-

Table 5.4: Ablation study demonstrating the impact of data augmentation and cyclic loss guidance on retrieval accuracy. The (-) symbol denotes the removal of the corresponding component. SW and MI denote swapping and mirroring, respectively.

	Dataset	AutoComPose	R@1	R@5	R@10	R@50
CLIP (RN50)	FIXMYPOSE	Our (full)	**8.24**	**27.45**	**38.63**	**63.53**
		(-) Cyclic	5.88	20.00	32.35	55.10
		(-) SW & MI	2.35	9.02	14.31	34.51
	PoseFixCPR	Our (full)	**61.36**	**75.84**	**80.22**	**90.32**
		(-) Cyclic	56.48	74.07	78.20	89.56
		(-) SW & MI	48.15	64.65	73.06	86.45
	AIST-CPR	Our (full)	**3.25**	**13.38**	**22.62**	**50.75**
		(-) Cyclic	3.13	12.00	19.75	44.37
		(-) SW & MI	3.00	7.75	14.25	36.25
CLIP (RN101)	FIXMYPOSE	Our (full)	**6.47**	**32.35**	**43.33**	**71.18**
		(-) Cyclic	**6.47**	24.51	39.41	64.31
		(-) SW & MI	5.88	18.82	30.39	56.08
	PoseFixCPR	Our (full)	**63.22**	**76.77**	**82.91**	**91.41**
		(-) Cyclic	59.43	75.76	81.14	90.99
		(-) SW & MI	52.86	69.02	76.09	86.45
	AIST-CPR	Our (full)	5.12	**16.25**	23.00	**52.88**
		(-) Cyclic	**5.25**	15.75	**23.75**	49.75
		(-) SW & MI	3.13	11.00	16.00	39.63

CPR. Notably, the AIST-CPR dataset lacks human-annotated transition description for the train set, highlighting the importance of our methodological approach in generating and utilizing such annotations.

5.4.3 Additional Analyses

Ablation Study of Each Component in AutoComPose. In Table 5.4, we demonstrate the critical role of dataset diversification and the effectiveness of cyclic loss guidance in optimizing model training. Notably, our results show consistent and substantial improvements in retrieval accuracy across all

Table 5.5: Impact of paraphrasing. The numbers 1, 3, and 5 indicate the number of paraphrased descriptions per pose transition.

Model	Paraphrasing	FIXMYPOSE (Size = 7106)			
		R@1	R@5	R@10	R@50
CLIP [155]	1	5.88	20.39	30.20	54.31
	3 (default)	8.24	27.45	38.63	63.53
	5	**9.02**	28.63	39.02	64.71
Combiner [150]	1	7.65	27.65	40.00	67.06
	3 (default)	**9.41**	**31.76**	**43.92**	**75.49**
	5	8.82	**31.37**	**43.53**	**77.45**

benchmark datasets and backbone architectures—at times achieving up to a threefold increase (e.g., 9.02 to 27.45). Importantly, *these gains come without any additional computational cost* during inference, making our approach highly efficient and scalable.

Impact of Paraphrasing. We conducted an empirical study to rigorously assess the impact of description diversification via paraphrasing (see Stage II in Section 5.3), where AutoComPose generates diverse linguistic expressions for the same pose transition. As shown in Table 5.5, we observe a consistent performance improvement, confirming that diversification enhances retrieval accuracy. This result supports the hypothesis that accounting for natural language variability improves model robustness, better aligning with the variations in user queries in real-world scenarios.

Paraphrasing and Cyclic Training for Other Methods. We evaluated the impact of paraphrasing and cyclic training on human-annotated and rule-based transition descriptions. Official paraphrases and reverse descriptions (via the pipeline in [157]) were used. As shown in Table 5.6, paraphrasing improved performance in both cases, while AutoComPose remained the most effective due to its superior design. Moreover, cyclic training enhanced the performance of the rule-based method, confirming its broader applicability.

Table 5.6: Effects of paraphrasing and cyclic training on human-annotated and rule-based descriptions (CLIP-RN50 setting).

Pose Transition Descriptions	Paraphrasing	PoseFixCPR (Size = 2376)			
		R@1	R@5	R@10	R@50
Human	1	48.73	65.15	72.14	86.11
	3	52.02	**71.55**	**77.44**	**88.55**
Rule-based	1	31.73	50.42	58.84	77.10
	3	42.00	59.51	67.09	84.34
	3 (w/ Cyclic)	**55.05**	70.20	75.67	88.47
AutoComPose	3	**61.36**	**75.84**	**80.22**	**90.32**

5.4.4 Qualitative Results

Fig. 5.4 presents example retrieval results obtained using our model trained on transition descriptions from AutoComPose. While the model is not flawless, we observe that images aligning with the given [reference image + transition description] are ranked among the top results, whereas less relevant images are positioned further down the ranking. Additional qualitative results are provided in Sec. 5.6.1.

5.5 Conclusion and Broader Impact

In this chapter, we introduced *AutoComPose*, the first MLLM-based framework for automatically generating pose transition descriptions, eliminating the need for costly human annotations and enhancing CPR accessibility. AutoComPose is dataset-agnostic, integrating seamlessly with any human image dataset. To foster future research, we release AIST-CPR and PoseFixCPR, as new testbeds for CPR evaluation. Beyond CPR, our work demonstrates MLLMs' potential for structured motion understanding, reducing annotation costs while improving scalability. By introducing benchmarks and

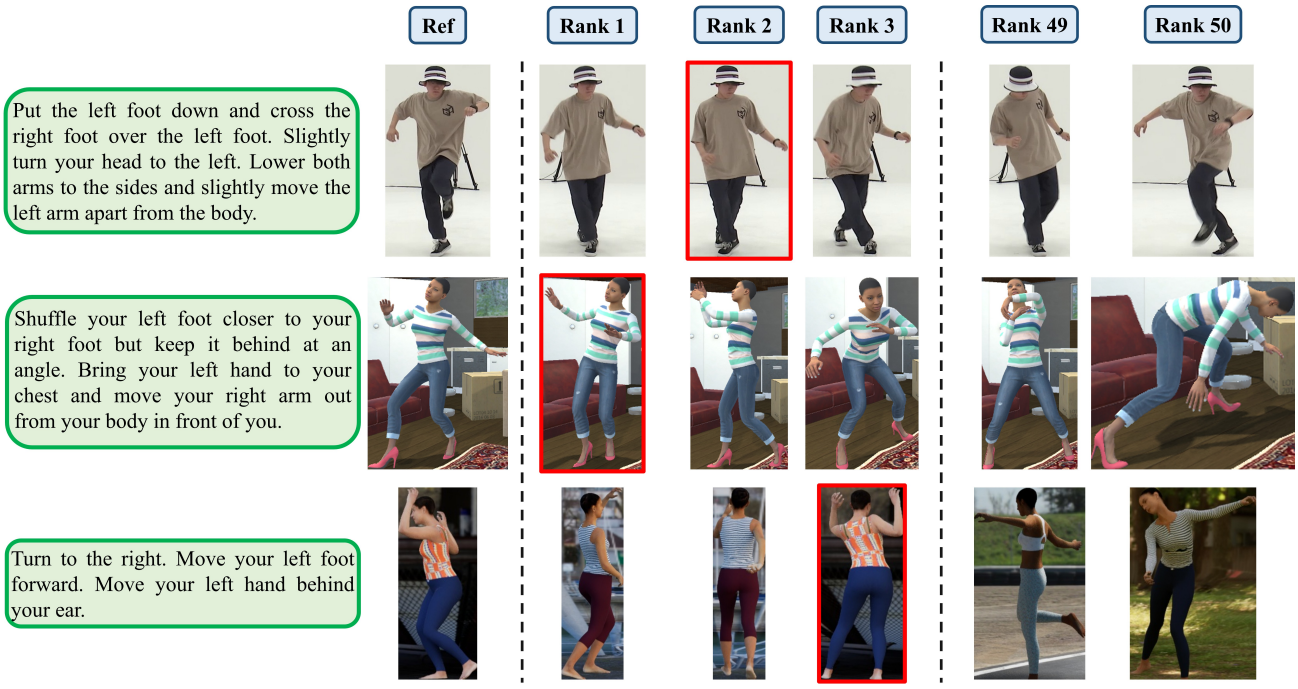


Figure 5.4: Example CPR results. Ranked retrieval results are shown for a given reference image (Ref) and a transitional pose description (green box) as the query. Red boxes indicate the correct target images. The top, middle, and bottom rows correspond to results from AIST-CPR, FIXMYPOSE, and PoseFixCPR, respectively.

a scalable annotation framework, we lay the foundation for advancing retrieval tasks and encourage broader adoption of auto-generated pose descriptions in vision-language research.

5.6 Additional Details

5.6.1 Additional Experiments

Qualitative Results. Similar to Fig. 5.4, we present additional qualitative results for FIXMYPOSE (Fig. 5.5), PoseFixCPR (Fig. 5.6), and AIST-CPR (Fig. 5.7), respectively. The observed trend remains consistent: images that closely align with the provided reference image and transition description are ranked higher, while less relevant images are ranked lower.

Effectiveness of Stage I. To validate the effectiveness of Stage I in AutoComPose (i.e., body part-

Table 5.7: Additional ablation study on AutoComPose. Results are reported under the CLIP-RN50 setting.

Pose Transition Descriptions	FIXMYPOSE (Size = 7106) R@1	R@5	R@10	R@50
AutoComPose (full)	**8.24**	**27.45**	**38.63**	**63.53**
(-) Stage I	**8.63**	**26.67**	**37.84**	**62.75**
w/ GPT-4o mini	6.67	19.80	30.20	56.86
Human	0.20	1.57	2.94	13.53

based descriptions), we modified the original prompts used by AutoComPose (Figures 5.13 and 5.14) to exclude body part-related queries (Figures 5.15 and 5.16). As shown in Table 5.7, removing Stage I results in a noticeable drop in performance, underscoring its critical role.

