DEGREE PROJECT IN MACHINE LEARNING,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2024

Monocular Dynamic Motion Capture: A Regression-Optimization Hybrid Approach

Master Thesis Report

Athanasios Charisoudis

KTH ROYAL INSTITUTE OF TECHNOLOGY
ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Authors

Athanasios Charisoudis <thacha@kth.se>
M.Sc. in Machine Learning
KTH Royal Institute of Technology

Place for Project

Stockholm, Sweden

Examiner

Jonas Beskow
Stockholm, Sweden
KTH Royal Institute of Technology

Supervisor

Hedvig Kjellström
Stockholm, Sweden
KTH Royal Institute of Technology

Abstract

Recovering 3D human motion from monocular video sequences poses a significant challenge in computer vision, particularly when the camera itself is in motion. The ambiguity introduced by dynamic recording setups necessitates methods to lift camera-local 3D human motions into a consistent, global world frame. This thesis proposes a novel, modular approach to monocular multi-person motion capture, combining regression techniques and global optimization for enhanced accuracy.

Our pipeline for 3D motion recovery begins with image-based detection to localize multiple human subjects within each frame. We then fit parametric human body models (SMPL) to estimate the subjects' 3D poses, resulting in camera-local human pose tracks. To recover camera motion, we implement a visual odometry (VO) algorithm. Next, we port a state-of-the-art global motion regression network to initially lift camera-local motions into a fixed world frame. Finally, we apply a global optimization process guided by re-projection quality, motion realism, and motion smoothness to refine the lifted motion estimates within the global 3D world frame.

The core contribution of this thesis is the demonstration of the effectiveness of combining global motion regression with optimization in a chained manner. Ablation studies confirm that this hybrid approach yields superior results compared to the isolated use of either regression or optimization techniques. Our experimental results show that the proposed method achieves performance closely aligned with the state-of-the-art in SMPL-based human motion recovery.

Keywords

motion recovery, monocular, dynamic, local-to-global motion lifting, optimization

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Hedvig Kjellström, for providing me with the incredible opportunity to explore this fascinating research area and conduct this thesis. Her guidance, insights, and unwavering support throughout this journey have been invaluable, while her enthusiasm and dedication have been truly inspiring. Furthermore, my gratitude also goes to the project examiner, Jonas Beskow, for taking the time to review this report and assign the discussion session and submission logistics.

My heartfelt thanks also extend to my family for their unconditional love, encouragement, and belief in me. Their support has been a constant source of strength, especially during challenging times.

This thesis marks the culmination of a remarkable journey, and I am sincerely grateful to all who have played a part in making it possible.

Acronyms

2D two-dimensional
3D three-dimensional
SOTA state-of-the-art
CG Computer Graphics
CNN Convolutional Neural Network
GCN Graph Convolutional Network
ViT Vision Transformer
CV Computer Vision
DNN Deep Neural Network
ML Machine Learning
AUC area under curve
DAG directed acyclic graph
ETL Execute-Transform-Load
HMR Human Mesh Recovery
IoU Intersection over Union
KL Kullback-Leibler
LBS Linear Blend Skinning
mAP mean Average Precision
MoCap Motion Capture
PA Procrustes Analysis
PCA Principal Component Analysis
SLAM Simultaneous Localization and Mapping
VAE Variational Autoencoder
VO Visual Odometry

Contents

1 Introduction **1**

1.1 Background . . . 1
1.2 Problem . . . 2
1.3 Purpose . . . 3
1.4 Goal . . . 3
1.5 Benefits, Ethics and Sustainability . . . 3
1.6 Methodology . . . 4
1.7 Delimitations . . . 6
1.8 Outline . . . 6

2 Extracting Information from Pixels **8**

2.1 Monocular Estimation of 2D Attributes . . . 8
2.1.1 Object Detection . . . 9
2.1.2 Instance Segmentation . . . 11
2.1.3 Object Tracking . . . 12
2.1.4 Human Pose Estimation . . . 14
2.1.5 Gender Estimation . . . 15
2.2 Modelling Human Body In 3D . . . 16
2.2.1 Direct Mesh Estimation . . . 17
2.2.2 Parametric Models . . . 18
2.3 Monocular Estimation of 3D Attributes . . . 19
2.3.1 Depth Estimation . . . 21
2.3.2 Estimation of SMPL Parameters . . . 22
2.3.3 Human Tracking in 3D . . . 23
2.3.4 Estimation of Contact Points . . . 23
2.4 Monocular Estimation of Camera Trajectory . . . 25
2.4.1 Patch-based Visual Odometry . . . 26

3 Human Motion Recovery **27**
3.1 Local Motion Recovery 28
3.1.1 Regression Based Methods 28
3.1.2 Optimization Based Methods 30
3.2 Global Motion Recovery 31
3.2.1 WHAM (Regression) 32
3.2.2 HuMoR (Motion Prior) 33
3.2.3 SLAHMR (Optimization) 35

4 Monocular Dynamic Motion Capture: A Regression-Optimization Hybrid Approach **37**
4.1 System Architecture 37
4.1.1 Frame-Level Processing (first level) 38
4.1.2 Patch-Level Processing (second level) 41
4.1.3 SMPL-Level Processing (third level) 46
4.1.4 Track-Level Processing (fourth level) 48
4.2 Experimentation Datasets 54
4.2.1 3D Poses In-The-Wild (3DPW) 54

5 Results **56**
5.1 Methodology in a Nutshell 56
5.2 Evaluation Procedure 57
5.2.1 Evaluation Metrics 58
5.2.2 Track Alignment 59
5.2.3 Bipartite Matching 60
5.2.4 Fair Comparison 60
5.3 Results on 3DPW 60

6 Conclusions **65**
6.1 Discussion 65
6.2 Future Work 65

References **67**

Chapter 1

Introduction

In this degree project the problem of recovering three-dimensional (3D) human body structure and motion dynamics from two-dimensional (2D) cues is negotiated. In particular, given a frame sequence shot by a monocular camera setup[1], we aim to invert the image formation process when it comes to humans and recover their body shape and pose in world coordinates. We focus on non-static cameras whose 3D pose (i.e. position and orientation) is considered unknown. Equivalently, the camera is considered to be extrinsically *uncalibrated*.

This first chapter is started by providing relevant background in section 1.1 and continues with a more detailed description of the problem in 1.2. The project motivation and goals follow in sections 1.3 and 1.4 respectively. The chapter is concluded with methodological clues in 1.6 as well as the outline of the rest of this report.

1.1 Background

This degree project involves theory and methodology from the intersection of Computer Vision (CV), Computer Graphics (CG), and Machine Learning (ML). In particular, ML-based CV methods are vital to recognize human subjects in a video, while Deep Neural Network (DNN) models are used to estimate the human body silhouette and pose in 3D. Finally, in order to visualize the 3D meshes and invert the

[1]Monocular capturing systems are those that comprise a single camera in the recording setup. In opposite, humans typically have binocular vision based on two eyes to perceive the three-dimensional world.

image formation process, we employ CG methods or *neural*[2] versions of them as explained later in the present document.

1.2 Problem

The problem of human Motion Capture (MoCap) from RGB cameras is a widely studied problem. Existing markerless[3] MoCap methods either need cameras to be calibrated or rely on static camera setups to infer plausible body motions [42, 49, 52]. This, in turn, leads to mesh and motion recovery relative to the camera coordinate frame. In addition, markerless MoCap methods like the ones explored in this project, rely on 2D joint detectors as the means to fit human body meshes to the depicted subjects. Such methods perform poorly when the cameras are moving and/or the motions are complex-enough for the 2D detectors to fail (e.g. in the presence of self-occlusions).

Relying on a single, uncalibrated, and non-static camera to estimate 3D human body shape and motion in a fixed (i.e. *global*) coordinate system is an intrinsically tough task. In addition to the aforementioned challenges, one can readily notice that depth ambiguity as well as entanglement of camera and human trajectories give rise to identifiability issues. Therefore, disambiguating camera from human motion, inferring depth and scale, and tackling self or other occlusions are among the key challenges for effective monocular dynamic MoCap systems.

We explore and extend recent methods that appeared in the literature and try to lift the aforementioned limitations and challenges, both in terms of camera calibration and monocular reconstruction [15, 50, 63]. In a nutshell, the research question investigated in this project is: *Given a monocular video, shot with a single, uncalibrated camera depicting humans moving in the wild, can we reconstruct camera trajectory, articulated meshes and motion in a global coordinate frame?*

[2]Neural versions of traditional CG methods in most cases refer to modifications of them so as to be differentiable. Differentiable rendering and rasterization are two prominent such examples [44].

[3]Markerless MoCap methods aim to reconstruct human body motion without using body markers as a sparse point cloud to guide 3D reconstruction. This is opposed to Marker-based MoCap methods which rely on those point clouds and the spatiotemporal correspondences for an accurate reconstruction. This is accomplished at the expense of being intrusive since the body has to be outfitted with optical markers.

1.3 Purpose

The degree project investigates methods for 3D estimation of human body shape and motion from single-view dynamic cameras. In this context, we aspire to develop and compare methods for single-view Human Mesh Recovery (HMR), MoCap, and camera trajectory reconstruction.

1.4 Goal

The aim of this project is to compare existing methods for HMR and MoCap while also propose alternations of those to account for design simplicity, resource efficiency, and effectiveness. Recent advances of generative modeling and optimization in CV are employed. In addition, a modular execution framework is also proposed to enable efficient execution of multi-staged CV tasks; we focus on dynamic monocular MoCap wherein a number of steps are involved to acquire useful 2D and 3D attributes from the input images. The deliverables comprise the following:

- implementation and comparison of state-of-the-art (SOTA) methods for monocular HMR
- implementation and comparison of SOTA method(s) for dynamic MoCap, i.e. methods that do not rely on static camera setups
- combinatory extension of modern dynamic MoCap methods by incorporating camera motion disambiguation and motion semantics
- an efficient execution framework for multi-step video processing that is based on Execute-Transform-Load (ETL)[4] design principles

1.5 Benefits, Ethics and Sustainability

Benefits The degree project benefits computer vision researchers, robotics and autonomous systems, game and urban developers, and others by advancing

[4]ETL design approach focuses on modularity by decomposing the data processing pipeline into distinct extraction, transformation, and loading functions, while leveraging libraries for data manipulation. Additionally, it incorporates node isolation and robust error handling. Our code draws from these concepts so as to be fully modular, re-usable and extensible while enabling parallel and independent processing of inputs using defined execution graphs.

monocular video analysis and enabling improved understanding and capabilities in various domains.

Ethical Concerns Privacy and informed consent are important ethical considerations when dealing with videos of humans. Respecting privacy rights and obtaining proper consent are crucial in the project. This mainly comes down to data collection and publishing agreements, alongside proper acknowledgement and redistribution statements.

Sustainability Aspects The project promotes sustainability by utilizing existing monocular video data, reducing the need for specialized equipment, and optimizing resource efficiency. The outcomes have potential applications in surveillance, robotics, and human behaviour modelling, contributing to sustainable technologies and solutions.

1.6 Methodology

The project can be split into three parts: recovery of human meshes, camera trajectory estimation, and global motion optimization. This split corresponds with recent literature trends, wherein one can found multiple disjoint studies along those areas. The input modality comprises monocular videos; therefore those estimation tasks are mostly executed per frame while some works also exploit inter-frame relations. In Figure 1.1 we supply a schematic of the path we follow towards extracting 3D human tracks from monocular inputs. This can readily be split into five parts: i) detection and tracking in 2D, ii) estimation of mesh-model parameters in camera frame, iii) estimation of camera trajectory, iv) initial regression of global human tracks and v) optimization of global tracks by leveraging learned human motion dynamics. The latter two comprise the denoted *Global Human Motion Recovery* stage.

Recovering human body mesh from 2D images has been thoroughly investigated. Sparse 2D keypoints corresponding to skeletal joints are regressed from the images to either directly fit a human 3D mesh [10, 29] or train a neural network to output the parameters of a parametric mesh model, such as SMPL [38]. As is the norm in literature, we also employ parametric models throughout this project, i.e. instead of regressing the mesh vertices directly, we estimate a set of parameters lying at a manifold of

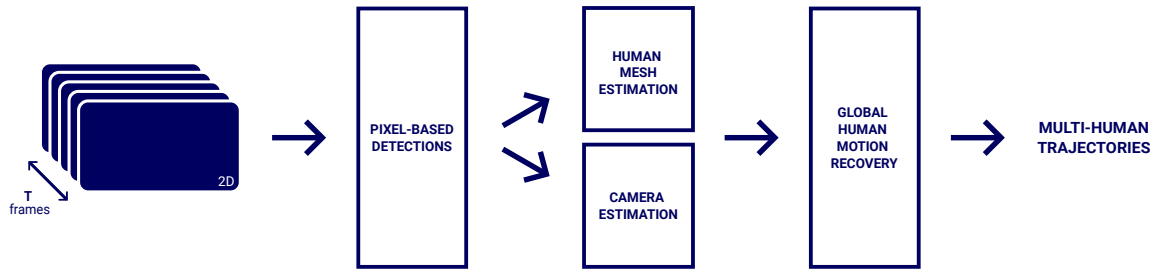


Figure 1.1: Pipeline of the methodological steps followed to extract 3D human shape and motion from monocular videos.

significantly lower dimensionality than the number of vertices. Numerous methods have been proposed to estimate parameters of such models from a single image; we explore some of those that use DNN with convolutional and attention layers.

Estimating camera poses from a monocular video is a fundamental problem in robotics and is tackled by either using Simultaneous Localization and Mapping (SLAM) or Visual Odometry (VO) odometry methods. In either case camera trajectory is explicitly inferred in a fixed world frame. Some HMR methods have also explored implicitly learning camera placement, by jointly optimizing it along meshes for plausible human motions [65]. Following modern methods for global motion recovery, however, we opted for explicitly estimating camera extrinsic sequence, denoted as *Camera Estimation* in Figure 1.1.

Motion estimation accounts for regressing global mesh pose and local (i.e. relative, based on the kinematic tree) joint transformations at each frame. The former is used to correctly place the mesh in the world coordinate frame, while the latter is necessary to produce the (parametric) human mesh and its motion. Global mesh recovery and motion estimation literature is somewhat limited as, especially for monocular cues, this is a fairly new direction. Two major works in this area are *WHAM* [50] and *SLAHMR* [63]. We extensively use WHAM as our baseline since its architecture allows for DNN-based regression of global human motions. SLAHMR has some quite inclusive and generic optimization assumptions that we also explore in this project, towards *hybrid regression-optimization* global HMR. Both such prior works employ visual SLAM to estimate camera trajectory from static scene background pixels [53, 54], which resonate our option to do likewise.

1.7 Delimitations

One of the main limitations of our work is the time and computational constraints. In particular, some of the employed methods require multi-GPU and multi-hour training or inference runs. We instead choose to downscale such techniques to enable training in academic hardware and reasonable time. Another limitation is the use of hand-crafted objectives for the optimization methods employed (based on SLAHMR). Future directions of this work could focus on offsetting the optimization scheme in neural networks that would learn to combine the error sources in a task-optimal manner.