Smaller, More Accessible MLLM. We conducted an experiment using GPT-4o mini (*gpt-4o-mini-2024-07-18*), a smaller model that is over 16 times more cost-efficient than GPT-4o. As shown in Table 5.7, the impact of AutoComPose remains substantial, demonstrating its effectiveness and applicability even with lightweight models.

5.6.2 Dataset Details

Triplet Examples and Word Clouds. We showcase example pairs from the three datasets in Fig. 5.8. We also compare word clouds generated from texts produced by AutoComPose with those from human annotators across the three datasets we used, as shown in Fig 5.9. Our findings indicate that texts generated by AutoComPose exhibit greater diversity and less bias.

FIXMYPOSE Filtering. A portion of the pose transition descriptions provided by FIXMYPOSE [23] contains instructions related to environment, such as guiding a pose transition by referring certain objects in a scene. Since our focus is on generating pose transition descriptions solely based on

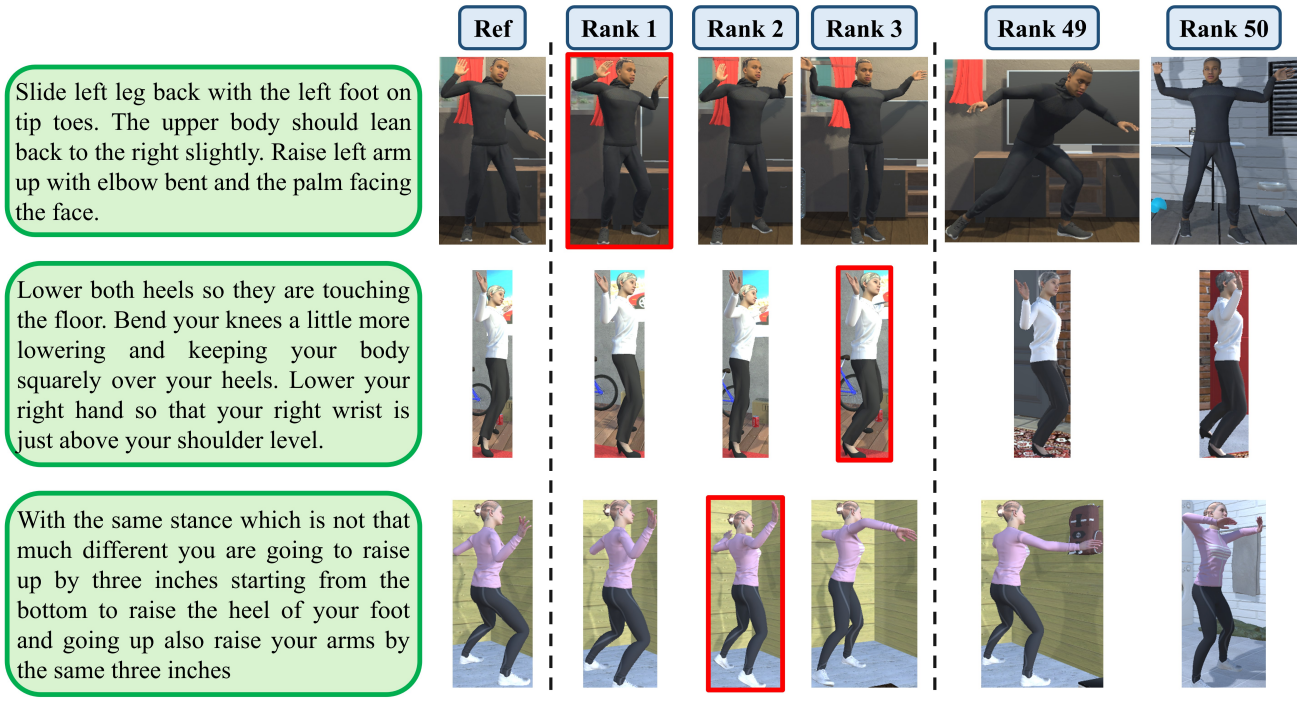


Figure 5.5: Additional CPR results on FIXMYPOSE. Ranked retrieval results are shown for a given reference image (Ref) and a transitional pose description (green box) as the query. Red boxes indicate the correct target images.

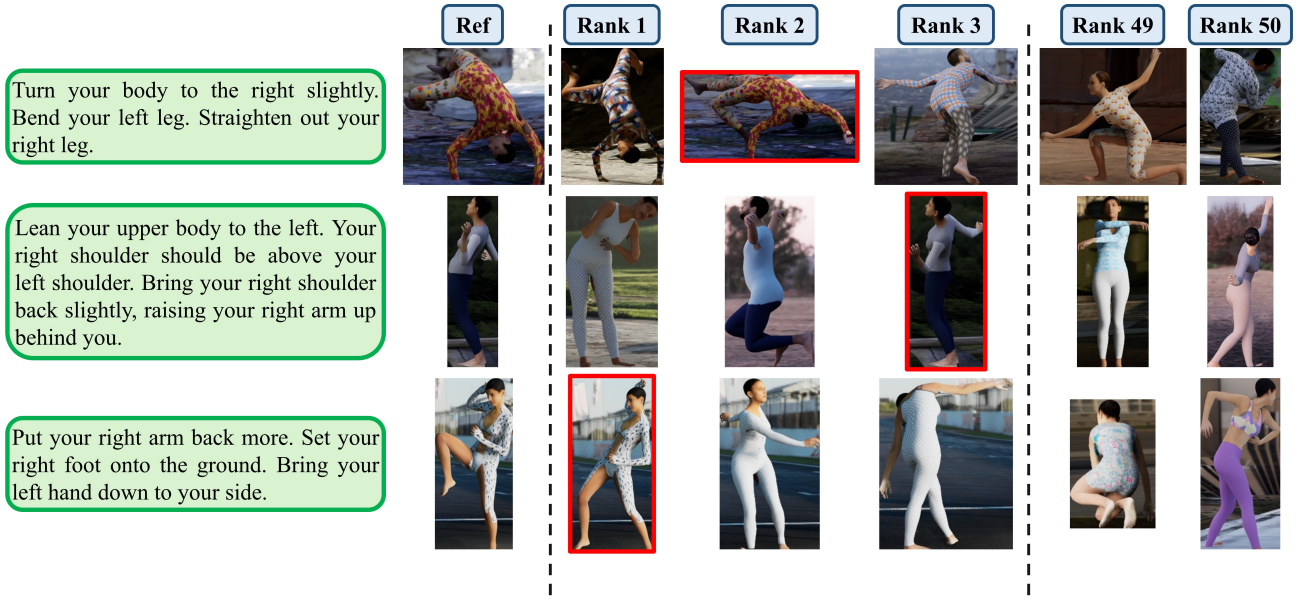


Figure 5.6: Additional CPR results on PoseFixCPR. Ranked retrieval results are shown for a given reference image (Ref) and a transitional pose description (green box) as the query. Red boxes indicate the correct target images.

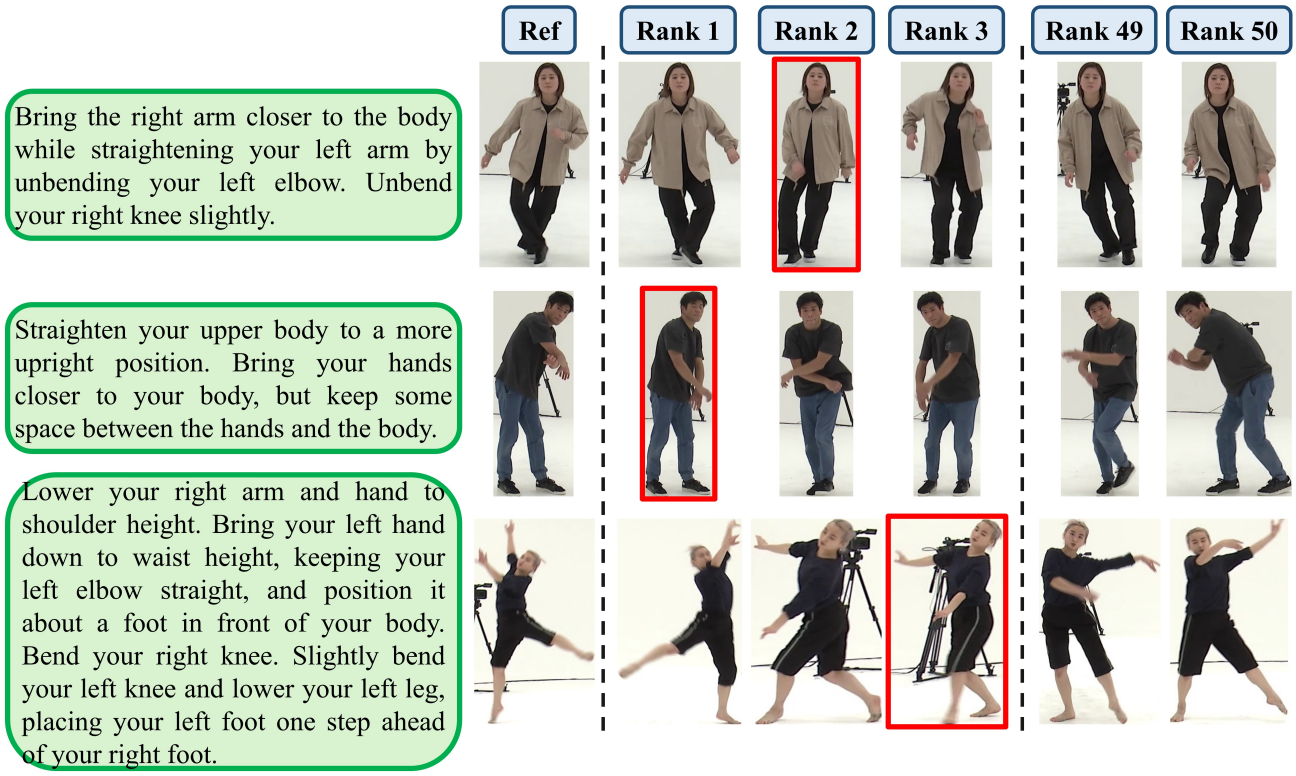


Figure 5.7: Additional CPR results on AIST-CPR. Ranked retrieval results are shown for a given reference image (Ref) and a transitional pose description (green box) as the query. Red boxes indicate the correct target images.

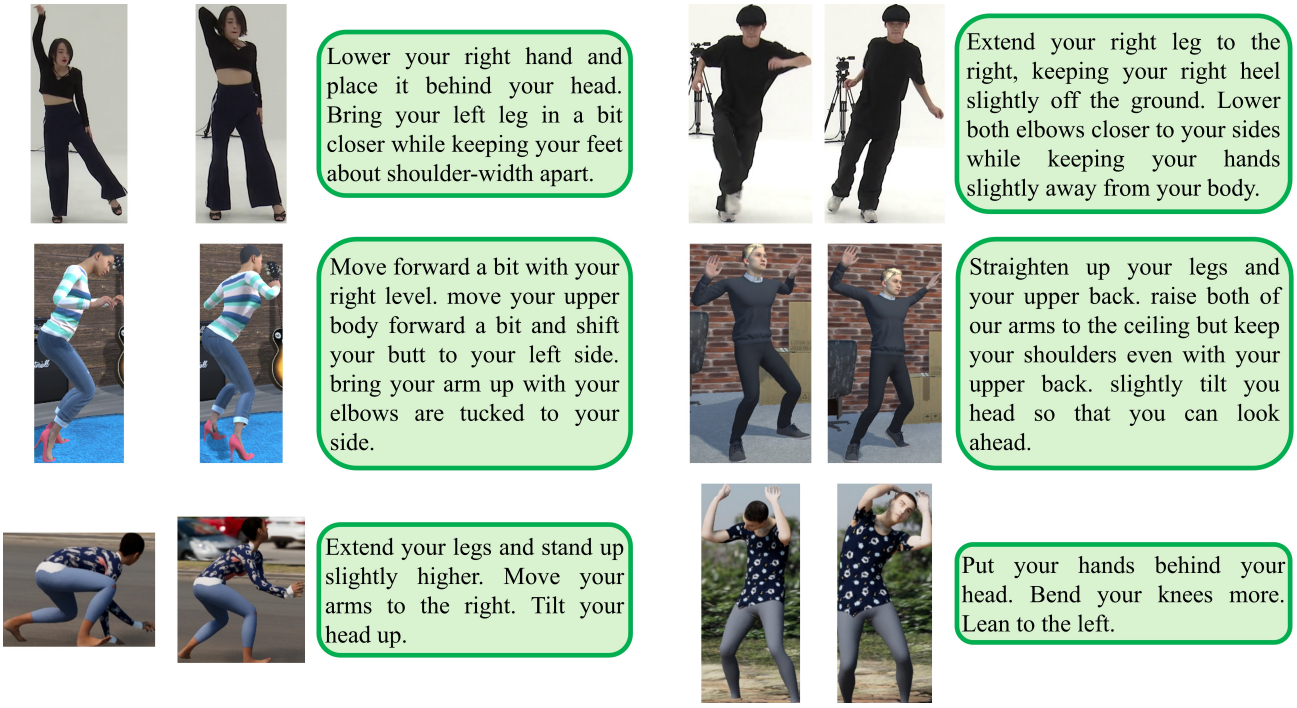


Figure 5.8: Example pose pairs from three datasets. The top, middle, and bottom rows correspond to AIST-CPR, FIXMYPOSE, and PoseFixCPR, respectively.

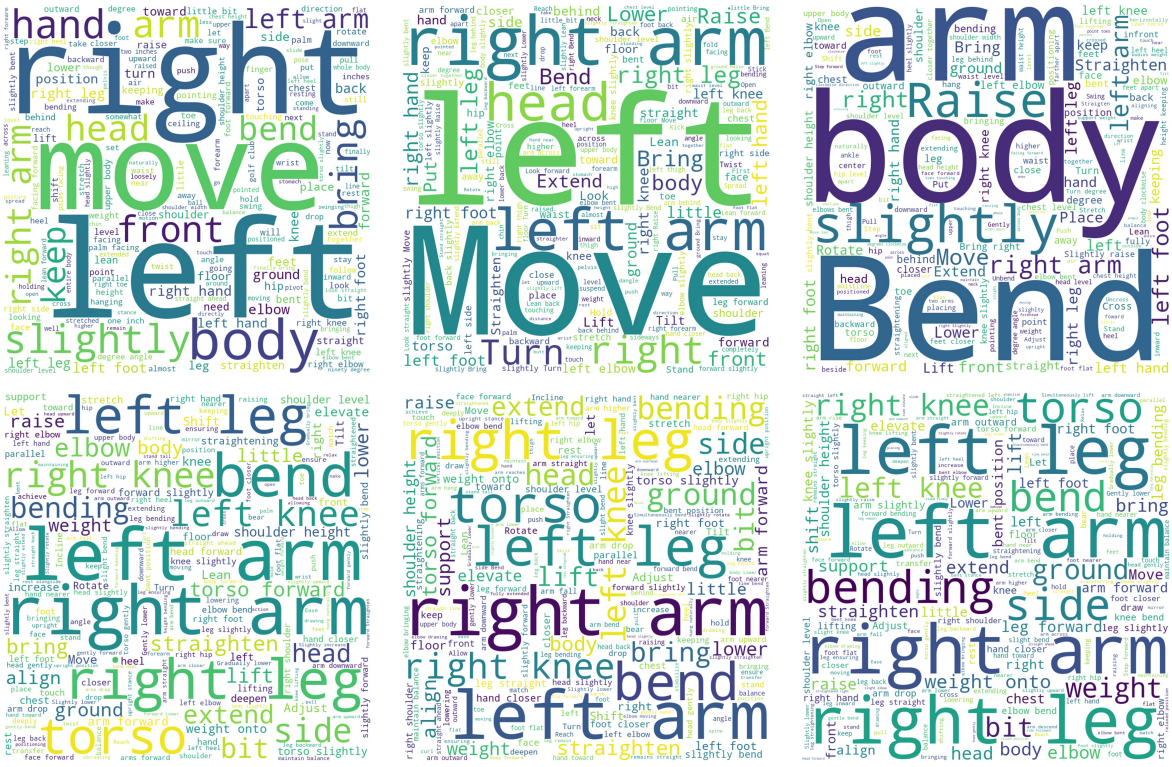


Figure 5.9: Word clouds generated from human-annotated texts (top) and AutoComPose-generated texts (bottom) across three datasets: FIXMYPOSE (left), PoseFixCPR (middle), and AIST-CPR (right).

body movements, we utilized the multimodal large language model (MLLM) [189] with *Prompt-1* (Fig. 5.10) to detect and filter out such descriptions. Three examples of environment-related descriptions were included in the prompt to guide the filtering process.

PoseFixCPR Rendering. We constructed PoseFixCPR by rendering 2D images (including masks) obtained from 3D pose pairs from PoseFix [157] using Unreal Engine [17] based on the BEDLAM [13] rendering pipeline. A subset of assets used for rendering—including body meshes, body and clothing UV texture maps, and high-dynamic range panoramic images (HDRIs) that used for image-based lighting—is shown in Fig. 5.11. A selection of output images is displayed in Fig. 5.12. Note that only the human regions are used for composed pose retrieval (CPR), which can be extracted using the generated masks.

5.6.3 MLLM Prompting Strategy

AutoComPose utilizes two prompts to communicate with the multimodal large language model (MLLM). *Prompt-2* (Fig. 5.13) is designed to guide the MLLM in generating body part-based pose transition descriptions. *Prompt-3* (Fig. 5.14) is used to helps the MLLM integrate these body part-based descriptions into more natural and complete sentences.

Prompt-1

You will be given an instruction guiding an individual to transition from their current position to a target position.
Your task is to determine whether the instruction includes any environmental direction descriptions—guiding the individual to move relative to a specific object nearby.
For example, the following instructions contain environmental direction descriptions, highlighted in quotation marks:

1. bring your right foot outwards at a 90 degree angle. bring your left and right hand up like a gun is being pointed at you ”keep your position facing the stereo system”.

2. ”lean your upper body forward to the plant pot” move the right leg to the right side a little bend the left leg a bit more move the right hand from waist level to beside the hip.

3. keep your feet and your hands on the floor as support push your arms up move your body and your legs up towards the ceiling ”lift up your head a little facing the rug”.

In contrast, the following instructions do not contain such descriptions:

1. position both your hands in front of your face is if you are grasping a pole. straighten the right leg so that the foot is flat on the floor. bend your left knee a little while pivoting on the ball of your foot. your head should face your hands.

2. first put your feet together take your right foot and put it slightly behind you. while keeping your lower body facing forward twist your upper body to the left. then hold your hands in the air at about the level of your head and turn your head to the left as well.

3. lift your right foot and pull it behind you. extend both of your hands gently outwards as if you are balancing on a surfboard while it is going under a big wave.

If the instruction contains environmental direction descriptions, respond with ”Yes.” Otherwise, respond with ”No.”
Provide only the requested answer (”Yes” or ”No”) with no additional text before or after.

Figure 5.10: Prompt used to remove environment-related descriptions in the FIXMYPOSE dataset.