1.8 Outline

The presented material is split in the following chapters:

- Chapter 2: Background of the tasks and methods used towards regressing human tracks in world coordinates is given. There, we explain methods for regressing useful 2D attributes, such as human detection and tracking, as well as 3D ones, including depth and mesh parameters estimation. We also present the chosen parameterization of the human body in 3D, and method for inferring camera trajectory directly from input pixels.
- Chapter 3: Methodology is laid for inferring human motion. This is split into camera-local and global motion recovery but is also clustered based on the chosen optimization approach: either DNN-based regression or optimization of hand-crafted objectives. This is a necessary foundation towards explaining our approach on hybrid regression-optimization HMR, i.e. combining regressive methods with motion-based optimization constructs.
- Chapter 4: In this fourth chapter we explain our developed system. This comprises the ETL-like execution framework to infer needed quantities, and the regression and the optimization steps of global human tracks. Our contributions on the existing methods as well as the dataset on which our experiments are performed, are also presented in this chapter.
- Chapter 5: Followingly, we present quantitative and qualitative results of our approach. This includes visualization of the intermediate extracted quantities, as well as visualizations and metrics of the regressed human meshes in 3D. The

corresponding metrics of literature methods are also given in this chapter.

- Chapter 6: This document is finalized by conclusive comments and remarks on future extensions.

Chapter 2

Extracting Information from Pixels

In this chapter, an overview of generic HMR methodology is presented, followed by a review of recent works on monocular human body shape and pose estimation, and camera trajectory regression. We focus on the SOTA works *Reconstructing World-grounded Humans with Accurate 3D Motion (WHAM)* for initially regressing global human tracks, and *Simultaneous Localization and HMR (SLAHMR)* that combines mesh recovery, camera pose estimation, global placement and disambiguation.

As those methods rely on estimating various cues from the input frames, we begin this chapter by describing monocular visual learners, i.e. deep models that estimate pixel-wise 2D attributes, necessary for the subsequent stages. We then proceed by describing *SMPL*, the chosen representation for humans in 3D, as well as monocular estimation and tracking of SMPL-based meshes. The chapter is finalized by reporting modern ways to regress global human motion in 3D from monocular inputs.

2.1 Monocular Estimation of 2D Attributes

With the term *monocular* we refer to the inputs being 2D cues shot by a single camera. Therefore, no other information, such as depth or stereo, is assumed present. We start this section by describing the chosen methods for human subject detection and pixel-wise segmentation, appearance-based tracking, and 2D pose estimation, on the input frames. Those are necessary intermediate steps towards isolating humans and followingly estimate relevant 3D attributes.

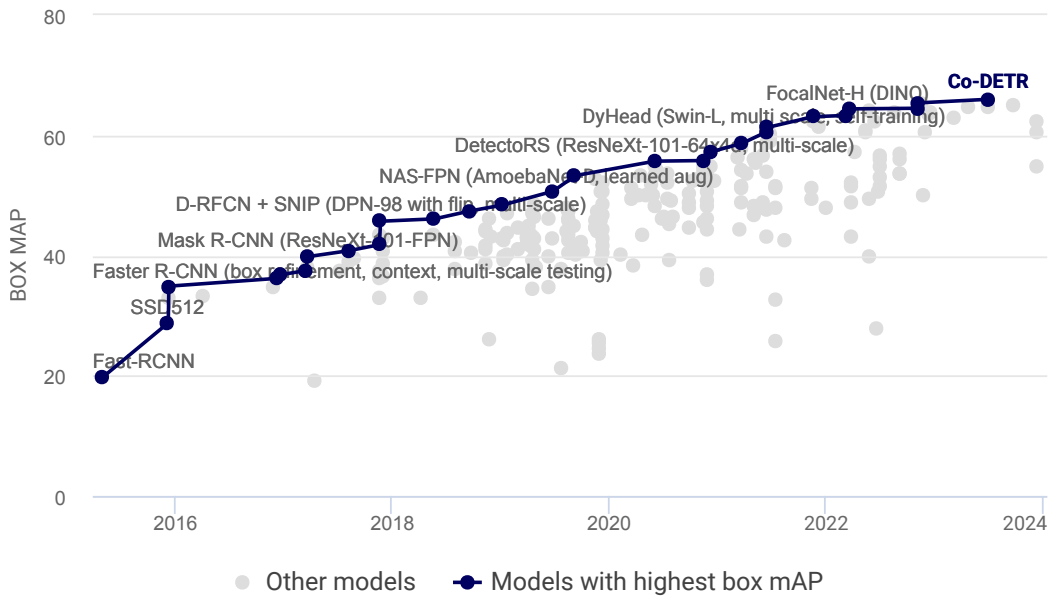


Figure 2.1: SOTA object detection methods on COCO test set. We opted for the current best performing model, *Co-DETR*, in terms of mean Average Precision (mAP).
Source: Papers with code (image taken February 2024).

2.1.1 Object Detection

Object detection is a branch of CV that deals with identifying and locating objects in images by means of class and object bounding box prediction respectively. This is the first operation on the input pixels, with its accuracy being crucial for the performance of the overall performance of the developed system. One can readily see that missing detection of necessary objects at this stage, there is small chance that those be recovered later on.

Given its importance, we opted for the SOTA object detection model, *Co-DETR* [67], as evaluated using the mAP metric in the wide and diverse dataset of real-world objects in context, COCO [35]. In Figure 2.1, we list object detections methods appeared in the literature and highlight the chosen one. To calculate mAP, the overlap between each predicted bounding box and the corresponding ground truth bounding box is determined combined with the area under curve (AUC) for the object class predictions and corresponding ground-truth classes. mAP takes values from 0.0 to 1.0 with higher values corresponding to more accurate bounding box and class detections.

DETR, proposed by Carion et al. in the work *End-to-end object detection with Transformers* [7], is a very successful model in object detection. As can be seen in Figure 2.2, DETR starts by extracting image features using a Convolutional

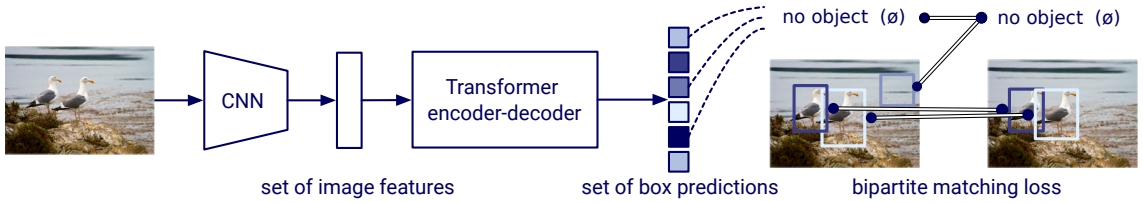


Figure 2.2: Direct predictions of bounding boxes and class probabilities by DETR, using global attention layer to input image features.
Source: End-to-end object detection with Transformers, Carion et al. [7]

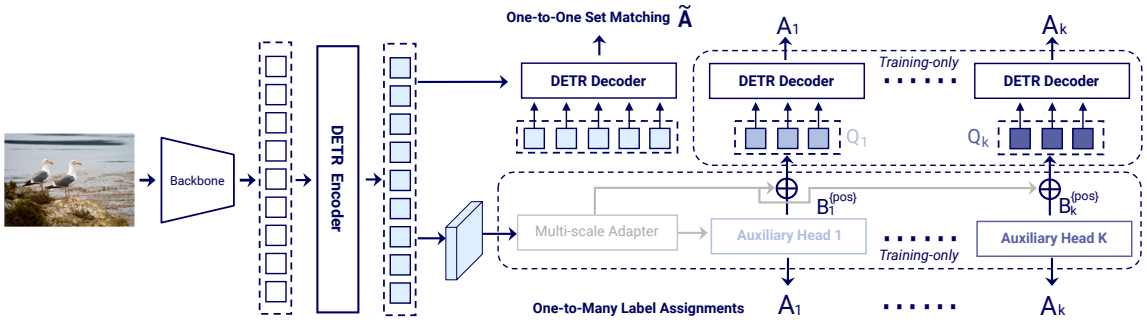


Figure 2.3: Complementary and dense error signals from additional decoding heads in Co-DETR, help boost learned embeddings by the DETR encoder during training.
Source: DETRs with Collaborative Hybrid Assignments Training, Zong et al. [67]

Neural Network (CNN). The transformer [56] encoder then processes these features, understanding the relationships between different parts of the image. It directly predicts a fixed set of objects and their bounding boxes, unlike traditional models that rely on complex hand-designed components. Finally, it performs one-to-one (bipartite) matching between predictions and ground truths and update the convolutional and attention weights based on this comparison.

DETR, suffers from sparse supervision in both its encoder and decoder. One-to-one matching of predictions to ground truth limits the learning signals, hindering feature representation learning in the encoder and attention learning in the decoder. Depicted in Figure 2.3, *Co-DETR* [67] extends DETR during training, by introducing auxiliary detection heads that run parallel to the main DETR attention heads, that leverage ”one-to-many” label assignment strategies (from Faster R-CNN [14]) to provide complementary dense error signals and thus boost the encoder’s ability to extract discriminative features.

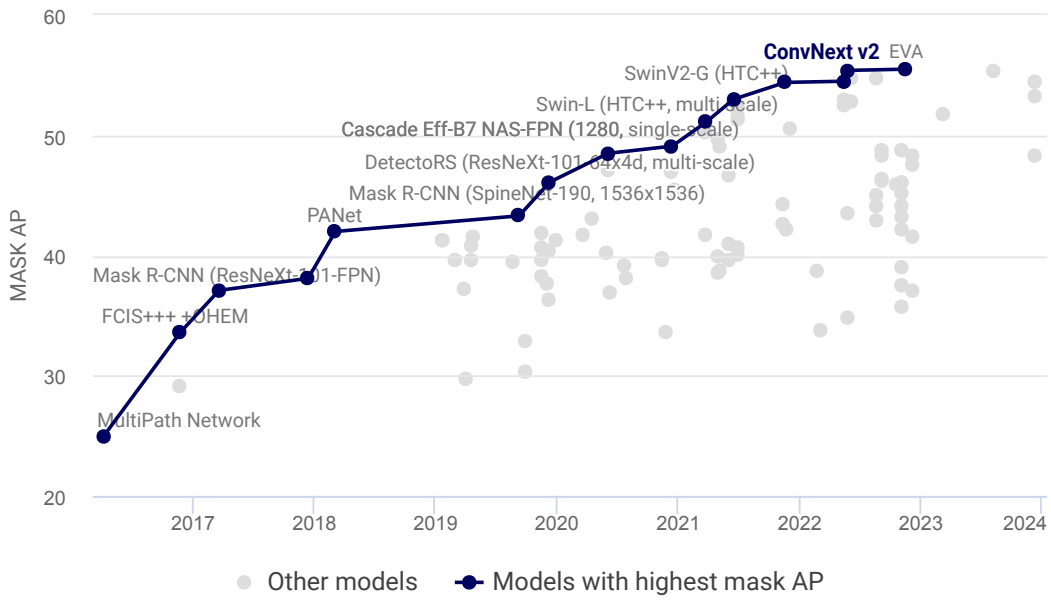


Figure 2.4: SOTA instance segmentation methods on COCO test set. We opted for the one of the current best performing models, *ConvNext v2*, in terms of mask-mAP. **Source:** Papers with code (image taken February 2024).

2.1.2 Instance Segmentation

While object detection focuses on regressing bounding boxes around identified objects, instance segmentation goes a step further and estimates a dense pixel-based mask isolating the identified objects in the input image. Instance segmentation is essential as it not only allows us to distinguish between the foreground and the background pixels corresponding to an object, but also help us identify different objects within the same original bounding box, even if those are of the same class. The way instance segmentation methods work is by determining the the class at pixel-level, i.e. for every pixel it estimates the probability of it belonging to any of the pre-defined object classes. This way, objects in the image can have rich outlining alongside the detected bounding box and class, enabling a wider spectrum of downstream tasks.

We segment instances of humans in the input frames, a step necessary towards isolating the apparent texture of every human subject at every timestep in the input video. Those textures are then aggregated and used towards improving and stabilizing human tracking during the visible frames, as will be explained in detail later on this document. Given the sensitivity of the aforementioned task to pixel-wise human masks, we chose to use one of the best-performing instance segmentation model in

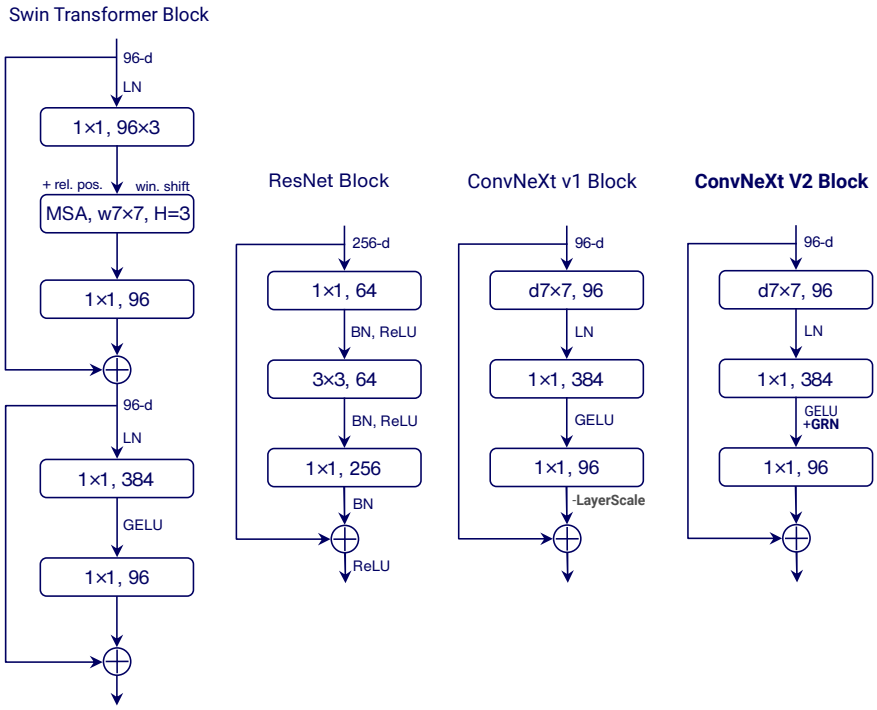


Figure 2.5: Comparison of the blocks of modern CNN models. As can be seen, ConvNext v2 highly resembles the successful ResNet [18] blocks, extending them with larger convolutional kernels and more efficient normalization layers inspired by the SwinTransformer [36].
Source: ConvNext v1 paper [37], and ConvNext v2 paper [60]

the COCO test set, *ConvNext v2*, as can be seen in Figure 2.4.

ConvNext v2, presented by Woo et al. [60] marks the SOTA method in instance segmentation among the ones using exclusively CNN as building blocks. Being inspired by Transformer in its design as well as leveraging depth-wise convolutions, novel normalization techniques, and the masked autoencoder (MAE) pre-training framework, it is able to learn powerful visual representations used in the downstream segmentation task. A schematic of the architecture is provided in Figure 2.5.

2.1.3 Object Tracking

Object detection and instance segmentation both occur at image-level, i.e. each frame in the video is processed independently from the neighboring ones in the input video. As a result, there is no notion of instance tracking and therefore every detection is assigned a different *tracking id* at every timestep or frame index. However, we can straightforwardly apply tracking-by-detection, a dominant paradigm in computer vision for tracking multiple objects in videos, which uses an association algorithm that

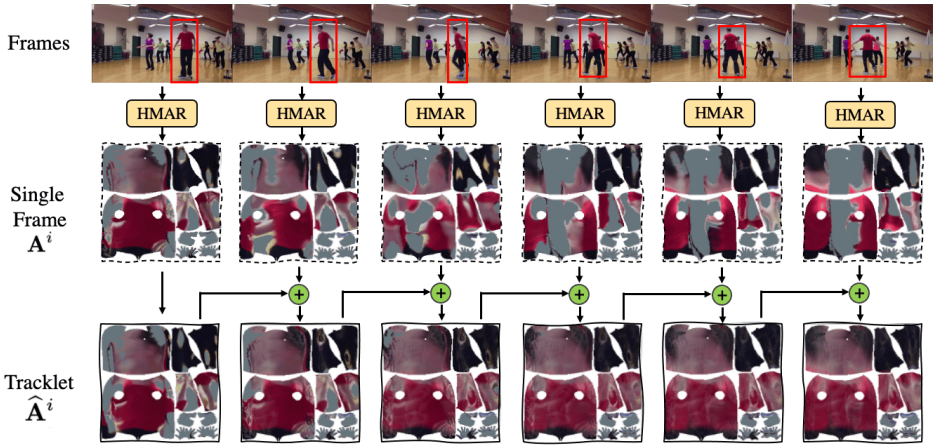


Figure 2.6: Extending DeepSORT by using regressed UV texture maps instead of CNN-based appearance embeddings. The maps are enriched through time via aggregation. **Source:** Tracking people by predicting 3D appearance, location and pose by Rajasegaran et al. [43]

takes new detections and links them with existing object tracks from previous frames. Different linking strategies exist; we focus on the widely popular and flexible method *DeepSORT* which in turn builds upon the *SORT* tracking framework.

SORT (Simple Online and Realtime Tracking) [4] presented by Bewley et al., uses a Kalman filter to predict the future positions of detected objects based on their previous movement patterns. It then associates new detections with existing tracks primarily based on Intersection over Union (IoU), a measure of overlap between predicted and detected bounding boxes. SORT focuses on simplicity and efficiency, which makes it operate fast, but with the downside of suffering in the presence of occlusions (i.e. it fails to correctly re-identify objects).

DeepSORT [59] presented by Wojke et al., integrates appearance information to improve the performance of SORT and enable tracking objects through longer periods of occlusions, effectively reducing the number of identity switches. To do this, DeepSORT obtains a vector for every image patch, by passing it through a pretrained image classifier and getting the output of the last convolutional layer. These ”appearance embeddings” are used in the association metric to help re-identify lost objects due to occlusions or out-of-frame motions. Rajasegaran et al. [43] extend DeepSORT by incorporating richer embeddings; when it comes to appearance those comprise aggregated UV texture maps as depicted in Figure 2.6.

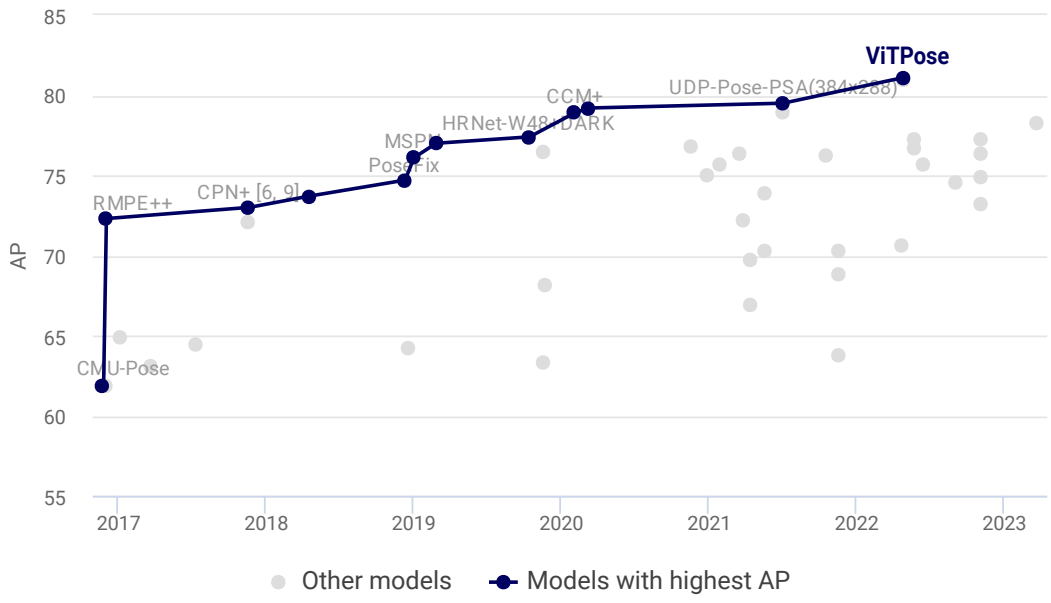


Figure 2.7: SOTA human pose estimation methods on COCO Keypoint test set. We opted for the current best performing model, *ViTPose*, in terms of joint location mAP. **Source:** Papers with code (image taken February 2024).

2.1.4 Human Pose Estimation

Another very useful modality that can be directly estimated from input images is the 2D position of human joints, as those can be used to constrain the optimization of 3D human body shape and camera pose. Given datasets with manual such annotations, many DNN models have been proposed to regress those positions from images. Again, we opted for the current best performing model, *VitPose*, as can be seen for the SOTA human pose estimation chart in Figure 2.7.

ViTPose presented by Xu et al. [62] is a family of models that establishes Vision Transformer (ViT) as remarkably effective backbones for human pose estimation. Instead of traditional CNN, ViTPose leverages plain, non-hierarchical ViTs to extract image features, coupled with a lightweight decoder for pose estimation, a schematic of which is given in Figure 2.8. Its scalable design offers a new performance-throughput trade-off within pose estimation, achieving excellent results on the COCO Keypoint dataset. We use VitPose to extract 2D joint locations of detected humans, and compare them with projected 3D joint locations coming from the estimated 3D human meshes.

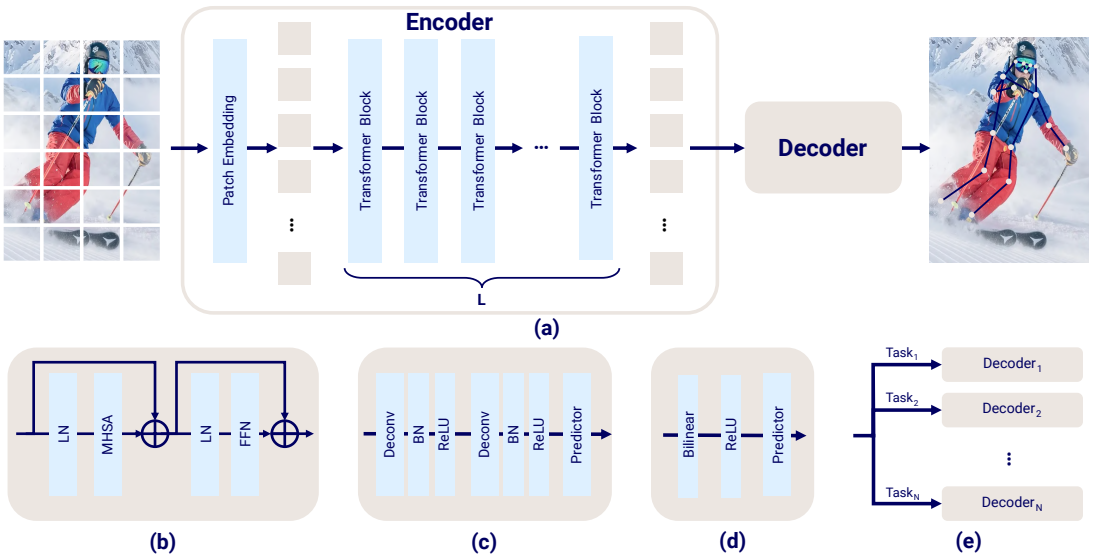


Figure 2.8: (a) The encoder-decoder architecture of ViTPose. (b) The transformer block. (c) The classic decoder. (d) The simple decoder. (e) Decoders for multiple datasets.
Source: ViTPose: Simple Vision Transformer Baselines for Human Pose Estimation, Xu et al. [62]

2.1.5 Gender Estimation

Gender plays a vital role in human body shape and pose-dependent shape deformation, as different genders tend to have larger offsets in the chests and hips areas and shifted average heights (at least for the scanned bodies used to train mesh regressors that are employed in this project). While some datasets provide ground truth genders for the depicted subjects, we opted to estimate gender independently for every detected human, in an effort to make the entire pipeline more flexible and realistic. Alternatively, one could have chosen to develop gender-agnostic HMR systems, that are usually implemented using an average, *neutral*, human mesh template, that hurts the fidelity of the generated meshes and motion.

We use a modern model for gender and age estimation from images, *MiVOLO*, presented by Kuprashevich and Tolstykh in [31]. As shown in Figure 2.9, MiVOLO leverages ViTs as backbones to extract useful representations from images containing the isolated body and the cropped face (of the same person), and accurately estimate age and gender. Being trained with challenging ”in the wild” datasets, it exhibits good generalizability; we found that it is biased towards male gender predictions when evaluated on the human motion estimation dataset we used in this project.

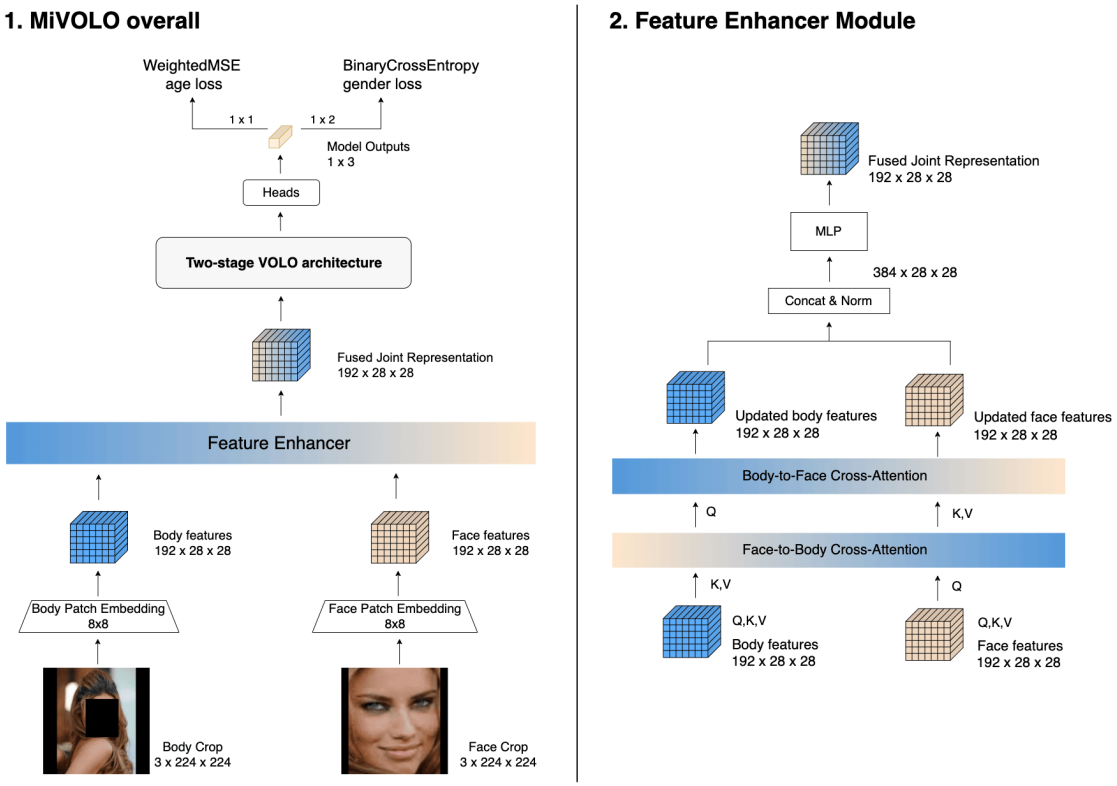


Figure 2.9: MiVOLO overall (left) and feature enhancer (right) architectures. The image features are early-fused in the *Feature Enhancer* using cross-attention, before being processed from another ViT-based backbone, *VOLO* [64]. The final prediction heads output the age and gender probabilities respectively.
Source: Multi-input Transformer for Age and Gender Estimation, Kuprashevich and Tolstykh [31]

2.2 Modelling Human Body In 3D

Three-dimensional human body shape models offer compelling advantages, allowing us to infer a subject's shape from incomplete or ambiguous 2D/3D data. The ultimate goal is to simulate humans comprehensively, including bones, joints, muscles, tissues, and skin [19]. However, practical limitations often restrict us to scanning the outer body surface with 3D scanners. Those scans have been used to develop statistical models that are easy to use and accurately represent the diversity of human forms. To represent the skin surface, a water-tight mesh[1] is employed.

Two broad categories of representations exist for creating 3D humans. The first directly manipulates the mesh itself, while the second, relies on a lower-dimensional set of parameters to control the final mesh. Estimation methods follow a similar

[1]A mesh is a fundamental data structure in computer graphics for representing the shape and surface of 3D objects. It consists of vertices, edges, and faces.

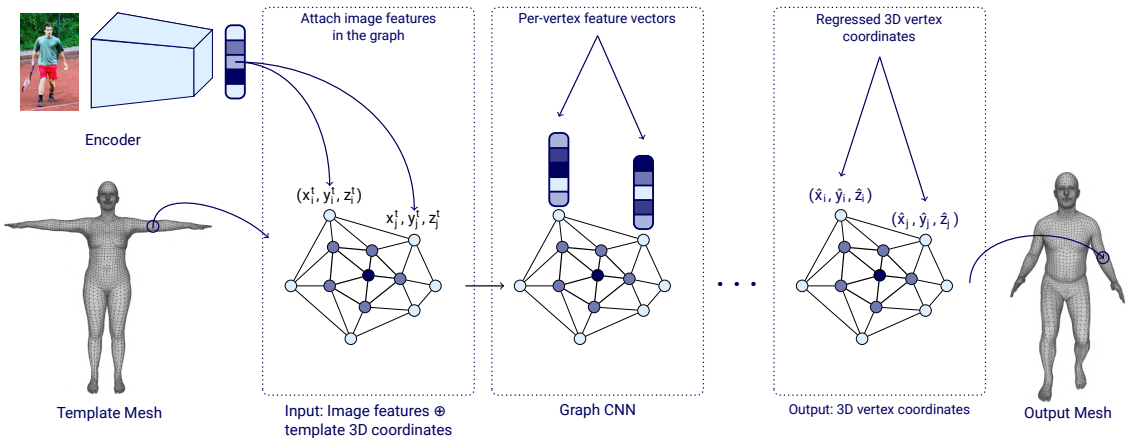


Figure 2.10: Overview of GCN for human mesh estimation. First, a CNN extracts image features, integrating them with a template 3D human mesh. Then GCN is employed to directly deform the mesh, matching it to the person in the input image.
Source: Convolutional Mesh Regression for Single-Image Human Shape Reconstruction, Kolotouros et al. [30]

division: some directly generate mesh vertices, while others produce a set of parameters that are then used to construct the mesh. Following standard practice in the literature, we focus on the latter parameter-based approach for 3D human modeling. However, for completeness, this section starts with a brief description of a prominent method from the direct mesh manipulation category.