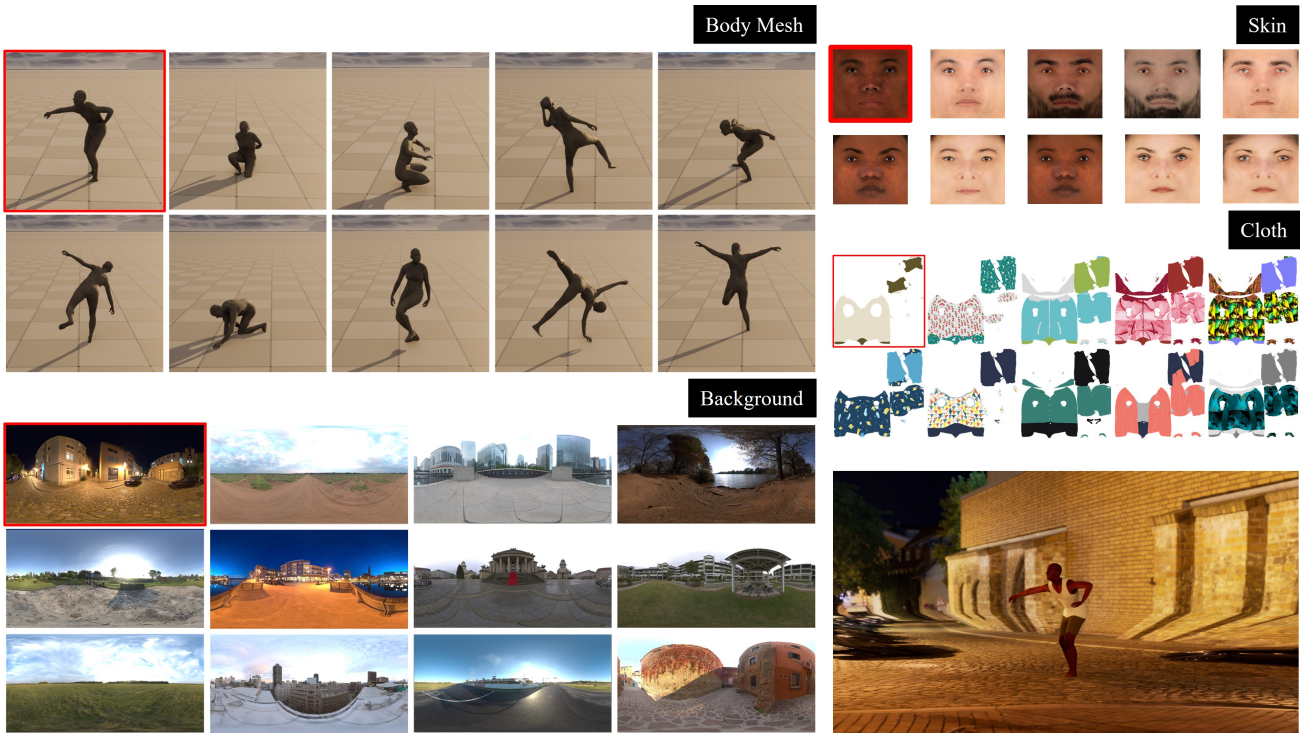


Figure 5.11: Sampled assets used for constructing PoseFixCPR. Highlighted assets (red boxes) were used to render the bottom-right image.



Figure 5.12: Sampled images from PoseFixCPR.

Prompt-2

You are a skilled fitness\dance instructor specializing in guiding individuals through pose transitions.
You will receive an image showing two poses side by side: the starting pose on the left and the target pose on the right.
Your task is to describe the transition from the starting pose to the target pose.
To achieve this:

- Compare the following body parts in the two poses: head, neck, left shoulder, right shoulder, left arm, right arm, left elbow, right elbow, left wrist, right wrist, left hand, right hand, torso, left hip, right hip, left leg, right leg, left knee, right knee, left ankle, right ankle, left foot, right foot.
- For each body part that requires adjustment, provide a clear and concise one-sentence description of the movement.
- Structure your description as follows: ”1. Right Arm: Lift the right arm above the head, transitioning smoothly from its resting position.” ”2. Left Leg: Straighten the left leg fully as it supports the body’s weight in standing position.”

Below are some important guidelines that must be followed:

- Mention as few body parts as possible; omit subtle or nearly imperceptible movements.
- If the person is facing the camera, the left hand of the person is defined as the hand on your (the viewer’s) right side.
- Provide only the requested descriptions of body part transitions, with no additional information before or after.

Figure 5.13: Prompt for generating body part-based pose transition descriptions.

Prompt-3

You will be given a set of bullet points describing a specific human pose transition.
Your task is to write five distinct, concise descriptions of the same pose transition. Structure your descriptions as follows: ”Description 1: [description].” ”Description 2: [description].”
Ensure your descriptions accurately capture the exact pose transition without altering it. To create variety, diversify your phrasing and avoid repeating expressions. You may incorporate analogies, such as ”lower your right arm to your side as if you’re holding a cane,” as long as the described pose transition remains unchanged.

Figure 5.14: Prompt for integrating and diversifying body part-based pose transition descriptions.

Modified Prompt-2

You are a skilled fitness\dance instructor specializing in guiding individuals through pose transitions.
You will receive an image showing two poses side by side: the starting pose on the left and the target pose on the right.
Your task is to describe the transition from the starting pose to the target pose.
Provide a clear and concise description of the transition. Structure your description as follows: "Description: [description]."
Below are some important guidelines that must be followed:

- Omit subtle or nearly imperceptible movements.
- If the person is facing the camera, the left hand of the person is defined as the hand on your (the viewer's) right side.
- Provide only the requested descriptions of transitions, with no additional information before or after.

Figure 5.15: Modified Prompt-2 used in the Stage I ablation study. The main modification, which removes body part-specific queries, is highlighted in red.

Modified Prompt-3

You will be given a description of a specific human pose transition.
Your task is to write five distinct, concise descriptions of the same pose transition. Structure your descriptions as follows: "Description 1: [description]." "Description 2: [description]."
Ensure your descriptions accurately capture the exact pose transition without altering it. To create variety, diversify your phrasing and avoid repeating expressions. You may incorporate analogies, such as "lower your right arm to your side as if you're holding a cane," as long as the described pose transition remains unchanged.

Figure 5.16: Modified Prompt-3 used in the Stage I ablation study. The main modification, which removes body part-specific queries, is highlighted in red.

Chapter 6: Conclusion and Future Work

6.1 Conclusion

This dissertation has tackled a fundamental challenge in human-centric vision: the scarcity of diverse, high-quality annotated data, especially in complex settings such as UAV-based detection and multimodal pose understanding. It demonstrates that data-centric approaches, including dataset curation, synthetic data utilization, and automated annotation, enable more systematic model assessment, significantly improve performance, and enhance scalability.

First, the dissertation introduced **Archangel** [24], a hybrid UAV-based human detection benchmark that integrates real, mannequin, and synthetic imagery with detailed metadata on camera viewpoints and human poses. Experimental results on Archangel show that incorporating real and synthetic data in a structured and appropriate manner is a promising way to improve detector accuracy. In addition, metadata-aware evaluation provides deeper insights into how models behave across various conditions, resulting in better architectural design.

Building on Archangel, the dissertation then proposed **Progressive Transformation Learning (PTL)** [26], an iterative sim-to-real framework for effectively utilizing synthetic data in detector training. By progressively selecting and transforming synthetic data based on measured domain discrepancies, quantified by Mahalanobis distance in the trained detector's representation space, PTL simultaneously enhances synthetic data realism, low-shot learning performance, and cross-domain generalization. This highlights the importance of carefully addressing domain gaps when synthetic data is incorporated into training pipelines.

Beyond PTL, the dissertation further introduced **SynPoseDiv** [27], a pose diversification framework for synthetic datasets. Although leveraging synthetic data in training can improve detection

performance, the diversity of the synthetic dataset, particularly pose diversity in human-centric vision, limits model improvement when additional synthetic data is used, especially in iterative approaches such as PTL. SynPoseDiv expands the coverage of plausible human poses in synthetic datasets by combining a diffusion-based pose generator and pose-guided image translation. The results demonstrate that improved pose diversity leads to better scalability and consistent performance gains, without the need to construct new synthetic datasets.

Finally, the dissertation addressed the challenge of scalable multimodal annotation in composed pose retrieval (CPR) through **AutoComPose** [28]. AutoComPose utilizes multimodal large language models (MLLMs) with carefully designed prompts to automatically generate structured and expressive pose transition descriptions suitable for CPR tasks. By introducing new benchmarks and a cyclic training strategy, AutoComPose demonstrates improved retrieval performance, reduces the need for costly manual annotations, and establishes a scalable pathway for advancing vision-language learning in human-centric applications.

In conclusion, this dissertation has highlighted that addressing data limitations is as crucial as model design in advancing human-centric vision. By systematically improving how data is constructed, enriched, and utilized, this work has provided practical and scalable solutions for challenging real-world applications. As research continues to progress toward more complex and multimodal scenarios, we believe that the contributions of this dissertation provide a strong foundation for future innovations in human-centric vision systems.

6.2 Broader Impact

The datasets and methods developed in this dissertation for UAV-based vision have the potential to improve detection systems, benefiting applications such as disaster response and search-and-rescue operations. The use of synthetic data to enhance cross-domain generalization may also enable faster deployment in new environments, a common scenario in real-world UAV missions. Additionally, Progressive Transformation Learning (PTL) provides a scalable framework for training detectors while accounting for domain gaps, with potential extensions beyond aerial-view human detection to tasks such as pose estimation and semantic segmentation.

For multimodal learning, automated annotation frameworks such as AutoComPose offer new opportunities for scaling vision-language datasets. These advancements could support a wide range of multimodal human-centric vision tasks, including pose retrieval, casualty identification, sports analysis, and human-computer interaction.

6.3 Limitations and Future Work

6.3.1 Generalization to More Diverse Synthetic and Real-World Datasets

Although Archangel enables more comprehensive and systematic model evaluation through its multiple subsets and detailed metadata, its overall diversity remains limited. For instance, Archangel-Synthetic includes only three categories of human poses (i.e., standing, squatting, and prone), which do not fully capture the richness and variability of human motion observed in real-world scenarios. As a result, it is crucial to further investigate whether the methodologies proposed in this dissertation, such as Progressive Transformation Learning (PTL), remain effective when applied to more complex

and diverse datasets.

Recent work such as SynPlay [16] provides a promising direction. SynPlay generates a significantly more diverse UAV-based synthetic dataset through a rule-guided motion generation framework, enabling richer pose variations and more realistic scene compositions. Empirical results suggest that, with increased data scale and pose diversity, PTL not only remains effective but can also outperform results obtained using both the original (Chapter 2) and pose-diversified (Chapter 4) versions of Archangel-Synthetic. These findings highlight the importance of dataset diversity in fully realizing the potential of synthetic data-driven learning.