2.2.1 Direct Mesh Estimation

One of the prominent methods to directly edit the human mesh to match the shape of humans depicted on the input images is the work by Kolotouros et al. named *Convolutional Mesh Regression for Single-Image Human Shape Reconstruction* [30]. The proposed architecture directly regresses the 3D coordinate offset of vertices on a template mesh. They achieve this using a Graph Convolutional Network (GCN) architecture. The GCN leverages the inherent structure of the mesh for more effective processing of image features, which are fused with the mesh vertices.

This framework allows for flexible 3D shape reconstruction without a rigid reliance on a predefined parametric body model. The latter however may also be seen as a drawback of this cluster of methods, as it allows non-plausible vertex deformation, as can be seen in Figure 2.11 where the output of the aforementioned model is compared with the output of the *SMPL* parametric-mesh model described in the next sub-section.

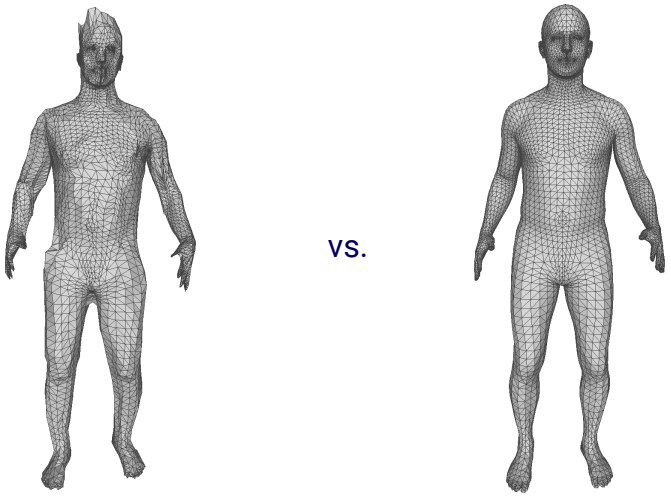


Figure 2.11: Poor fitting of the human mesh by regressing on the vertex-level (left). Instead, parametric human modals only allow plausible mesh fits (right).
Source: Convolutional Mesh Regression for Single-Image Human Shape Reconstruction, Kolotouros et al. [30]

2.2.2 Parametric Models

Instead of directly regressing mesh vertices, a number of techniques have been proposed that operate on a low-dimensional manifold of mesh parameters [38, 61]. Those are used by developed mesh regressors to produce the final vertex set, and usually a set of 3D joint locations found internally. In this project we make use of the widely popular parametric mesh model, *SMPL*, which is described next.

The SMPL Model Family

Loper et al. presented in their paper *SMPL: A Skinned Multi-Person Linear Model* [38] a learned model of human body shape and pose-dependent shape variation from 3D scans, which parameterizes the skin mesh using a set of body shape parameters and joint angles (of an underlying skeleton). The parameters are typically named $\vec{\beta}$ and $\vec{\theta}$ respectively and are used to linearly blend a template mesh in rest (T-) pose, $\overline{T}$, with a set of learned shape-dependent and pose-dependent vertex deformations (or *blend shapes*). The "blended" mesh is then posed based on the $\vec{\theta}$ parameters using a set of learned skin-to-joints influence weights, $\mathcal{W}$ and Linear Blend Skinning (LBS)[2]. As can be seen in Figure 2.12, $\mathcal{W}$ learns an association between each vertex and different bones,

[2]LBS is a fundamental skinning technique where each vertex on a 3D mesh is transformed by a weighted combination of nearby bone transformations. The SMPL model employs LBS to articulate its mesh based on joint rotations (pose parameters) and corrective blend shapes (shape parameters).

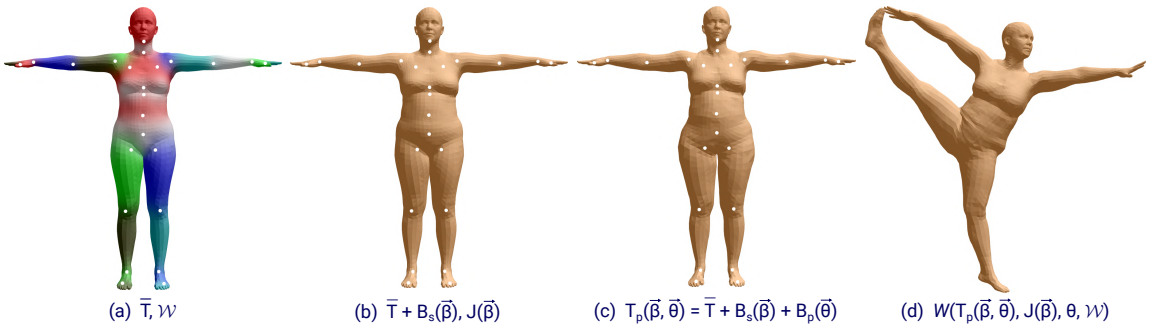


Figure 2.12: The SMPL model. (a) Template mesh with blend weights indicated by color and joints shown in white. (b) With identity-driven blend shape contribution only; vertex and joint locations are linear in shape vector $\vec{\beta}$. (c) With the addition of of pose blend shapes in preparation for the split pose; note the expansion of the hips. (d) Deformed vertices reposed by blend skinning for the split pose.
Source: SMPL: A Skinned Multi-Person Linear Model, Loper et al. [38]

enabling smooth deformations as the model is posed or its shape is altered.

Described by the low-dimensional parameter set of $\vec{\beta}$ (usually 10 Principal Component Analysis (PCA) coefficients) and $\vec{\theta}$ (23×3 joint angles in axis-angle representation, describing the orientation of each of the 23 joints in SMPL's kinematic tree), the model can produce a sufficiently wide spectrum of human body pose and shape variations, as exemplified in Figure 2.13. In addition, as the final mesh is created using blend skinning, SMPL is compatible with existing graphics pipelines and can be readily used in all modern 3D modeling software, such as Blender [9] and Autodesk Maya [2], linking ML-based mesh estimation methods to existing animation tools. We employ SMPL as the sole model for regressing human meshes in this project; for every detected human in the each video frame we thus have to estimate its corresponding SMPL parameters, a process analyzed in 2.3.2.

2.3 Monocular Estimation of 3D Attributes

Multiple methods have appeared in literature that aim to regress 3D quantities directly from 2D cues. In particular, we focus on ones estimating SMPL parameters, SMPL mesh textures, likelihood of vertex contact as well as (relative) depth directly from pixels. We analyze the methods used in this project in the rest of this section starting from monocular depth estimation.

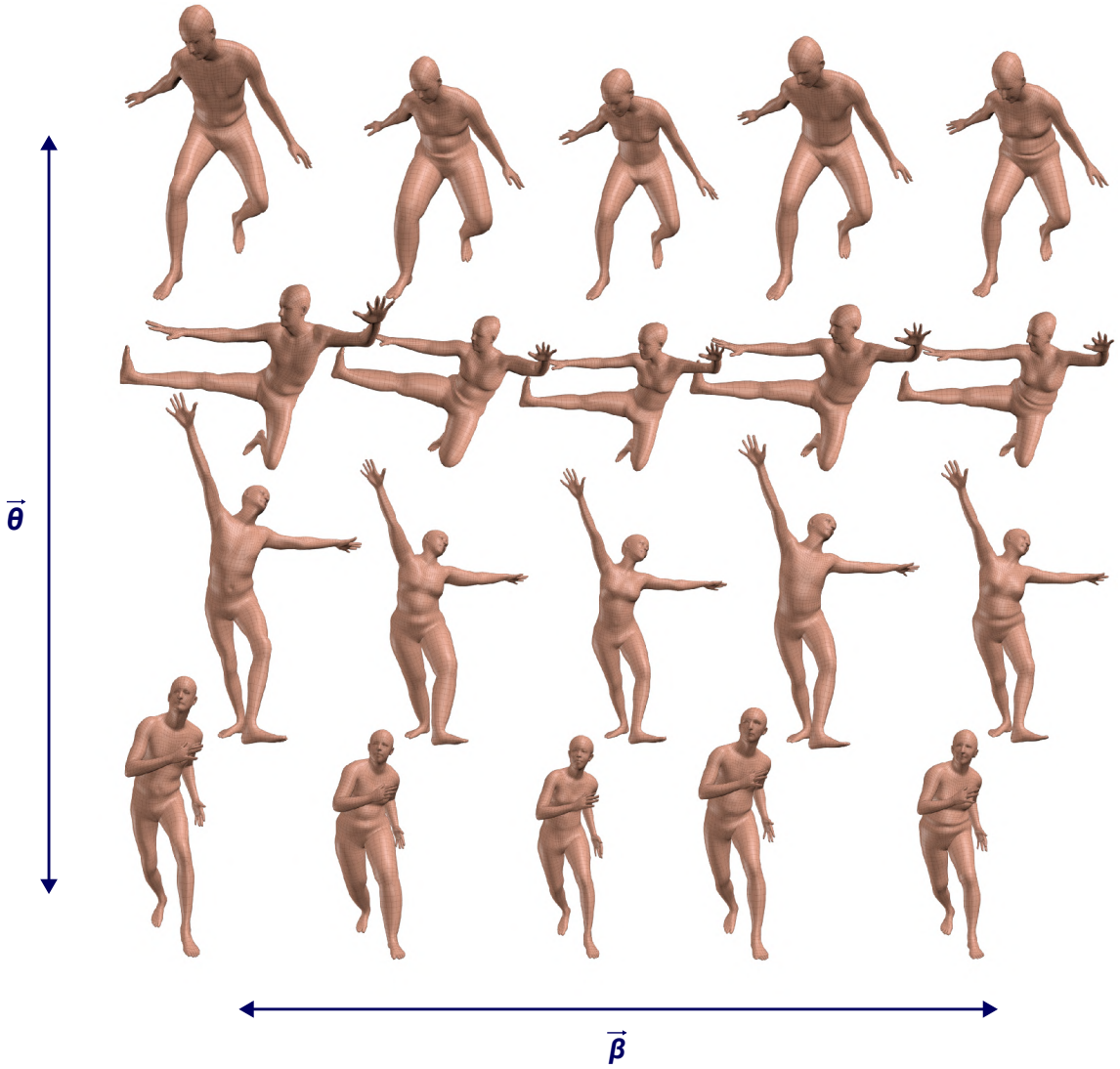


Figure 2.13: Decomposition of SMPL parameters into pose and shape: Shape parameters, $\vec{\beta}$, vary across different subjects from left to right, while pose parameters, $\vec{\theta}$, vary from top to bottom for each subject.
Source: SMPL: A Skinned Multi-Person Linear Model, Loper et al. [38]

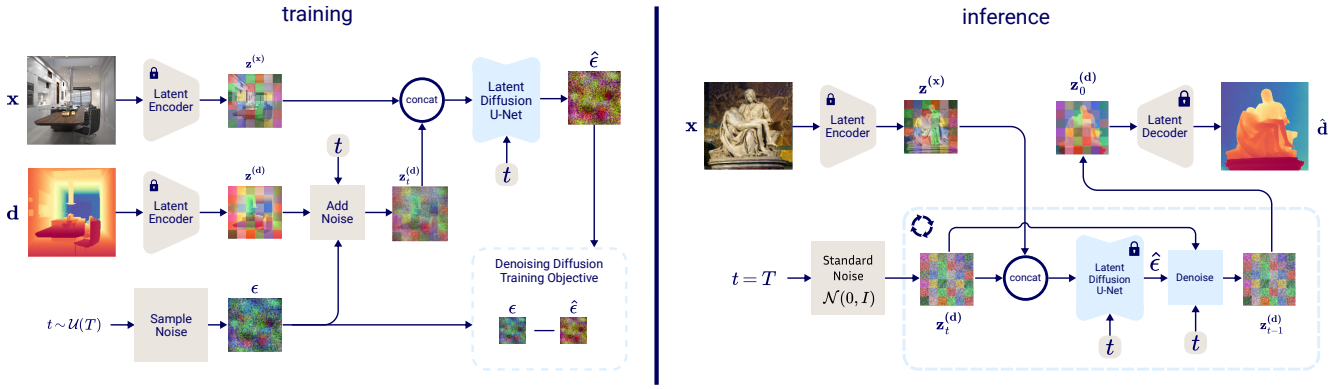


Figure 2.14: Training (left) and inference (right) pipelines of the Marigold monocular depth estimation method which is based on Stable Diffusion. The denoising U-Net learns to undo the depth noising step, starting from a noised version of the ground-truth depth (training) or gaussian noise (inference).
Source: Repurposing Diffusion-Based Image Generators for Monocular Depth Estimation, Ke et al. [25]

2.3.1 Depth Estimation

Monocular depth estimation is a fundamental computer vision problem concerned with the reconstruction of 3D scene depth from a single 2D image. Due to the inherent ambiguity of inferring depth from a monocular view, this task necessitates the combined use of visual cues (e.g., relative size, defocus, and parallax) with sophisticated machine learning algorithms that provide depth priors. We employ depth estimation in this project in order to enhance the performance of camera pose estimation, as depth information is needed to find correspondences among image patches between two consecutive frames. For this reason, it suffices for our task to just know the ordering of the pixels, i.e. the depth "bin" that each pixel falls into, which is commonly referred to as *relative depth estimation* (in contrast with absolute depth estimation which is the metric depth distance of every pixel back-projected to the world).

The chosen method to estimate depth from input images is *Marigold*, presented by Ke et al. in their work *Repurposing Diffusion-Based Image Generators for Monocular Depth Estimation* [25]. In contrast to traditional methods that directly regress absolute depth values, Marigold adopts an approach focused on estimating relative depth relationships between image pixels. Marigold exploits the rich visual knowledge embedded within large generative models like Stable Diffusion [47] instead of directly predicting depth maps. In particular, the authors propose a latent diffusion model[3]

[3]Latent Diffusion Models employ an autoencoder so as to enable training and inference in the latent

that is trained on purely synthetic image-depth latent pairs. Fine-tuning Marigold on synthetic data further refines this capability, resulting in a computationally efficient model that produces accurate depth estimates. This unique adaptation of diffusion-based image generators demonstrates the potential to unlock their latent scene understanding for tasks beyond image creation.

2.3.2 Estimation of SMPL Parameters

Estimating the parameters that generate the SMPL mesh from a single image is a difficult, though very rewarding task. As such, a number of techniques with relying on minimization of mesh re-projection fidelity criteria. In most cases the criterion is joint re-projection error, i.e. how far are the projected SMPL joints from the provided (or detected as described in 2.1.4) ones. Additionally, some methods have proposed to maximize the realness of the 3D pose in order to discourage unlikely complex mesh placements. In any case, it is important that the reader realizes that this estimation results in *camera-local* placement of the meshes in 3D. As it will be explained later, this is fine if the camera is static (fixed in the world) but poses quite some challenges in dynamic recording setups.

Having robust estimates of the parameters of the human mesh for every detection is crucial, and therefore we chose to use the SOTA model (pretrained) for monocular estimation of SMPL parameters, *HMR 2.0* presented by Goel et al. [15]. As shown in Figure 2.15, HMR 2.0 utilizes ViT backbones to extract useful embeddings from cropped image patches, that are then used in a ViT decoder which outputs the regressed SMPL parameters as well as the camera translation in the world. The latter is not robust and mostly wrong when the camera is not static, based on which we neglect this part of the model's output (we assume dynamic recording setups). We now have a way to estimate camera-local SMPL meshes for every detected human bounding box and at every frame. Followingly, we describe how the extracted mesh information is combined to make the tracking of human subjects more robust.

space of it. This way they are much more computationally efficient while also showcasing improved performance.