In addition to limitations in synthetic dataset, the real-world datasets [9, 10, 11] used in this dissertation are also relatively constrained. They are primary collected from common, everyday scenarios with relatively clean visual conditions. However, many real-world UAV applications, such as search and rescue in disaster management, present much more challenging conditions, including cluttered backgrounds, severe occlusions, and degraded image quality [190]. Therefore, an important direction for future work is to evaluate and extend the proposed frameworks under more difficult and diverse real-world scenarios.

6.3.2 A Deeper Understanding of Progressive Transformation Learning

Although this dissertation demonstrates the effectiveness Progressive Transformation Learning (PTL) across various settings (Chapter 3), its underlying mechanisms remain insufficiently understood. Several fundamental questions remain open: which components contribute most to sim-to-real transformation? Can the observed performance improvements be sustained over a larger number of PTL iterations?

A preliminary study into these questions has been presented in [126], where the roles of three major factors in PTL-based sim-to-real transformation—real data, synthetic data, and synthetic data pool—are systematically investigated. This study offers several valuable insights into the use of synthetic data in detector training. However, a deeper and more comprehensive understanding is still needed. Such understanding is essential for improving the scalability of PTL and extending it to other tasks beyond aerial-view human detection, such as pose estimation or multimodal learning.

6.3.3 Controllable Composed Pose Retrieval

AutoComPose (Chapter 5) employs standard vision-language models [155] as the backbone for the retrieval task. While it demonstrates strong retrieval performance, one key limitation is that the model tends to rely on all visual factors present in the reference image, such as human pose, appearance, and viewing perspective, when retrieving target images. This behavior is not always desirable, particularly in applications where certain visual factors should be selectively ignored. For example, in multi-camera surveillance systems, the model may need to retrieve images that match a target pose and human appearance while being invariant to differences in viewing angle. Conversely, in sports analysis scenarios, it may be more important to match pose and camera perspective while allowing variation in appearance, enabling the analysis of different players under consistent viewpoints.

A promising direction for future work is the development of controllable CPR frameworks that allow users to dynamically adjust retrieval criteria without requiring retraining or fine-tuning. Such models would enable flexible selection of relevant factors, such as appearance or viewpoint, depending on the task requirements.

Achieving this goal requires advancements in both data and model design. First, new CPR

datasets that explicitly support disentangled representations of visual factors are required [191]. Constructing such datasets from the real-world is highly impractical due to the difficulty of independently varying each visual factor. Synthetic datasets could provide a more feasible solution for generating controlled variations in this context. In addition, retrieval models must be redesigned to support disentangled and controllable representations, which may involve learning factorized embeddings or incorporating explicit control signals that guide the retrieval process.

Bibliography

[1] Piotr Dollár, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: A benchmark. In *2009 IEEE conference on computer vision and pattern recognition*, pages 304–311. IEEE, 2009.

[2] Yufei Xu, Jing Zhang, Qiming Zhang, and Dacheng Tao. ViTPose: Simple vision transformer baselines for human pose estimation. *Advances in neural information processing systems*, 35: 38571–38584, 2022.

[3] Ginger Delmas, Philippe Weinzaepfel, Thomas Lucas, Francesc Moreno-Noguer, and Grégory Rogez. PoseScript: 3D human poses from natural language. In *ECCV*, 2022.

[4] Mickael Cormier, Aris Clepe, Andreas Specker, and Jürgen Beyerer. Where are we with human pose estimation in real-world surveillance? In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 591–601, 2022.

[5] Sasa Sambolek and Marina Ivasic-Kos. Automatic person detection in search and rescue operations using deep CNN detectors. *Ieee Access*, 9:37905–37922, 2021.

[6] Graham Thomas, Rikke Gade, Thomas B Moeslund, Peter Carr, and Adrian Hilton. Computer vision for sports: Current applications and research topics. *Computer Vision and Image Understanding*, 159:3–18, 2017.

[7] Chiao-Yi Wang, Faranguisse Kakhi Sadrieh, Yi-Ting Shen, Giovanni Oppizzi, Li-Qun Zhang, and Yang Tao. EgoFall: Real-time privacy-preserving fall risk assessment with a single on-body tracking camera. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 2025.

[8] Rawal Khirodkar, Timur Bagautdinov, Julieta Martinez, Su Zhaoen, Austin James, Peter Selednik, Stuart Anderson, and Shunsuke Saito. Sapiens: Foundation for human vision models. In *European Conference on Computer Vision*, pages 206–228. Springer, 2024.

[9] Pengfei Zhu, Longyin Wen, Dawei Du, Xiao Bian, Heng Fan, Qinghua Hu, and Haibin Ling. Detection and tracking meet drones challenge. *IEEE transactions on pattern analysis and machine intelligence*, 44(11):7380–7399, 2021.

[10] Mohammadamin Barekatain, Miquel Martí, Hsueh-Fu Shih, Samuel Murray, Kotaro Nakayama, Yutaka Matsuo, and Helmut Prendinger. Okutama-Action: An aerial view video dataset for concurrent human action detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 28–35, 2017.

[11] ICG, TU Graz. Semantic drone dataset. `http://dronedataset.icg.tugraz.at`, 2019.

[12] Zhenyu Wu, Karthik Suresh, Priya Narayanan, Hongyu Xu, Heesung Kwon, and Zhangyang Wang. Delving into robust object detection from unmanned aerial vehicles: A deep nuisance disentanglement approach. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1201–1210, 2019.

[13] Michael J. Black, Priyanka Patel, Joachim Tesch, and Jinlong Yang. BEDLAM: A synthetic dataset of bodies exhibiting detailed lifelike animated motion. In *Proceedings IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 8726–8737, June 2023.

[14] Joachim Tesch, Giorgio Becherini, Prerana Achar, Anastasios Yiannakidis, Muhammed Kocabas, Priyanka Patel, and Michael J. Black. BEDLAM2.0: Synthetic humans and cameras in motion. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025.

[15] Sahil Khose, Anisha Pal, Aayushi Agarwal, Deepanshi, Judy Hoffman, and Prithvijit Chattopadhyay. Skyscenes: A synthetic dataset for aerial scene understanding. In *European Conference on Computer Vision*, pages 19–35. Springer, 2024.

[16] Jinsub Yim, Hyungtae Lee, Sungmin Eum, Yi-Ting Shen, Yan Zhang, Heesung Kwon, and Shuvra S. Bhattacharyya. SynPlay: Large-scale synthetic human data with real-world diversity for aerial-view perception. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 938–947, 2026.

[17] Unreal Engine 5. `https://www.unrealengine.com/en-US`, 2025.

[18] Arthur Juliani, Vincent-Pierre Berges, Ervin Teng, Andrew Cohen, Jonathan Harper, Chris Elion, Chris Goy, Yuan Gao, Hunter Henry, Marwan Mattar, et al. Unity: A general platform for intelligent agents. *arXiv preprint arXiv:1809.02627*, 2018.

[19] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. CyCADA: Cycle-consistent adversarial domain adaptation. In *International conference on machine learning*, pages 1989–1998. Pmlr, 2018.

[20] Poojan Oza, Vishwanath A Sindagi, Vishal M Patel, et al. Unsupervised domain adaptation of object detectors: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(6):4018–4040, 2023.

[21] ChangHee Yang, Hyeonseop Song, Seokhun Choi, Seungwoo Lee, Jaechul Kim, and Hoseok Do. PoseSyn: Synthesizing diverse 3D pose data from in-the-wild 2D data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5611–5621, October 2025.

[22] Shaina Raza, Aravind Narayanan, Vahid Reza Khazaie, Ashmal Vayani, Ahmed Y Radwan, Mukund S Chettiar, Amandeep Singh, Mubarak Shah, and Deval Pandya. HumaniBench: A human-centric framework for large multimodal models evaluation. *arXiv preprint arXiv:2505.11454*, 2025.

[23] Jongseok Kim, Youngjae Yu, Hoeseong Kim, and Gunhee Kim. Dual compositional learning in interactive image retrieval. In *AAAI*, pages 1771–1779, 2021.

[24] Yi-Ting Shen, Yaesop Lee, Heesung Kwon, Damon M. Conover, Shuvra S. Bhattacharyya, Nikolas Vale, Joshua D. Gray, G. Jeremy Leongs, Kenneth Evensen, and Frank Skirlo. Archangel: A hybrid UAV-based human detection benchmark with position and pose metadata. *IEEE Access*, 11:80958–80972, 2023.

[25] Glenn Jocher, Ayush Chaurasia, Alex Stoken, Jirka Borovec, NanoCode012, Yonghye Kwon, TaoXie, Jiacong Fang, imyhxy, Kalen Michael, Lorna, Abhiram V, Diego Montes, Jebastin Nadar, Laughing, tkianai, yxNONG, Piotr Skalski, Zhiqiang Wang, Adam Hogan, Cristi Fati, Lorenzo Mammana, AlexWang1900, Deep Patel, Ding Yiwei, Felix You, Jan Hajek, Laurentiu Diaconu, and Mai Thanh Minh. ultralytics/yolov5: v6.1 - TensorRT, TensorFlow Edge TPU and OpenVINO export and inference, Feb 2022. URL `https://doi.org/10.5281/zenodo.6222936`.

[26] Yi-Ting Shen, Hyungtae Lee, Heesung Kwon, and Shuvra S. Bhattacharyya. Progressive transformation learning for leveraging virtual images in training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 835–844, 2023.

[27] Yi-Ting Shen, Hyungtae Lee, Heesung Kwon, and Shuvra S. Bhattacharyya. Diversifying human pose in synthetic data for aerial-view human detection. In *2025 IEEE International Conference on Image Processing (ICIP)*, pages 1648–1653. IEEE, 2025.

[28] Yi-Ting Shen, Sungmin Eum, Doheon Lee, Rohit Shete, Chiao-Yi Wang, Heesung Kwon, and Shuvra S. Bhattacharyya. AutoComPose: Automatic generation of pose transition descriptions for composed pose retrieval using multimodal LLMs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7409–7418, 2025.

[29] Milan Erdelj and Enrico Natalizio. UAV-assisted disaster management: Applications and open issues. In *2016 International Conference on Computing, Networking and Communications (ICNC)*, pages 1–5, 2016. doi: 10.1109/ICCNC.2016.7440563.

[30] Amira Chriki, Haifa Touati, Hichem Snoussi, and Farouk Kamoun. UAV-based surveillance system: An anomaly detection approach. In *2020 IEEE Symposium on computers and communications (ISCC)*, pages 1–6. IEEE, 2020.

[31] Mohammed Yaqot and Brenno C Menezes. Unmanned aerial vehicle (UAV) in precision agriculture: Business information technology towards farming as a service. In *2021 1st International Conference on Emerging Smart Technologies and Applications (eSmarTA)*, pages 1–7. IEEE, 2021.

[32] Muhammad Arsalan Khan, Wim Ectors, Tom Bellemans, Yassine Ruichek, Davy Janssens, Geert Wets, et al. Unmanned aerial vehicle-based traffic analysis: A case study to analyze traffic streams at urban roundabouts. *Procedia computer science*, 130:636–643, 2018.

[33] Hamid Menouar, Ismail Guvenc, Kemal Akkaya, A Selcuk Uluagac, Abdullah Kadri, and Adem Tuncer. UAV-enabled intelligent transportation systems for the smart city: Applications and challenges. *IEEE Communications Magazine*, 55(3):22–28, 2017.

[34] Jia Deng, Wei Dong, Socher Richard, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. doi: 10.1109/CVPR.2009.5206848.

[35] Mark Everingham, SM Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111(1):98–136, 2015.

[36] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[37] Hongyang Yu, Guorong Li, Weigang Zhang, Qingming Huang, Dawei Du, Qi Tian, and Nicu Sebe. The unmanned aerial vehicle benchmark: Object detection, tracking and baseline. *International Journal of Computer Vision*, 128(5):1141–1159, 2020.

[38] Alexandre Robicquet, Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Learning social etiquette: Human trajectory understanding in crowded scenes. In *European conference on computer vision*, pages 549–565. Springer, 2016.

[39] Yaesop Lee, Eung-Joo Lee, Damon M Conover, Yi-Ting Shen, Heesung Kwon, Shuvra S. Bhattacharyya, Jason Hill, Kenneth Evensen, and G Jeremy Leong. Archangel dataset: UAV-based imagery with position and pose metadata. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications IV*, volume 12113, pages 544–557. SPIE, 2022.

[40] Eung-Joo Lee, Damon M Conover, Shuvra S. Bhattacharyya, Heesung Kwon, Jason Hill, and Kenneth Evensen. Validation of object detection in UAV-based images using synthetic data. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications III*, volume 11746, pages 584–601. SPIE, 2021.

[41] Sergey I Nikolenko. *Synthetic data for deep learning*, volume 174. Springer, 2021.

[42] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, Tom Duerig, and Vittorio Ferrari. The Open Images Dataset V4: Unified image classification, object detection, and visual relationship detection at scale. *IJCV*, 2020.

[43] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2446–2454, 2020.

[44] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE, 2012.

[45] Shuai Shao, Zeming Li, Tianyuan Zhang, Chao Peng, Gang Yu, Xiangyu Zhang, Jing Li, and Jian Sun. Objects365: A large-scale, high-quality dataset for object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 8430–8439, 2019.

[46] Priya Narayanan, Xin Hu, Zhenyu Wu, Matthew D Thielke, John G Rogers, Andre V Harrison, John A D'Agostino, James D Brown, Long P Quang, James R Uplinger, et al. A multipurpose real haze benchmark with quantifiable haze levels and ground truth. *arXiv preprint arXiv:2206.06427*, 2022.

[47] UG^2 Prize Challenge. UG^2 Prize Challenge. `http://cvpr2022.UG2challenge.org/index.html`, 2022. Accessed: 2022-07-25.

[48] Benjamin Kiefer, David Ott, and Andreas Zell. Leveraging synthetic data in object detection on unmanned aerial vehicles. In *2022 26th International Conference on Pattern Recognition (ICPR)*, pages 3564–3571. IEEE, 2022.

[49] Jian Ding, Nan Xue, Gui-Song Xia, Xiang Bai, Wen Yang, Michael Yang, Serge Belongie, Jiebo Luo, Mihai Datcu, Marcello Pelillo, et al. Object detection in aerial images: A large-scale benchmark and challenges. *IEEE transactions on pattern analysis and machine intelligence*, 2021.

[50] Meng-Ru Hsieh, Yen-Liang Lin, and Winston H Hsu. Drone-based object counting by spatially regularized regional proposal network. In *Proceedings of the IEEE international conference on computer vision*, pages 4145–4153, 2017.

[51] Murari Mandal, Lav Kush Kumar, and Santosh Kumar Vipparthi. MOR-UAV: A benchmark dataset and baselines for moving object recognition in UAV videos. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 2626–2635, 2020.

[52] Matthias Mueller, Neil Smith, and Bernard Ghanem. A benchmark and simulator for UAV tracking. In *European conference on computer vision*, pages 445–461. Springer, 2016.

[53] Isha Kalra, Maneet Singh, Shruti Nagpal, Richa Singh, Mayank Vatsa, and PB Sujit. DroneSurf: Benchmark dataset for drone-based face recognition. In *2019 14th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2019)*, pages 1–7. IEEE, 2019.

[54] Ilker Bozcan and Erdal Kayacan. AU-AIR: A multi-modal unmanned aerial vehicle dataset for low altitude traffic surveillance. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8504–8510. IEEE, 2020.

[55] Tianjiao Li, Jun Liu, Wei Zhang, Yun Ni, Wenqian Wang, and Zhiheng Li. UAV-Human: A large benchmark for human behavior understanding with unmanned aerial vehicles. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16266–16275, 2021.

[56] Leon Amadeus Varga, Benjamin Kiefer, Martin Messmer, and Andreas Zell. SeaDronesSee: A maritime benchmark for detecting humans in open water. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2260–2270, 2022.

[57] Syed Sahil Abbas Zaidi, Mohammad Samar Ansari, Asra Aslam, Nadia Kanwal, Mamoona Asghar, and Brian Lee. A survey of modern deep learning based object detection models. *Digital Signal Processing*, page 103514, 2022.

[58] Xin Wu, Wei Li, Danfeng Hong, Ran Tao, and Qian Du. Deep learning for unmanned aerial vehicle-based object detection and tracking: A survey. *IEEE Geoscience and Remote Sensing Magazine*, 10(1):91–124, 2021.

[59] Xuehui Yu, Zhenjun Han, Yuqi Gong, Nan Jan, Jian Zhao, Qixiang Ye, Jie Chen, Yuan Feng, Bin Zhang, Xiaodi Wang, et al. The 1st tiny object detection challenge: Methods and results. In *European Conference on Computer Vision*, pages 315–323. Springer, 2020.

[60] Ziming Liu, Guangyu Gao, Lin Sun, and Zhiyuan Fang. HRDNet: High-resolution detection network for small objects. In *2021 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2021.

[61] Yingjie Liu, Fengbao Yang, and Peng Hu. Small-object detection in UAV-captured images via multi-branch parallel feature pyramid networks. *IEEE access*, 8:145740–145750, 2020.

[62] Fan Yang, Heng Fan, Peng Chu, Erik Blasch, and Haibin Ling. Clustered object detection in aerial images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8311–8320, 2019.

[63] Nassim Ammour, Haikel Alhichri, Yakoub Bazi, Bilel Benjdira, Naif Alajlan, and Mansour Zuair. Deep learning approach for car detection in UAV imagery. *Remote Sensing*, 9(4):312, 2017.

[64] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. YOLOv4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[65] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2):318–327, 2020. doi: 10.1109/TPAMI.2018.2858826.

[66] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. CenterNet: Keypoint triplets for object detection. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6568–6577, 2019. doi: 10.1109/ICCV.2019.00667.

[67] Mingxing Tan, Ruoming Pang, and Quoc V. Le. EfficientDet: Scalable and efficient object detection. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10778–10787, 2020. doi: 10.1109/CVPR42600.2020.01079.

[68] Mingjie Liu, Xianhao Wang, Anjian Zhou, Xiuyuan Fu, Yiwei Ma, and Changhao Piao. UAV-YOLO: Small object detection on unmanned aerial vehicle perspective. *Sensors*, 20(8):2238, 2020.

[69] U3DC. Ml-ImageSynthesis, 2017. URL `https://github.com/U3DC/Image-Synthesis-for-Machine-Learning`.

[70] Alon Brutzkus and Amir Globerson. Why do larger models generalize better? A theoretical perspective via the XOR problem. In *International Conference on Machine Learning*, pages 822–830. PMLR, 2019.

[71] Ying Huang, Bin Sun, Haipeng Kan, Jiankai Zhuang, and Zengchang Qin. FollowMeUp Sports: New benchmark for 2D human keypoint recognition. In *Chinese Conference on Pattern Recognition and Computer Vision (PRCV)*, pages 110–121. Springer, 2019.

[72] Jorg Bornschein, Francesco Visin, and Simon Osindero. Small data, big decisions: Model selection in the small-data regime. In *International conference on machine learning*, pages 1035–1044. PMLR, 2020.

[73] Viktor Seib, Benjamin Lange, and Stefan Wirtz. Mixing real and synthetic data to enhance neural network training - A review of current approaches. *arXiv preprint arXiv:2007.08781*, 2020.

[74] Kaiyang Zhou, Ziwei Liu, Yu Qiao, Tao Xiang, and Chen Change Loy. Domain generalization: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[75] Yen Nhi Truong Vu, Richard Wang, Niranjan Balachandar, Can Liu, Andrew Y Ng, and Pranav Rajpurkar. MedAug: Contrastive learning leveraging patient metadata improves representations for chest x-ray interpretation. In *Machine Learning for Healthcare Conference*, pages 755–769. PMLR, 2021.

[76] Takaki Yamada, Miquel Massot-Campos, Adam Prügel-Bennett, Stefan B Williams, Oscar Pizarro, and Blair Thornton. Leveraging metadata in representation learning with georeferenced seafloor imagery. *IEEE Robotics and Automation Letters*, 6(4):7815–7822, 2021.