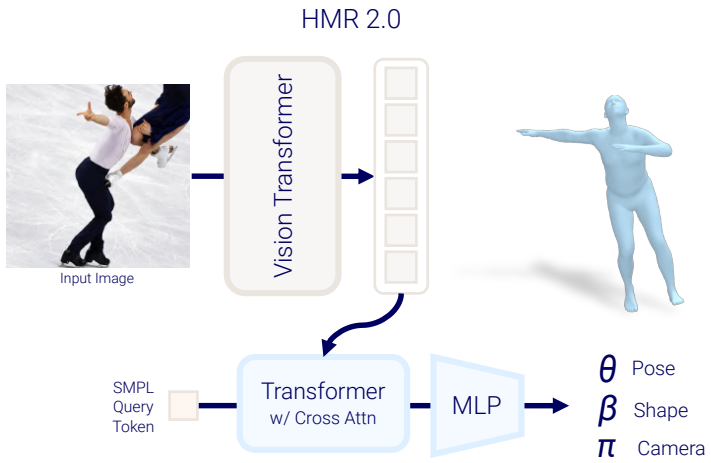


Figure 2.15: HMR 2.0, a fully "transformerized" version of a network for Human Mesh Recovery.
Source: Humans in 4D: Reconstructing and Tracking Humans with Transformers, Goel et al. [15]

2.3.3 Human Tracking in 3D

In 2.1.3 the object tracking problem was presented along with DeepSORT [59], a flexible architecture for efficient object tracking and re-identification. It was then mentioned, that by construction DeepSORT can incorporate additional cues in the matching process (of new detections to the existing tracks). *PHALP* presented by Rajasegaran et al. [43], is a DeepSORT-based tracker that incorporates the following additional attributes to make the tracking more robust:

- **pose**: flattened vector of the 23 regressed joint angles from SMPL
- **location**: camera and pelvis translations from the world origin (or relative pelvis translation in the case of moving cameras)
- **appearance**: flattened UV-texture maps regressed by projecting SMPL meshes to the input image and picking colors at pre-defined vertex indices.

PHALP is the default tracker used by Goel et al. [15] after employing HMR 2.0 to estimate SMPL parameters; we follow the same approach in this project as well to derive camera-local human tracks.

2.3.4 Estimation of Contact Points

Humans interact with the world via touching, walking, sitting etc., all of which have contacts between outer skin and scene surfaces in common. As such, having contact

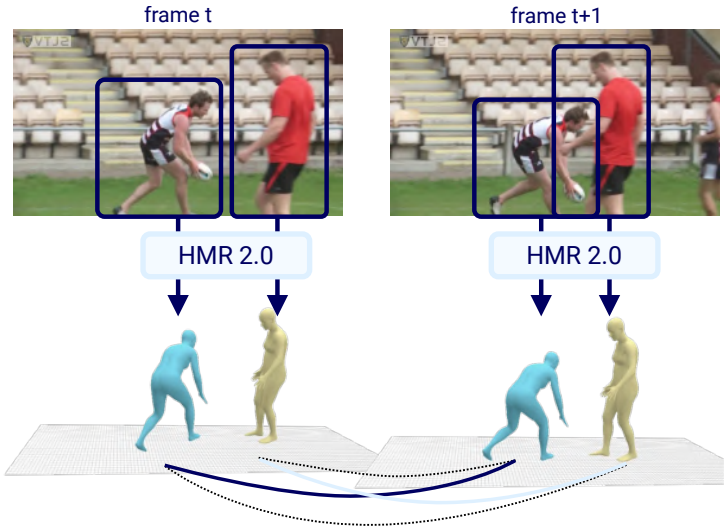


Figure 2.16: PHALP extends DeepSORT by incorporating SMPL pose, location, and appearance information to boost tracking performance.
Source: Tracking People by Predicting 3D Appearance, Location and Pose, Rajasegaran et al. [43]

heatmaps on top of SMPL vertices directly from 2D cues can be really helpful as it enables more physically-plausible placement of the human body meshes in 3D. In addition, contact information helps preventing the well-known floor penetration problem prevalent in monocular SMPL estimation techniques [55].

Huang et al. presented in their work entitled *Capturing and Inferring Dense Full-Body Human-Scene Contact* [21] a model for estimating per/vertex contact probabilities from images, *BSTRO*, along with a dataset with SMPL meshes and contact probabilities aligned to visual cues, *RICH*. As can be seen in Figure 2.17, BSTRO is able to provide accurate contact heatmaps even for occluded vertices or complex poses of the fitted mesh. This is accomplished by training a ViT backbone on the SMPL template mesh along with regressed SMPL parameters, to correctly classify each vertex as being in contact. In this project we use BSTRO (pretrained) to get possible points of contact between the regressed bodies and the scene surfaces. We focus mainly on the feet, as knowing whether they touch the ground is a necessary cue to encourage realistic placement and motion of the body in the world frame. Feet contact probabilities are also employed in the *WHAM* method for global human track regression (presented in the next chapter).

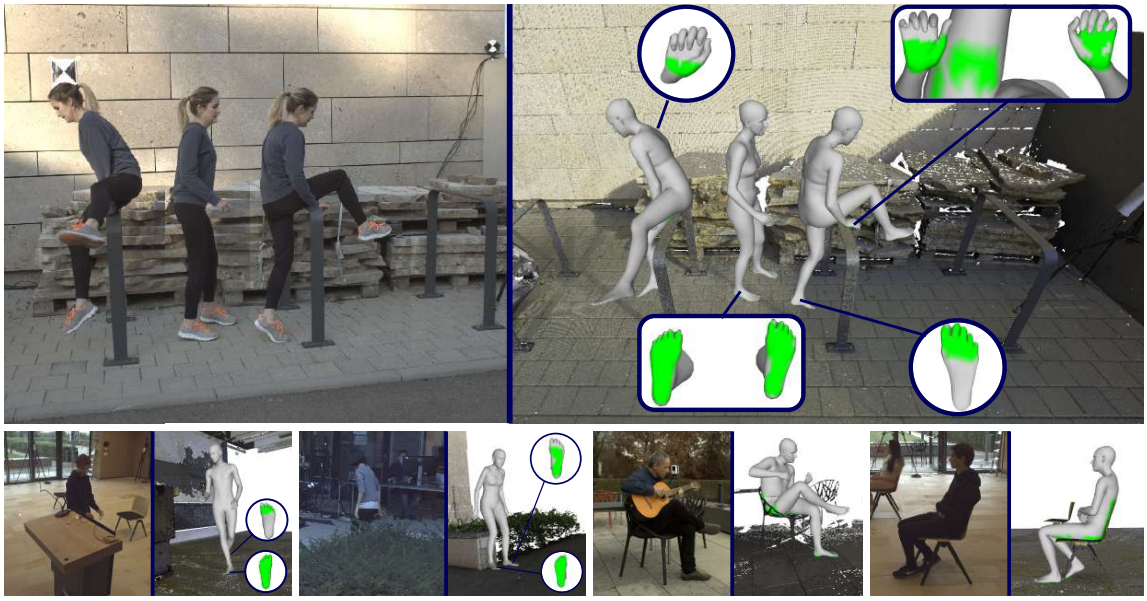


Figure 2.17: Input images and predicted SMPL vertex contacts from the BSTRO model. BSTRO can successfully capture contacts even for unseen body parts. The images are from the RICH dataset presented along with the aforementioned model.
Source: Capturing and Inferring Dense Full-Body Human-Scene Contact, Huang et al. [21]

2.4 Monocular Estimation of Camera Trajectory

One of the main assumptions or challenges of the work presented in this document is that the cameras need not be fixed nor extrinsically calibrated. Therefore, in order to globally place the regressed human meshes, we need to estimate the pose of the camera as well. A number of methods have been proposed to deal with this inherent ambiguity, either implicitly [65], or explicitly [50, 63] by incorporating a camera trajectory estimation stage. This section focuses on the latter, i.e. the problem of regressing a sequence of camera poses in a fixes world frame is negotiated.

Visual-SLAM and VO address this fundamental problem in robotics of determining an agent's as well as the camera pose within an environment from visual cues; we focus exclusively on the camera hereby. While VO incrementally estimates the camera's trajectory by analyzing changes between consecutive image frames, SLAM tackles the more expansive task of building a persistent map of the environment in concurrence with self-localization. In essence, SLAM augments VO's trajectory estimation with the ability to recognize previously visited locations and refine the understanding of the spatial environment making it significantly more computationally expensive. As such and based on findings of previous works [63], we employ VO methods in this

project to maintain its ability to run in resource-constrained environments, focusing on its modern take, patch-based VO described next.

2.4.1 Patch-based Visual Odometry

Patch-based VO departs from traditional feature-based VO methods. Rather than exclusively relying on distinct keypoints like corners, patch-based methods extract small image regions (patches) and track their movement across frames. These patches often contain richer texture information than isolated keypoints, potentially improving robustness in scenarios with repetitive textures or low feature density. Patch-based approaches often leverage advanced techniques like deep learning-based patch descriptors to compute similarity between patches in different frames, aiding in the estimation of camera motion. Hereby we employ the SOTA method *DPVO*.

DPVO, presented by Teed et al. in their work *Deep Patch Visual Odometry* [54], revolutionizes patch-based visual odometry by introducing deep learning and a recurrent architecture. Its core involves a residual network for extracting descriptive patches from images, coupled with a recurrent network that dynamically tracks these patches over time. The camera pose is then estimated using differentiable version of a bundle adjustment [6] layer, where the objective is to minimize the re-projection error of the tracked patches in a recurrent fashion as depicted if Figure 2.18. This unique combination leads to a computationally efficient and highly accurate model that outperforms prior patch-based and dense visual odometry techniques.

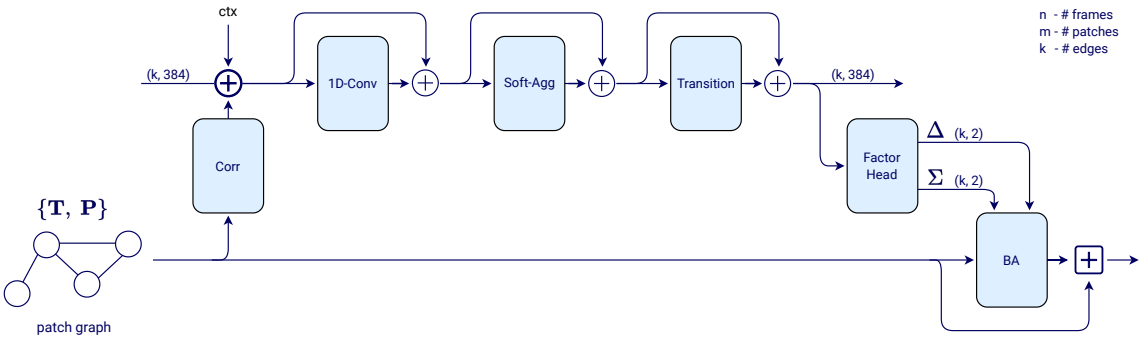


Figure 2.18: Recurrent update operator of DPVO. Correlation features are extracted from temporally neighboring patches (based on a patch-to-frame graph) and injected into the hidden state. Then, the hidden state is processed by convolutions and a transition block. The factor head produces camera trajectory revisions which are used by the bundle adjustment layer to update the camera poses and the depth of patches. Each ”+” operation is a residual connection followed by layer normalization [3].
Source: Deep Patch Visual Odometry, Teed et al. [54]

Chapter 3

Human Motion Recovery

The first step in the process of global human motion recovery from videos is detecting and identifying the human subjects depicted throughout the given frames. Subsequently, the parameters of SMPL models are regressed resulting in realistic 3D human bodies that, at best, are consistent with visual cues up to a projective transformation [17]. When the camera parameters (mostly the extrinsic ones) are unknown an additional processing step is taken in order to resolve the inherent ambiguity between body and camera motion. Methods for tackling all these steps directly from the input images were presented in chapter 2.

In this chapter we focus on another aspect of human motion recovery, that of regression- vs. optimization-based processing of both camera-local and global tracks, and describe each through related literature works. This is necessary in order to lay the foundation and introduce the reader to our approach, which is a **hybrid regression-optimization** one. For clarity matters, we repeat here the aforementioned ways of clustering HMR methods based on operational domain, method mechanics, and literature trends:

- **Local vs. Global** Camera-local methods assume that the camera is placed at the world origin and therefore all the inferred meshes are relative to the camera pose. In contrast, global methods focus on the discovery of both the camera and the mesh pose relative to a fixed world frame of reference (see Figure 3.1).
- **Regression vs. Optimization** Regression-based methods usually employ DNN architectures that are trained in maximum likelihood manner to directly estimate the mesh parameters. This is in contrary to optimization-based methods where

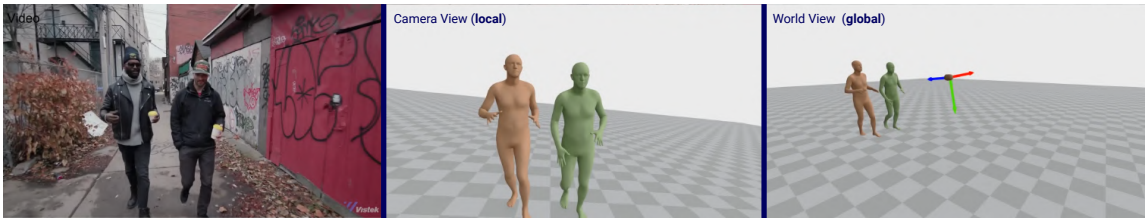


Figure 3.1: Visualizing camera-local (center) and global placement (right) of SMPL meshes regressed directly from the input image (left).
Source: PACE: Human and Camera Motion Estimation from in-the-wild Videos, Kocabas et al. [27]

an initial guess of the mesh parameters is iteratively refined by minimizing some prescribed objectives as depicted in Figure 3.2 (bottom).

3.1 Local Motion Recovery

Local HMR accounts for the estimation of human meshes in a camera-based coordinate system with the camera itself usually lying at the origin. When operating on such a space it is impossible to accurately infer human motion as camera movements directly appear as translations of the meshes, except from when the cameras are static, fixed, in the world. In the latter scenario, the human motion can be correctly estimated up to a translational or similarity (i.e. scaled euclidean) transformation. As we focus on dynamic recording setups in this project, however, camera-local inference of the SMPL meshes can only act as an initial step; after inferring camera pose an additional disambiguation task is required for (at least) plausible global motions.

3.1.1 Regression Based Methods

In 2.3.2 the SOTA method for camera-local HMR was presented. This uses an encoder-decoder architecture based on ViT, and therefore it estimates the SMPL parameters in a single inference step. As such, HMR 2.0 is categorized as a regression-based method. In general, all methods that estimate SMPL parameters using visual learners fall in this category, and usually have an image encoding and an SMPL decoding network as depicted in Figure 3.2 (top). The main advantage of these techniques is their efficiency; the parameter estimates are produced in a matter of ms (modern GPU time) as a single forward pass through the learner usually suffices.

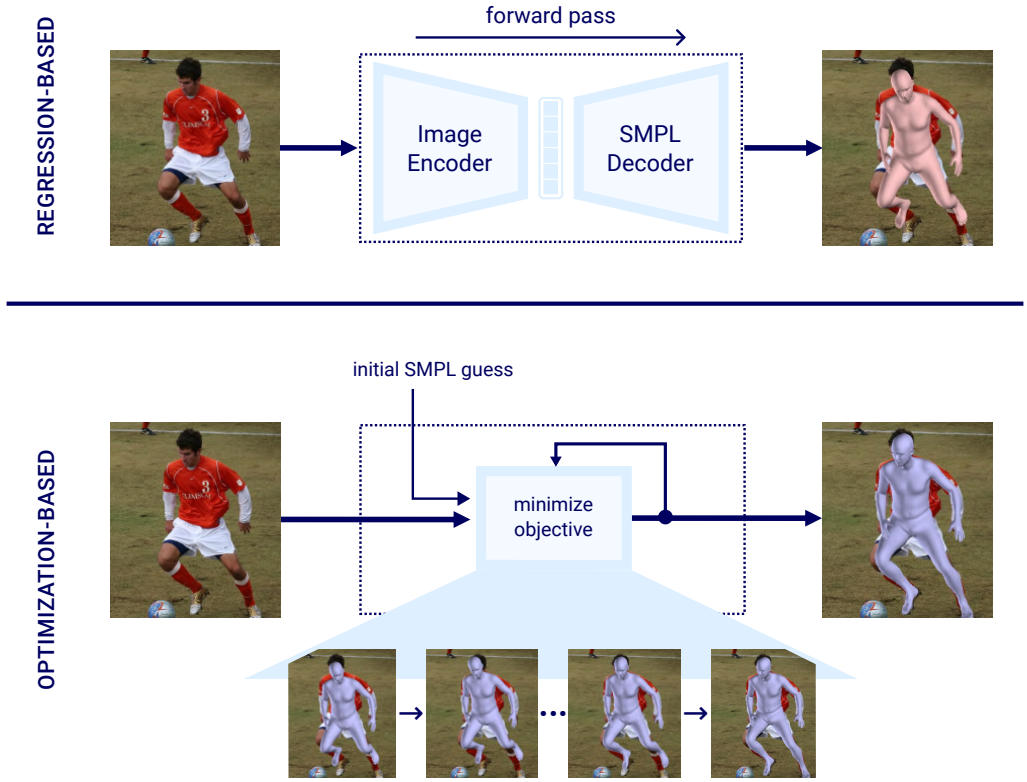


Figure 3.2: Schematic of the generic paradigms of regression-based (top) and optimization-based (bottom) methods to estimate SMPL parameters given a monocular image as input.
Source: Images are taken by the work of Kolotouros et al., *Learning to Reconstruct 3D Human Pose and Shape via Model-fitting in the Loop* [28]

3.1.2 Optimization Based Methods

While methods that directly regress mesh or model parameters operate with high efficiency they are known to produce implausible or erroneous fits, especially when those are projected to the input image [5, 42]. One prominent such example is that projected joints of camera-local regressed SMPL meshes tend to have a constant bias from the 2D keypoints that are provided (or estimated, see 2.1.4) [28]. Therefore, a number of techniques have been presented that iteratively enhance an initial estimate of SMPL parameters in order to derive plausible and re-projection-consistent human meshes, the most popular of which are described below.

SMPLify [5] was among the first methods to successfully optimize SMPL mesh parameters towards minimizing a re-projection objective. In particular, it used a CNN to estimate 2D keypoints of the human joints, which were then used to provide the SMPL update direction: the mesh parameters were altered so as to minimize the difference between the projection of the posed SMPL model joints and the detected keypoints. In addition, the authors proposed a inter-penetration criterion to prevent impossible poses due to self-penetrating body parts. The main drawback of this method, is its high sensitivity to the mesh initialization, as well as its proneness to local minima (as is always the case with gradient based optimization).

SPIN [28] builds on SMPLify and improves the initial state of the optimization by training a regression-based network of SMPL parameters from the input image. More specifically, the model employs a CNN trained to predict the SMPL shape parameters, $\vec{\beta}$ and joint rotations, $\vec{\theta}$ directly from the pixels. While this network produces meshes with the problems noted on the previous subsection, its output is still a much better initialization point for SMPLify than the 2D pose keypoints. Then, SMPLify iteratively refines the parameters to minimize the re-projection error. The process is repeated by feeding the optimization output back as input until the optimization procedure yields negligible modification. SPIN, though more computationally intensive, it yields significantly better results than SMPLify with pose-based initialization. The entire pipeline is depicted in Figure 3.3. In this project, **we draw significant inspiration from SPIN's core idea** in this project; we expand this concept to global human tracks estimation by combining a regression based with an iterative optimization method.

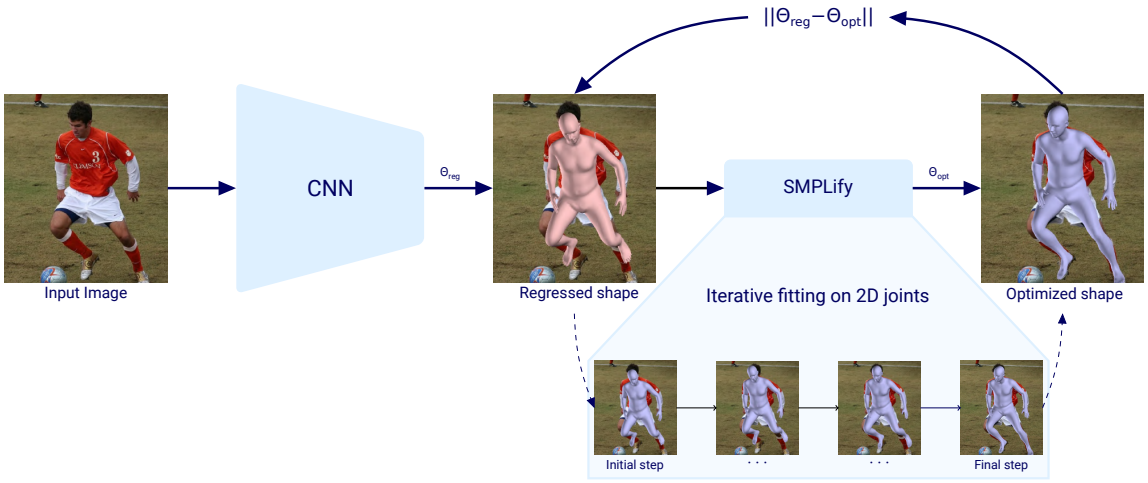


Figure 3.3: SPIN trains a deep network for 3D human pose and shape estimation through a tight collaboration between a regression-based and an iterative optimization-based approach. During training, the network predicts the SMPL parameters $\left(\vec{\beta}, \vec{\theta}\right)_{\text{reg}}$, which are used to initialize SMPLify, the iterative optimization routine that fits the model to 2D keypoints, deriving the updated parameters, $\left(\vec{\beta}, \vec{\theta}\right)_{\text{opt}}$.
Source: Learning to Reconstruct 3D Human Pose and Shape via Model-fitting in the Loop, Kolotouros et al. [28]

3.2 Global Motion Recovery

While camera-local motion recovery deals with estimating mesh motions relative to the camera coordinate system, global HMR focuses on placement of the meshes (and the camera) in a fixed world frame. As the sole input to the system is a video shot by (a potentially moving) monocular camera, estimating the local human tracks is almost inevitably a starting point due to the intrinsic difficulty of simultaneous regression of global mesh and camera parameters. Therefore, in order to estimate global human motion, an extra step is needed, that of ”lifting” the camera-based motion to the world coordinates.

Figure 3.4 is an updated version of the initial schematic of our approach (see Figure 1.1) where the separate stages of camera estimation, local mesh regression, and global motion recovery are visible. We begin this section by describing *WHAM*, a recent regression-based approach that tackles the local-to-global lifting problem, while followingly the *SLAHMR* optimization-based method for the same task is presented. In this project, both of those methods are employed as described in the corresponding subsections.

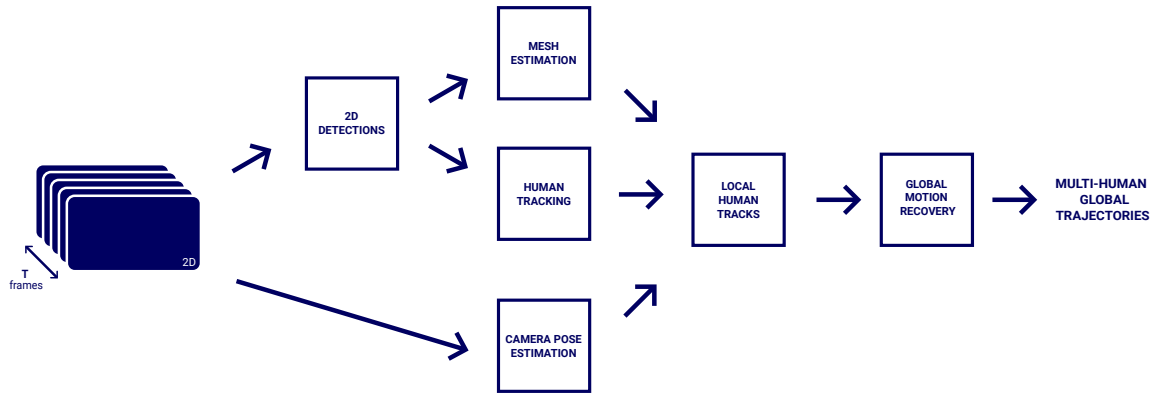


Figure 3.4: Increased level of detail of the schematic of our developed pipeline, underlining the distinct steps dedicated to camera pose estimation, camera-local mesh regression, and global motion recovery.

3.2.1 WHAM (Regression)

As has been shown in recent global motion estimation literature [66], (detected) 2D pose sequence conveys semantic information necessary to infer plausible motions that describe actions, apart from providing the re-projection error term that needs to be minimized by every HMR system. It is therefore natural in the context of global motion regression to combine camera-local deep estimator constructs with motion semantics in order to have pixel-aligned yet meaningful 3D motions; *WHAM* is model presented recently that successfully implements this idea.

Proposed by Shin et al., *WHAM* [50], trains an end-to-end model for global human motion capture from monocular videos. A camera-local encoder-decoder architecture is used to estimate SMPL parameters in camera-coordinates. It uses the cropped image patch and the output of the trained 2D pose encoder as inputs, while it also produces estimated feet-ground contact probabilities (two per foot, heel and big toe). This is trained to produce SMPL meshes with minimal re-projection error. To account for global motion, mesh orientation and translation in world coordinates need to be estimated; WHAM uses the 2D pose embedding and camera pose estimated using off-the-shelf SLAM [53], to generate initial world mesh poses in a recurrent fashion. A last stage integrates the estimated ground contacts to account for physically possible global motions. It should be noted that those two stages, depicted in Figure 3.5 bottom, solely output a sequence of mesh orientations and translations; the rest SMPL parameters are maintained from the camera-local ”path” (Figure 3.5 top).

In our port of WHAM, we replace camera-local estimates with the ones produced by the

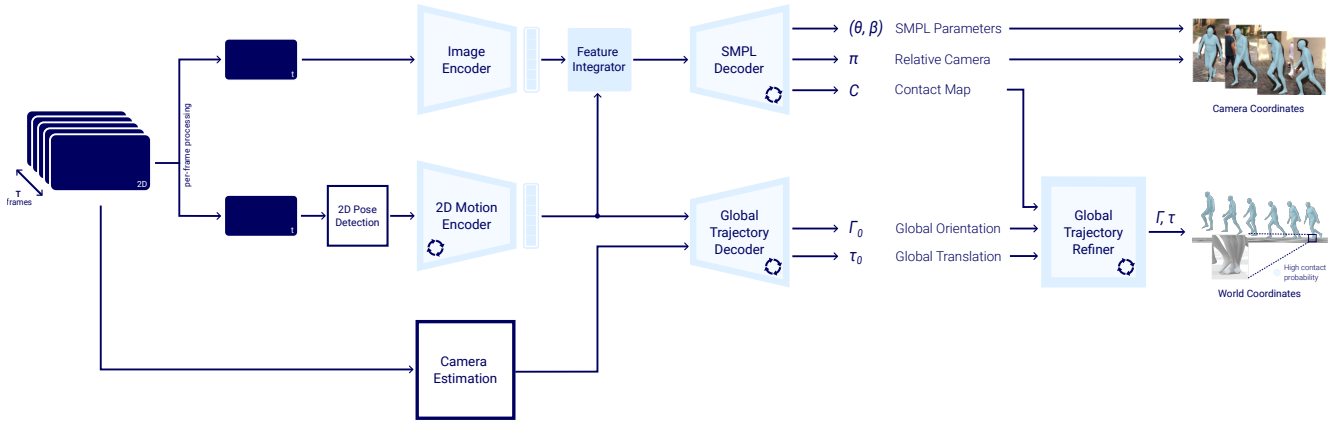


Figure 3.5: Schematic of the WHAM pipeline. Given a sequence of 2D keypoints estimated by a pretrained detector it encodes it to the motion feature that is fed to the global trajectory decoder along with estimated camera poses. The trajectory refiner updates the global mesh pose using the foot-ground contact, while the SMPL decoder estimated camera-local mesh parameters and ground contact probabilities. The final output of WHAM is pixel-aligned 3D human motion plus global pose placement.
Source: Reconstructed from *WHAM: Reconstructing World-grounded Humans with Accurate 3D Motion*, Shin et al. [50]

SOTA method HMR 2.0 (see 2.3.2). In addition, we use contacts estimated from BSTRO which is more accurate than WHAM and also provides richer contact information (see 2.3.4). We also use a more efficient camera estimator, based on VO rather than the full Visual-SLAM. In order to use the learned motion semantics by the ”2D Motion Encoder” and global mesh regressors, those parts of WHAM are used pretrained and frozen, as will be explained in more detail in the next chapter.

3.2.2 HuMoR (Motion Prior)

Before introducing the used global motion optimization framework, hereby a motion prior is introduced that is used to plausibly ”animate” inferred meshes in the world coordinates. Human motion priors are actually necessary components of all optimization-based methods as there is no useful input cue related to global 3D motion, when capturing on monocular settings. In addition, this objective ensures that human bodies perform realistic motions when seen exclusively in the global frame of reference. *HuMoR* presented by Rempe et al. [45], is the most popular among the SMPL-based motion priors, and it is based on the premise that realistic human motion exhibits smooth transitions and adheres to physical constraints. To model this, it defines a state representation using SMPL pose parameters, global mesh pose, joint angles, their corresponding velocities (angular for angles, linear otherwise), and feet-

ground contact.

The core of HuMoR is a conditional Variational Autoencoder (cVAE) that is shown in Figure 3.6. The model receives a pair of states at consecutive timesteps (t-1 and t) as input and is trained to reconstruct the state at time t. The latent prior is conditioned on the previous state. This training process teaches HuMoR to learn a powerful probability distribution of realistic human poses and their transitions. Once trained, HuMoR can act as a powerful prior for motion estimation tasks. If presented with a new state pair, the model can calculate provide two loss terms:

- Prior Loss: This measures the distance of the conditional prior distribution from the standard Gaussian distribution (i.e. the prior of the conditional prior) using the Kullback-Leibler (KL) divergence.
- Posterior Loss: This measures the difference between the encoder's output (i.e. the posterior distribution of the latents) and the conditional prior network's output; in practice it is the KL divergence of two Gaussians as both networks predict Gaussian mean and variance parameters.