[77] Lanlan Liu, Michael Muelly, Jia Deng, Tomas Pfister, and Li-Jia Li. Generative modeling for small-data object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6073–6081, 2019.

[78] Amlan Kar, Aayush Prakash, Ming-Yu Liu, Eric Cameracci, Justin Yuan, Matt Rusiniak, David Acuna, Antonio Torralba, and Sanja Fidler. Meta-Sim: Learning to generate synthetic datasets. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4551–4560, 2019.

[79] Jiteng Mu, Weichao Qiu, Gregory Hager, and Alan Yuille. Learning from synthetic animals. In *Proc. CVPR*, 2020.

[80] Chen Li and Gim Hee Lee. From synthetic to real: Unsupervised domain adaptation for animal pose estimation. In *Proc. CVPR*, 2021.

[81] Haibo Qiu, Baosheng Yu, Dihong Gong, Zhifeng Li, Wei Liu, and Dacheng Tao. SynFace: Face recognition with synthetic data. In *Proc. ICCV*, 2021.

[82] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. arXiv:1411.1784, 2014. URL `https://arxiv.org/abs/1411.1784`.

[83] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Proc. NeurIPS*, 2018.

[84] Prasanta Chandra Mahalanobis. On the generalised distance in statistics. *Proc. National Institute of Sciences of India.*, 1936.

[85] Ankur Handa, Viorica Pătrăucean, Vijay Badrinarayanan, Simon Stent, and Roberto Cipolla. SceneNet: Understanding real world indoor scenes with synthetic data. In *Proc. CVPR*, 2016.

[86] Dahun Kim, Sanghyun Woo, Joon-Young Lee, and In So Kweon. Video panoptic segmentation. In *Proc. CVPR*, 2016.

[87] Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. Virtual worlds as proxy for multi-object tracking analysis. In *Proc. CVPR*, 2016.

[88] Matteo Fabbri, Fabio Lanzi, Simone Calderara, Andrea Palazzi, Roberto Vezzani, and Rita Cucchiara. Learning to detect and track visible and occluded body joints in a virtual world. In *Proc. ECCV*, 2018.

[89] Gul Varol, Javier Romero, Xavier Martin, Naureen Mahmood, Michael J. Black, Ivan Laptev, and Cordelia Schmid. Learning from synthetic humans. In *Proc. CVPR*, 2018.

[90] Samarth Mishra, Rameswar Panda, Cheng Perng Phoo, Chun-Fu (Richard) Chen, Leonid Karlinsky, Kate Saenko, Venkatesh Saligrama, and Rogerio S. Feris. Task2Sim: Towards effective pre-training and transfer from synthetic data. In *Proc. CVPR*, 2022.

[91] Xi Guo, Wei Wu, Dongliang Wang, Jing Su, Haisheng Su, Weihao Gan, Jian Huang, and Qin Yang. Learning video representations of human motion from synthetic data. In *Proc. CVPR*, 2022.

[92] Zhao Jin, Yinjie Lei, Naveed Akhtar, Haifeng Li, and Munawar Hayat. Deformation and correspondence aware unsupervised synthetic-to-real scene flow estimation for point clouds. In *Proc. CVPR*, 2022.

[93] Kyungjune Baek and Hyunjung Shim. Commonality in natural images rescues GANs: Pretraining GANs with generic and privacy-free synthetic data. In *Proc. CVPR*, 2022.

[94] Qing Liu, Adam Kortylewski, Zhishuai Zhang, Zizhang Li, Mengqi Guo, Qihao Liu, Xiaoding Yuan, Jiteng Mu, Weichao Qiu, and Alan Yuille. Learning part segmentation through unsupervised domain adaptation from synthetic vehicles. In *Proc. CVPR*, 2022.

[95] Qi Yan, Jianhao Zheng, Simon Reding, Shanci Li, and Iordan Doytchinov. CrossLoc: Scalable aerial localization assisted by multimodal synthetic data. In *Proc. CVPR*, 2022.

[96] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M. Lopez. The SYNTHIA dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proc. CVPR*, 2016.

[97] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *Proc. ECCV*, 2016.

[98] Scott E. Fahlman and Christian Lebiere. The cascade-correlation learning architecture. In *Proc. NeurIPS*, 1989.

[99] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Proc. NeurIPS*, 2006.

[100] Leslie N. Smith, Emily M. Hand, and Timothy Doster. Gradual DropIn of layers to train very deep neural networks. In *Proc. CVPR*, 2016.

[101] Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2Net: Accelerating learning via knowledge transfer. In *Proc. ICLR*, 2016.

[102] Tao Wei, Changhu Wang, Yong Rui, and Chang Wen Chen. Network morphism. In *Proc. ICML*, 2016.

[103] Linyuan Gong, Di He, Zhuohan Li, Tao Qin, Liwei Wang, and Tie-Yan Liu. Efficient training of BERT by progressively stacking. In *Proc. ICML*, 2019.

[104] Changlin Li, Bohan Zhuang, Guangrun Wang, Xiaodan Liang, Xiaojun Chang, and Yi Yang. Automated progressive learning for efficient training of vision transformers. In *Proc. CVPR*, 2022.

[105] Minjia Zhang and Yuxiong He. Accelerating training of transformer-based language models with progressive layer dropping. In *Proc. NeurIPS*, 2020.

[106] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *Proc. ICLR*, 2018.

[107] Yoshua Bengio, Jérŏme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proc. ICML*, 2009.

[108] Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. Baby Steps: How "Less is More" in unsupervised dependency parsing. In *Proc. NeurIPS Workshop*, 2009.

[109] Faisal Khan, Bilge Mutlu, and Jerry Zhu. How do humans teach: On curriculum learning and teaching dimension. In *Proc. NeurIPS*, 2011.

[110] Alex Graves, Marc G Bellemare, Jacob Menick, Remi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks. In *Proc. ICML*, 2017.

[111] Guangcong Wang, Xiaohua Xie, Jianhuang Lai, and Jiaxuan Zhuo. Deep growing learning. In *Proc. ICCV*, 2017.

[112] Can Wang, Sheng Jin, Yingda Guan, Wentao Liu, Chen Qian, Ping Luo, and Wanli Ouyang. Pseudo-labeled auto-curriculum learning for semi-supervised keypoint localization. In *Proc. ICLR*, 2022.

[113] Tianyue Cao, Yongxin Wang, Yifan Xing, Tianjun Xiao, Tong He, Zheng Zhang, Hao Zhou, and Joseph Tighe. PSS: Progressive sample selection for open-world visual representation learning. In *Proc. ECCV*, 2022.

[114] M. Pawan Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In *Proc. NeurIPS*, 2010.

[115] Prannay Kaul, Weidi Xie, and Andrew Zisserman. Label, verify, correct: A simple few shot object detection method. In *Proc. CVPR*, 2022.

[116] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proc. CVPR*, 2020.

[117] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Proc. ICML*, 2020.

[118] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent - A new approach to self-supervised learning. In *Proc. NeurIPS*, 2020.

[119] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. In *Proc. CVPR*, 2017.

[120] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[121] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single shot multibox detector. In *Proc. ECCV*, 2016.

[122] Chuyi Li, Lulu Li, Hongliang Jiang, Kaiheng Weng, Yifei Geng, Liang Li, Zaidan Ke, Qingyuan Li, Meng Cheng, Weiqiang Nie, Yiduo Li, Bo Zhang, Yufei Liang, Linyuan Zhou, Xiaoming Xu, Xiangxiang Chu, Xiaoming Wei, and Xiaolin Wei. YOLOv6: A single-stage object detection framework for industrial applications. arXiv:2209.02976, 2022. URL `https://arxiv.org/abs/2209.02976`.

[123] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proc. CVPR*, 2023.

[124] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. CycleGAN and pix2pix in PyTorch. `https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix`, 2017.

[125] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

[126] Hyungtae Lee, Yan Zhang, Yi-Ting Shen, Heesung Kwon, and Shuvra S. Bhattacharyya. Exploring the impact of synthetic data for aerial-view human detection. *arXiv preprint arXiv:2405.15203*, 2024.

[127] Zhengxia Zou, Keyan Chen, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object detection in 20 years: A survey. *Proc. of the IEEE*, 111(3):257–276, 2023.

[128] Guy Tevet, Sigal Raab, Brian Gordon, Yonatan Shafir, Daniel Cohen-Or, and Amit H. Bermano. Human motion diffusion model. In *Proc. ICLR*, 2023.

[129] Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numer. Math.*, 1: 269–271, December 1959.

[130] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3.6M: Large scale datasets and predictive methods for 3D human sensing in natural environments. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(7):1325–1339, July 2014.

[131] Valentin Gabeur, Jean-Sebastien Franco, Xavier Martin, Cordelia Schmid, and Gregory Rogez. Moulding humans: Non-parametric 3D human shape estimation from single images. In *Proc. ICCV*, 2019.

[132] Mihai Zanfir, Elisabeta Oneata, Alin-Ionut Popa, Andrei Zanfir, and Cristian Sminchisescu. Human synthesis and scene compositing. In *Proc. AAAI*, 2020.

[133] Zigang Geng, Chunyu Wang, Yixuan Wei, Ze Liu, Houqiang Li, and Han Hu. Human pose as compositional tokens. In *Proc. CVPR*, 2023.

[134] Hai Ci, Mingdong Wu, Wentao Zhu, Xiaoxuan Ma, Hao Dong, Fangwei Zhong, and Yizhou Wang. GFPose: Learning 3D human pose prior with gradient fields. In *Proc. CVPR*, 2023.

[135] Andrey Davydov, Anastasia Remizova, Victor Constantin, Sina Honari, Mathieu Salzmann, and Pascal Fua. Adversarial parametric pose prior. In *Proc. CVPR*, 2022.

[136] Nannan Li, Kevin J. Shih, and Bryan A. Plummer. Collecting the puzzle pieces: Disentangled self-driven human pose transfer by permuting textures. In *Proc. ICCV*, 2023.

[137] Xiao Han, Xiatian Zhu, Jiankang Deng, Yi-Zhe Song, and Tao Xiang. Controllable person image synthesis with pose-constrained latent diffusion. In *Proc. ICCV*, 2023.

[138] Youxin Pang, Yong Zhang, Weize Quan, Yanbo Fan, Xiaodong Cun, Ying Shan, and Dong-Ming Yan. DPE: Disentanglement of pose and expression for general video portrait editing. In *Proc. CVPR*, 2023.

[139] Yurui Ren, Xiaoqing Fan, Ge Li, Shan Liu, and Thomas H. Li. Neural texture extraction and distribution for controllable person image synthesis. In *CVPR*, 2022.

[140] Roman Suvorov, Elizaveta Logacheva, Anton Mashikhin, Anastasia Remizova, Arsenii Ashukha, Aleksei Silvestrov, Naejin Kong, Harshith Goka, Kiwoong Park, and Victor Lempitsky. Resolution-robust large mask inpainting with fourier convolutions. In *Proc. WACV*, 2022.

[141] Amin Ansarian and Maria A Amer. Realistic augmentation for effective 2D human pose estimation under occlusion. In *ICIP*, 2021.

[142] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *CVPR*, 2017.

[143] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local nash equilibrium. In *NeurIPS*, 2017.

[144] Dario Pavllo, Christoph Feichtenhofer, David Grangier, and Michael Auli. 3D human pose estimation in video with temporal convolutions and semi-supervised training. In *Proc. CVPR*, 2019.

[145] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. DeepFashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proc. CVPR*, 2016.

[146] Greg Mori, Caroline Pantofaru, Nisarg Kothari, Thomas Leung, George Toderici, Alexander Toshev, and Weilong Yang. Pose embeddings: A deep architecture for learning to match human poses, 2015. URL https://arxiv.org/abs/1507.00302.

[147] Tomas Jenicek and Ondřej Chum. Linking art through human poses. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1338–1345. IEEE, 2019.

[148] Yao Feng, Jing Lin, Sai Kumar Dwivedi, Yu Sun, Priyanka Patel, and Michael J. Black. ChatPose: Chatting about 3D human pose. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[149] Ge Wang, Fei Yu, Jian Li, Qi Jia, and Shouhong Ding. Adaptive uncertainty-based learning for text-based person retrieval. In *AAAI*, 2024.

[150] Alberto Baldrati, Marco Bertini, Tiberio Uricchio, and Alberto Del Bimbo. Composed image retrieval using contrastive learning and task-oriented clip-based features. *ACM Transactions on Multimedia Computing, Communications and Applications*, 20(3):1–24, 2023.

[151] Zheyuan Liu, Cristian Rodriguez-Opazo, Damien Teney, and Stephen Gould. Image retrieval on real-life images with pre-trained vision-and-language models. In *ICCV*, pages 2125–2134, 2021.

[152] Hui Wu, Yupeng Gao, Xiaoxiao Guo, Ziad Al-Halah, Steven Rennie, Kristen Grauman, and Rogerio Feris. Fashion IQ: A new dataset towards retrieving images by natural language feedback. In *CVPR*, pages 11307–11317, 2021.

[153] Kuniaki Saito, Kihyuk Sohn, Xiang Zhang, Chun-Liang Li, Chen-Yu Lee, Kate Saenko, and Tomas Pfister. Pic2Word: Mapping pictures to words for zero-shot composed image retrieval. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19305–19314, 2023.

[154] Young Kyun Jang, Donghyun Kim, Zihang Meng, Dat Huynh, and Ser-Nam Lim. Visual delta generator with large multi-modal models for semi-supervised composed image retrieval. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16805–16814, 2024.

[155] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2021.

[156] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. BLIP: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International conference on machine learning*, pages 12888–12900. PMLR, 2022.

[157] Ginger Delmas, Philippe Weinzaepfel, Francesc Moreno-Noguer, and Grégory Rogez. PoseFix: correcting 3D human poses with natural language. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15018–15028, 2023.

[158] Jiayang Wu, Wensheng Gan, Zefeng Chen, Shicheng Wan, and S Yu Philip. Multimodal large language models: A survey. In *2023 IEEE International Conference on Big Data (BigData)*, pages 2247–2256. IEEE, 2023.

[159] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26296–26306, 2024.

[160] Shengqiong Wu, Hao Fei, Leigang Qu, Wei Ji, and Tat-Seng Chua. NExT-GPT: Any-to-any multimodal LLM. In *Forty-first International Conference on Machine Learning*, 2024.

[161] Zechen Bai, Pichao Wang, Tianjun Xiao, Tong He, Zongbo Han, Zheng Zhang, and Mike Zheng Shou. Hallucination of multimodal large language models: A survey. *arXiv preprint arXiv:2404.18930*, 2024.

[162] Ruilong Li, Shan Yang, David A Ross, and Angjoo Kanazawa. AI choreographer: Music conditioned 3D dance generation with AIST++. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 13401–13412, 2021.

[163] Vittorio Ferrari, Manuel Marin-Jimenez, and Andrew Zisserman. Pose search: Retrieving people using their pose. In *CVPR*, 2009.

[164] Hanyu Wang, Junhui Guo, Deyou Ma Yan, Weimin Quan, and Xinge Zhang. Learning 3D keypoint descriptors for non-rigid shape matching. In *ECCV*, 2018.

[165] Georgios Pavlakos, Xiaowei Zhou, and Kostas Daniilidis. Ordinal depth supervision for 3D human pose estimation. In *CVPR*, 2018.

[166] Haokun Wen, Xuemeng Song, Xiaolin Chen, Yinwei Wei, Liqiang Nie, and Tat-Seng Chua. Simple but effective raw-data level multimodal fusion for composed image retrieval. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 229–239, 2024.

[167] Alberto Baldrati, Marcella Cornia, Lorenzo Baraldi, and Rita Cucchiara. Zero-shot composed image retrieval with textual inversion. In *CVPR*, 2023.

[168] Shu Zhao and Huijuan Xu. NEUCORE: Neural concept reasoning for composed image retrieval. In *Proceedings of UniReps: the First Workshop on Unifying Representations in Neural Models*, pages 47–59. PMLR, 2024.

[169] Xuemeng Song, Haoqiang Lin, Haokun Wen, Bohan Hou, Mingzhu Xu, and Liqiang Nie. A comprehensive survey on composed image retrieval. *ACM Transactions on Information Systems*, 1(1):1–45, 2024.

[170] Jiannan Wu, Muyan Zhong, Sen Xing, Zeqiang Lai, Zhaoyang Liu, Wenhai Wang, Zhe Chen, Xizhou Zhu, Lewei Lu, Ping Luo, et al. VisionLLM v2: An end-to-end generalist multimodal large language model for hundreds of vision-language tasks. *NeurIPS*, 2024.

[171] Jieneng Chen, Luoxin Ye, Ju He, Zhao-Yang Wang, Daniel Khashabi, and Alan Yuille. Efficient large multi-modal models via visual context compression. In *NeurIPS*, 2024.

[172] Yifan Zhang, Cheng Huang, and Limin Wang. MotionGPT: Large language model for human motion generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[173] Zhen Tan, Alimohammad Beigi, Song Wang, Ruocheng Guo, Amrita Bhattacharjee, Bohan Jiang, Mansooreh Karami, Jundong Li, Lu Cheng, and Huan Liu. Large language models for data annotation and synthesis: A survey. *EMNLP*, 2024.

[174] Hongjin Su, Jungo Kasai, Chen Henry Wu, Weijia Shi, Tianlu Wang, Jiayi Xin, Rui Zhang, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Tao Yu. Selective annotation makes language models better few-shot learners. In *ICLR*, 2023.

[175] Ruirui Chen, Chengwei Qin, Weifeng Jiang, and Dongkyu Choi. Is a large language model a good annotator for event extraction? In *AAAI*, 2024.

[176] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2D pose estimation using part affinity fields. In *CVPR*, 2017.

[177] Riza Alp Güler, Natalia Neverova, and Iasonas Kokkinos. DensePose: Dense human pose estimation in the wild. In *CVPR*, 2018.

[178] Mathis Petrovich, Michael Black, and Gül Varol. Action-conditioned 3D human motion synthesis with transformer VAE. In *ICCV*, 2021.

[179] Connor Shorten and Taghi M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):1–48, 2019. doi: 10.1186/s40537-019-0197-0.

[180] Seungmin Lee, Dongwan Kim, and Bohyung Han. CoSMo: Content-style modulation for image retrieval with text feedback. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 802–812, 2021.

[181] Nam Vo, Lu Jiang, Chen Sun, Kevin Murphy, Li-Jia Li, Li Fei-Fei, and James Hays. Composing text and image for image retrieval - An empirical odyssey. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6439–6448, 2019.

[182] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. LoRA: Low-rank adaptation of large language models. *ICLR*, 1 (2):3, 2022.

[183] Naureen Mahmood, Nima Ghorbani, Nikolaus F Troje, Gerard Pons-Moll, and Michael J Black. AMASS: Archive of motion capture as surface shapes. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5442–5451, 2019.

[184] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, 34(6):248:1–248:16, October 2015.

[185] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A. A. Osman, Dimitrios Tzionas, and Michael J. Black. Expressive body capture: 3D hands, face, and body from a single image. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 10975–10985, 2019.

[186] Priyanka Patel, Chun-Hao P. Huang, Joachim Tesch, David T. Hoffmann, Shashank Tripathi, and Michael J. Black. AGORA: Avatars in geography optimized for regression analysis. In *Proceedings IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2021.

[187] Germany Meshcapade GmbH, Tubingen. `https://meshcapade.com/`, 2025.

[188] Poly Haven. `https://polyhaven.com/hdris`, 2025.

[189] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. GPT-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

[190] Ragib Amin Nihal, Benjamin Yen, Katsutoshi Itoyama, and Kazuhiro Nakadai. UAV-enhanced combination to application: Comprehensive analysis and benchmarking of a human detection dataset for disaster scenarios. In *International Conference on Pattern Recognition*, pages 145–162. Springer, 2024.

[191] Xin Wang, Hong Chen, Si'ao Tang, Zihao Wu, and Wenwu Zhu. Disentangled representation learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):9677–9696, 2024.