By combining these losses, HuMoR helps ensure that during motion estimation, the generated poses are not only likely under "normal" movement but also maintain fluidity and realism as defined by its implicit understanding of transition dynamics.

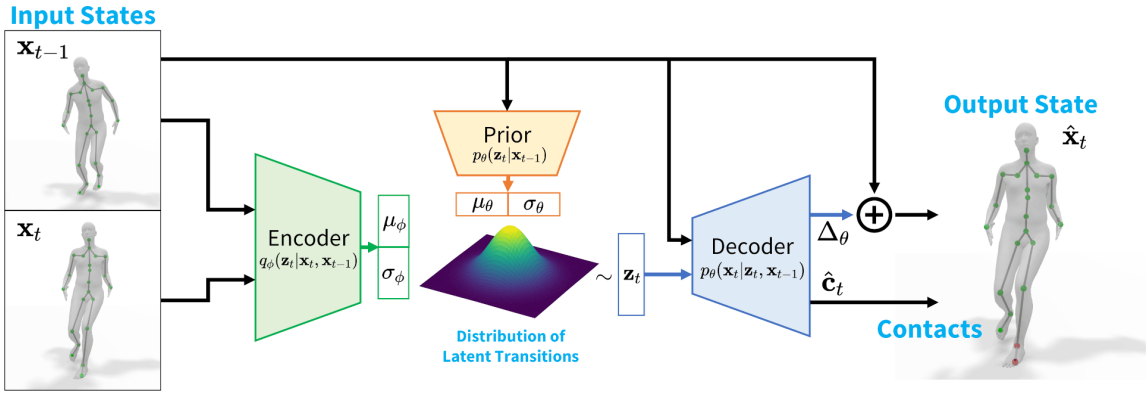


Figure 3.6: HuMoR cVAE Architecture. During training, given the previous state x_{t-1} and ground truth current state x_t, the model reconstructs $\hat{x}_t$ by sampling from the encoder distribution. At test time HuMoR can (i) generate the next state from x_{t-1} by sampling from the prior distribution and decoding, (ii) infer a latent transition z_t with the encoder, or (iii) evaluate the likelihood of a given z_t with the conditional prior. For motion optimization the latter is used.
Source: HuMoR: 3D Human Motion Model for Robust Pose Estimation, Rempe et al. [45]

3.2.3 SLAHMR (Optimization)

A very effective approach for local-to-global motion lifting using optimization only is the recently presented method *SLAHMR*. Proposed by Ye et al. in their work *Decoupling Human and Camera Motion from Videos in the Wild* [63], SLAHMR, defines the following hand-crafted optimization objectives to convert the camera-local human tracks to realistic global motions using the estimated camera trajectory (see Figure 3.7):

- **Joints re-projection**: the regressed SMPL joints should be close to the (detected) 2D keypoints when projected using the regressed camera pose and scale. The optimizer alters the SMPL parameters (mainly global orientation and translation) to minimize this criterion.

- **Motion prior**: the sequence of SMPL parameters should comprise plausible temporal transitions that not only are smooth but also realistic. To encourage this, the optimizer refines the SMPL parameters in pairs by maximizing the transition likelihood of a learned human prior, *HuMoR* [45], which was explained in 3.2.2.

- **Smoothness of the parameters**: as smooth parameters play a key role in perceived motion fidelity, SLAHMR employs parameter smoothing during the optimization in the following ways:

    - shape smoothing by using a standard Gaussian prior (or equivalently by applying weight decay to the $\vec{\beta}$ parameters)

    - pose smoothing in the latent space of *VPoser* [42], a Variational Autoencoder (VAE) for the SMPL body pose $\vec{\theta}$, using a standard Gaussian prior (for the latents)

    - temporal smoothing of the joints regressed from the SMPL mesh (as this regression is differentiable the errors will be back-propagated to SMPL parameters)

In our port of SLAHMR, we use all of its objectives as originally defined. The main difference is that the input to the ”Minimization” stage is the global track obtained from the local-to-global regression stage; in this project we use WHAM to provide this, a model presented in 3.2.1.

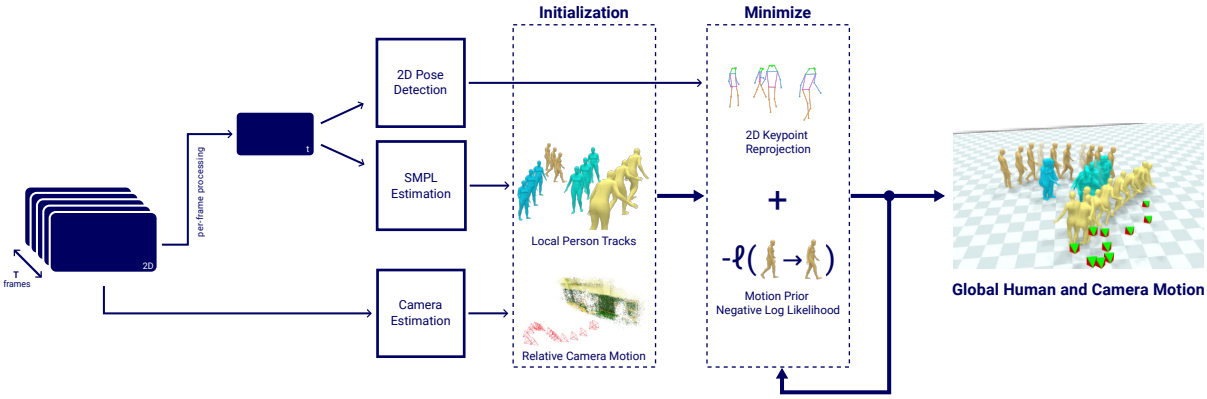


Figure 3.7: Schematic of the SLAHMR optimization proposal. Given a sequence of frames wherein each 2D keypoints and local SMPL parameters have been estimated, while camera trajectory has been inferred using SLAM, the optimization optimizes for accurate re-projection and motion likelihood. As a result, the camera-local human tracks are converted to realistic global motions as shown on the right.
Source: Reconstructed from *Decoupling Human and Camera Motion from Videos in the Wild*, Ye et al. [63]

Chapter 4

Monocular Dynamic Motion Capture: A Regression-Optimization Hybrid Approach

After having described how useful quantities for motion capturing can be inferred from the input pixels, as well as ways to reason about local and global motion patterns in either a regression or an optimization based manner, its time to present the developed system for the purposes of this project. An overview of our pipeline with all the implemented nodes is given for reference in Figure 4.1, while in the sections that follow the technical details of the intermediate components are given along with the extensions of literature works towards a *Regression-Optimization Hybrid Approach for Monocular Dynamic Motion Capture*.

4.1 System Architecture

As was mentioned in the first chapter, the developed system, comprises a modular, ETL-based, architecture where each processing node operates on aggregated outputs of nodes from the previous level; either in frame- or in patch-wide context. In the rest of this section, ordered processing steps and technical details of the employed models are presented. In Table 4.1), all the models that are used in this project along with the corresponding number of trained parameters are listed for reference. We remind the reader at this point that the overall aim of the developed pipeline is to generate plausible

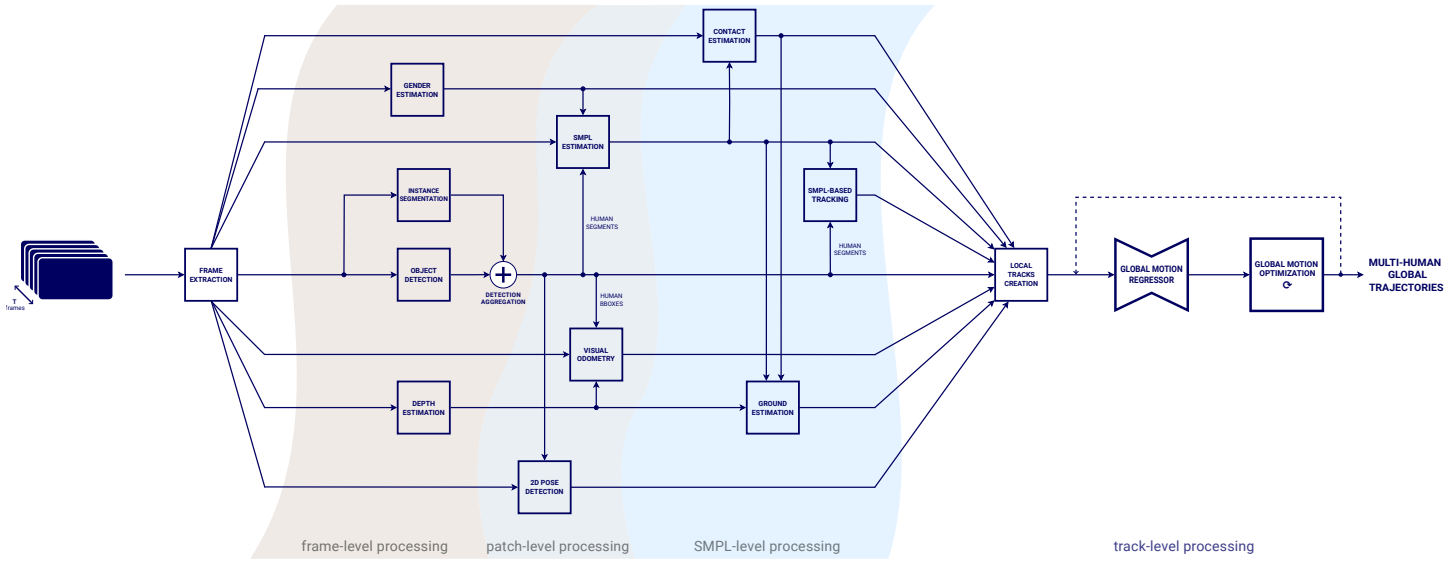


Figure 4.1: Our implemented pipeline in full level of detail. The monocular input video is processed through a series of nodes that are grouped into four categories based on the working domain: *frame-level*, that operate on the input frames, *patch-level*, that operate on each human detection separately, *SMPL-level*, that operate or based on the SMPL parameters, and *track-level*, that optimize the entire motion. The connections denote dependencies in the execution graph. Note that this figure is best seen in zoom.

multi-human global trajectories given a monocular (RGB) video as input.

4.1.1 Frame-Level Processing (first level)

The first level of execution nodes is directly fed with the input frames. This comprises object detection and segmentation, gender and depth estimation, as well as detection aggregation. In contrast to those, nodes deeper into the execution graph are either operating on a patch (i.e. cropped detection) level or on the estimated SMPL parameters of the detected humans. The technical details of this level nodes' implementation follows.

Human Detection

Our object detector, Co-DETR [67], presented in 2.1.1, receives as input a frame of the input video and outputs a set of bounding boxes and corresponding confidence scores for the *person* class. The model receives the frame resized and padded to $1024 \times 1024px$ in the configuration used, and uses 236M parameters (Co-DETR-Large). We use the OpenMMLab's `mmdetection` [8] implementation of Co-DETR, which among others provides utilities to resize detections from the dimensions required by the model

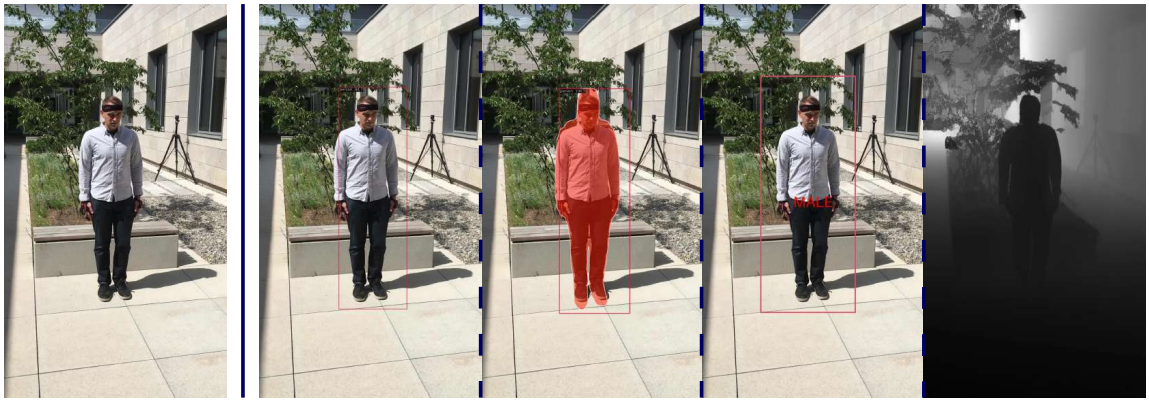
Table 4.1: Details of the models used in our pipeline. Depth denotes execution graph generation index, while RNN denotes Recurrent Neural Network (mainly LSTM [20]) backbone, and Stable Diffusion v2 was presented in [47]. The models are sorted in ascending depth and parameter count order.

Depth	# Parameters	Model	Task	Network Type
1	3.4M	DPVO	Visual Odometry	RNN
	68.1M	Marigold	Depth Estimation	Stable Diffusion
	108.1M	ConvNext v2	Instance Segmentation	CNN
	236M	Co-DETR	Object Detection	Transformers
2	95.6M	MiVOLO	Gender Estimation	Transformers
	308.5M	ViTPose	2D Pose Estimation	Transformers
	670.5M	HMR 2.0	SMPL Estimation	Transformers
3	23.3M	PHALP	Tracking	DeepSORT
	39.5M	HMAR	Texture Estimation	CNN
	243.1M	BSTRO	Contact Estimation	Transformers
4	0.67M	Vposer	SMPL Pose Encoding	MLP
	9.7M	HuMoR	Human Motion Prior	MLP
	47.7M	WHAM	Global Motion Recovery	RNN

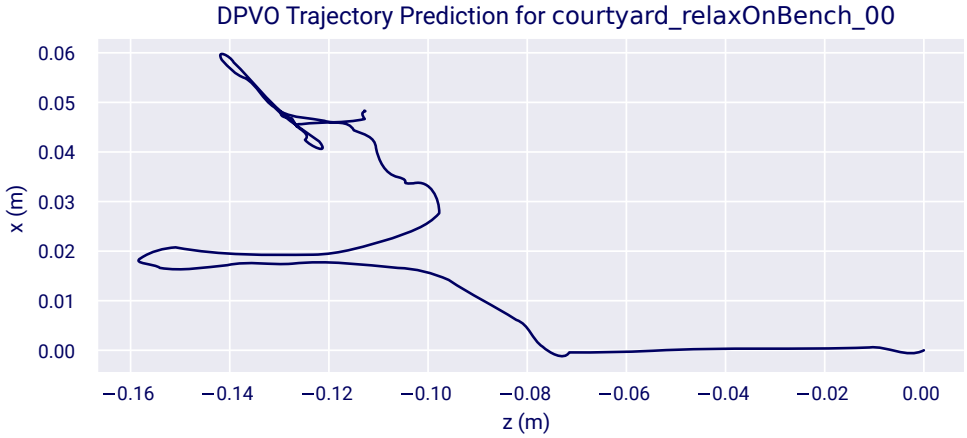
to the original frame dimensions. Example of an input frame and the corresponding person bounding box and score is given in Figure 4.2.

Human Segmentation

For instance segmentation, we opted for ConvNext v2 [60], presented in 2.1.2. It receives the same input as Co-DETR, i.e. a frame of the input video, and outputs a set of bounding boxes, instance segmentation masks, and corresponding confidence scores for the *person* class. ConvNext operates on images resized to $224 \times 224px$, while in the configuration used it contains 108.1M parameters (ConvNextv2-Large). We use the `mmdetection` implementation of ConvNext v2 as was the case with Co-DETR. An example of the same input frame with the corresponding person mask and score is given in Figure 4.2.



(a) Frame-level Detections



(b) Camera Trajectory

Figure 4.2: Output of first-level processing nodes for an input frame of the dataset considered in this project, *3DPW*. (a) From left to right: input, object detection, instance segmentation, gender, and depth estimation. The image and the outputs are (intrinsically or scaled to) $1080 \times 1920px$. (b) Camera trajectory (X-Z translational component) estimated using DPVO [54].

Gender Estimation

Genders are estimated using MiVOLO [31], described in 2.1.5. It also receives a frame of the input video and outputs the gender and age of every detected person (and/or face). MiVOLO operates on images resized to $224 \times 224px$, while in the configuration used it contains 95.6M. We use the PyTorch-based implementation of MiVOLO's authors. In the aforementioned example of Figure 4.2, the corresponding detections with annotated genders are also given. It is noteworthy, that MiVOLO exhibits significant bias towards male predictions, which can (partially) be attributed to the large covariate shift derived from the difference between the dataset we are using , *3DPW*, and the one it was trained on, *Lagenda* [31]. In an attempt to compensate for this bias, we require lower female thresholds when we account for track-wide genders in 4.1.4. We underline

that all detections, coming either from the object detector, the instance segmentor, or the gender estimator, are associated and aggregated based on inter-box weighted IoU (the weights being the detection scores).

Depth Estimation

To estimate frame-wide depth maps, we use Marigold [25], described in 2.3.1. It receives the same input as the detectors, i.e. a frame of the input video, and outputs the relative depth for every pixel, with a maximum of 16-bit level storage or $65K$ discrete depth levels. Marigold operates on images resized to $384 \times 512px$ (width $\times$ height), while in the configuration used it enfolds 68.1M parameters in the autoencoder and denoising networks. We use the Hugging Face's `diffusers`-based implementation which automates noise sampling and denoising, towards regressing the depth. Example of an input frame and the corresponding depth map is given in Figure 4.2. We remind the reader that we estimate depth to assist the camera pose estimation by providing a better depth initialization for visual odometry. However, since our visual odometry also uses human detections, we defer its description to the patch-level processing subsection that follows.

4.1.2 Patch-Level Processing (second level)

In the next processing stage, nodes will analyze patches, which are cropped sections of the input frames. These patches are derived from the detected (and aggregated) human bounding boxes. Since these boxes typically fit closely around the human silhouette, and the models require a different aspect ratio, we need to create resized, human-centered image patches, that will then be compatible with the stage's DNN learners. This process involves two steps: processing the detections and isolating the human subjects, as outlined below.

Box Processing

First, we resize the detected human bounding boxes to match the required aspect ratios. We add padding to prevent the boxes from being too tight, which could cut off parts of the body due to detection errors. The humans remain centered during resizing, but the new bounding box includes more of the surrounding image. Since most models

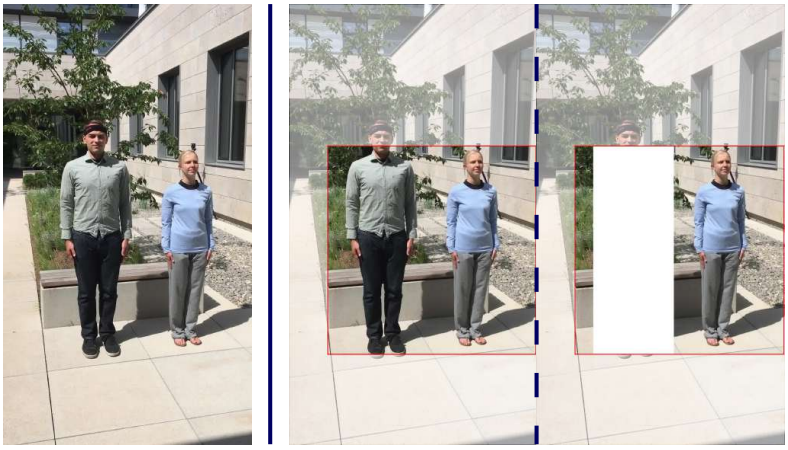


Figure 4.3: Comparison of original (center) and isolated (right) slicing of for an input frame (left) of the dataset considered in this project, *3DPW*. In this example, the detected bounding box corresponding to the female, was first converted to a square aspect ratio before slicing. As a result, part of the male subject is also included in the original, non-isolated, sliced patch, whose tight box is masked-out using white color.

require square image patches (especially for re-projecting 3D data), we ”squarify” the bounding boxes. Here’s the complete processing for each bounding box:

1. $5px$ padding is applied to the original (tight) bounding boxes

2. The boxes are resized so as to have equal widths and heights

3. The bounding boxes are translated inside the frame (in case they overflow the frame after resizing)

4. For the bounding boxes that spill outside the frame, cropping is performed which may result to aspect ratio offsets

As shown in Figure 4.1, this processing level includes nodes that estimate SMPL parameters and 2D pose keypoints for detected human subjects. Figure 4.4 provides an example input patch (two people are shown, the focus is on the female), along with the output from this level’s nodes. Note that the ’squarification’ of bounding boxes can occasionally introduce aspect ratio errors, even for boxes that fit within the frame. This primarily affects the accuracy of mesh orthographic projection to the patch (which assumes square frames), an issue acknowledged in relevant literature [32]. Future updates to the box resizing code should address this issue, which is also visible in the SMPL estimation example in Figure 4.4.

Figure 4.4: Output of second-level processing nodes for an input patch of the same frame as in Figure 4.2. From left to right: input patch derived by resizing and squarifying the female's tight bounding box, estimated 2D pose keypoints in COCO 17 format, projected SMPL mesh generated by the estimated parameters, and its corresponding 2D keypoints. Note that small errors in the resizing of the bounding box that resulted in it being non-square, were propagated to the projection of SMPL mesh and joints. This figure is best seen in zoom.

Subject Isolation

Resizing the original bounding boxes can lead to other human subjects partially appearing within the box of interest. This is especially common when multiple people are close together in the input image. As a result, 2D detectors requiring human-centered inputs may fail or underperform. To mitigate this, we isolate subjects within their resized bounding boxes by "whitening out" pixels that belong to other nearby tight bounding boxes, as shown in Figure 4.3. If two bounding boxes overlap, we whiten all pixels belonging to other humans, except those within the focused human's segmentation mask. This maintains visual cues while isolating the subject during patch creation. Importantly, we found that whitening out these regions instead of blackening them, significantly improved performance. This may be because models have been trained to interpret black as shadows or corners, causing confusion when used for masking.

2D Pose Estimation

As was explained in 2.1.4, estimating 2D pose directly from the pixels is quite useful as it can act as the supervising signal for optimizing the human tracks and inferring global human motion semantics. To estimate pose keypoints for the detected persons, we

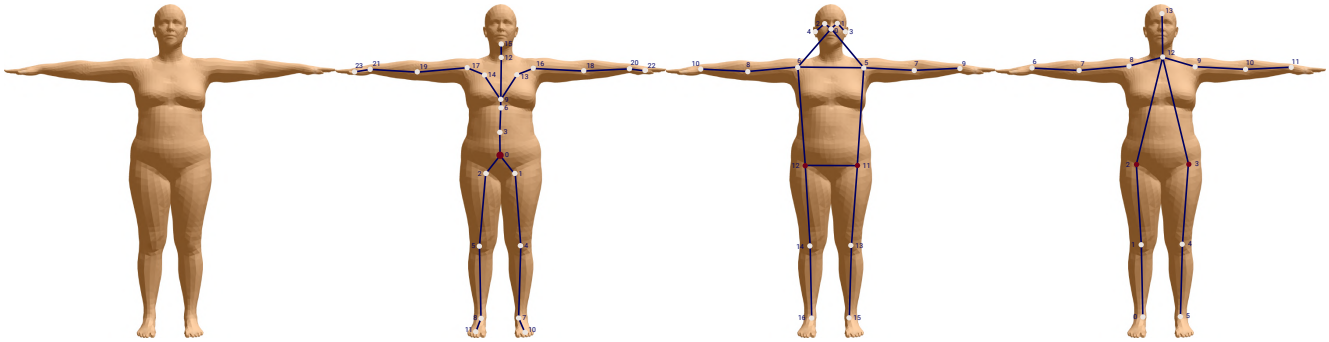


Figure 4.5: Visualizing (projected) joint representations and corresponding skeletons. From left to right: input mesh, the 24 joints regressed from SMPL, the 17 joints from 2017 COCO Keypoint Challenge that is used by ViTPose, and the 14 MPII [1] joints used for metrics computations and given here for reference. The red keypoints refer to the pelvis joint that acts as the mesh root (i.e. the mesh frame is placed at the pelvis). When there's no pelvis joint (3rd and 4th column) the hip joints are used to approximate it. This figure is best seen in zoom.

use ViTPose [62]. It receives as input a patch with the person in focus isolated from others, and outputs the (x, y) pixel coordinates of body joints. ViTPose was trained to predict 17 keypoint heatmaps, for the joints defined in the COCO dataset [35] and 2017 Keypoint Challenge. In contrast, the SMPL defines 24 in its joint regressor, **J**. In Figure 4.5 we present the joints and kinematic tree of the two models as well as the joint representation used for evaluating the models, coming from MPII pose estimation dataset [1] . ViTPose operates on human-centered patches resized to $256 \times 256px$, while in the configuration used it enfolds 308.5M parameters. We use the OpenMMLab's `mmpose` implementation of ViTPose which handles heatmap sampling and visualization of the detected pose. Example of an input patch and the corresponding 2D pose is given in Figure 4.4.

SMPL Parameters Estimation

In subsection 2.2.2 we presented the chosen parametric representation of human body in 3D, SMPL, while in 2.3.2 the SOTA method for estimating SMPL parameters from images, HMR 2.0 [15] was explained. HMR 2.0 receives as input a patch with the person in focus isolated from others, resized to $256 \times 256px$, and outputs the following:

- $\vec{\theta}$: SMPL body pose comprising the joint rotations relative to their parent in the kinematic tree, i.e. a 23×3-dimensional vector (the pelvis joint has no parent). The rotations are regressed in angle-axis format (popularized by the work of Olinde Rodrigues [46]).

- $\vec{\beta}$: SMPL body shape parameters, a 10-dimensional vector with the weights of the 10 first PCA coefficients.
- $\vec{\pi}$: Relative camera translation and scale. The translation considers normalized offset from the patch center, while scale is used alongside a pre-defined focal-length[1] to estimate camera-to-mesh translation along the Z-axis. The default focal-length value of HMR 2.0 is $5000mm$.

We use the default version and implementation of HMR 2.0 which is based in PyTorch and operates on neutral genders only. As a result, when we try to use the estimated parameters for gendered SMPL mesh generation, we get small errors (most prominently the heights are off as males have different height distributions than females and normal genders, which is captured in the PCA of the shapes). To alleviate this we regress gender-neutral parameters for this stage but replace with the estimated gender when we do the global track optimization. An example of gendered mesh placement along with SMPL-derived joint locations is given in Figure 4.4.

Visual Odometry

We employ the SOTA Visual Odometry method, DPVO [54] (introduced in 2.4.1), for camera pose estimation. While DPVO typically operates on entire frames, we optimize its use by leveraging human detections. VO algorithms underperform when visual samples include dynamic foregrounds, as they rely on correspondences from static regions for camera displacement estimation. Scenes in HMR datasets often feature centrally-located humans in motion, increasing the likelihood of the correspondence matching algorithm sampling foreground pixels. DPVO, like other VO methods, uses gradient-based sampling of image pixels, i.e. it draws the samples for high intensity-gradient image regions. We enhance this process by zeroing out gradients within human segmentation masks, effectively discouraging the sampler from selecting those pixels.

One of the key components and factor to DPVO's effectiveness is its ability to jointly optimize for camera pose and patch depth for each of the tracked patches through time. This optimization is initialized by randomly assigning depth values to the pixel centroid of each patch, which is sub-optimal and the actual cause of odometry errors

[1]The focal length is the distance between the center of a lens and the point where it focuses light rays to form a sharp image on the camera sensor.

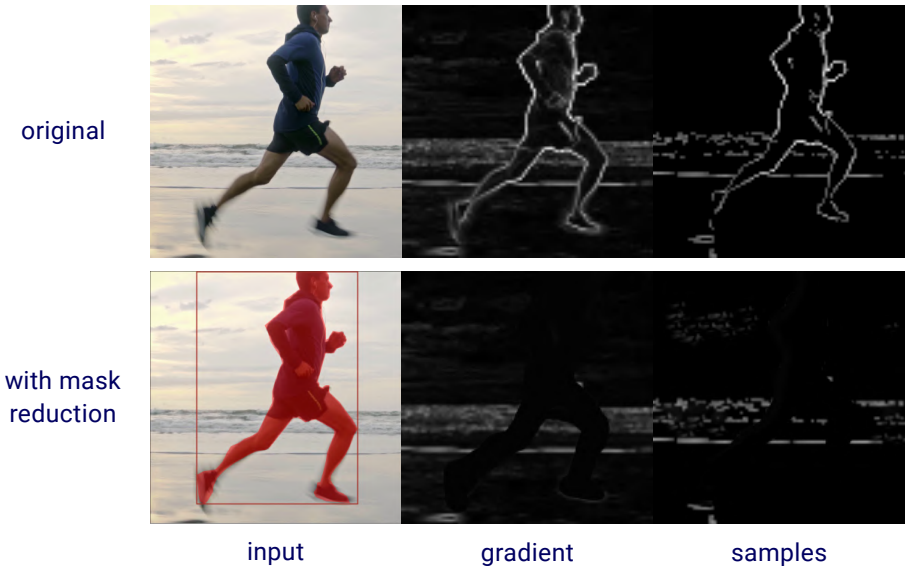


Figure 4.6: Effect of zeroing-out the intensity gradient within human segmentation masks (our improvement to DPVO's correspondence sampling). From left to right: input image, image gradient (after 4x downsampling), and 1000 binomial samples from it (gradient as probability mass function). Top: DPVO's standard sampling (note pixels at foreground border are likely selected). Bottom: Our improved sampling with foreground masking, reducing the selection of sky pixels.

or failure [54]. To alleviate this, we extend DPVO by incorporating estimated per-pixel depth values as described above. In particular, the patch depths are extracted and max-pooled to match the spatial dimensions of the patch features extracted from DPVO's feature extractor, *ResNet* [18]. Then, instead of randomly initializing patch depths we use the values from monocular depth estimation.

4.1.3 SMPL-Level Processing (third level)

After regressing mesh parameters for each detected human, we operate in a combined 2D-3D domain. This allows us to perform texture-aware human tracking, estimate scene contacts, and fit local ground planes using visual cues, as well as SMPL body pose and shape. These components are essential for creating concise, camera-local 3D human tracks that will subsequently be converted to global ones using the estimated camera trajectory. We enumerate this stages' nodes followingly.

Appearance-based Tracking

In 2.3.3, we introduced our chosen tracker, PHALP [43], which extends DeepSORT with information from estimated SMPL meshes (pose, global placement, and regressed UV

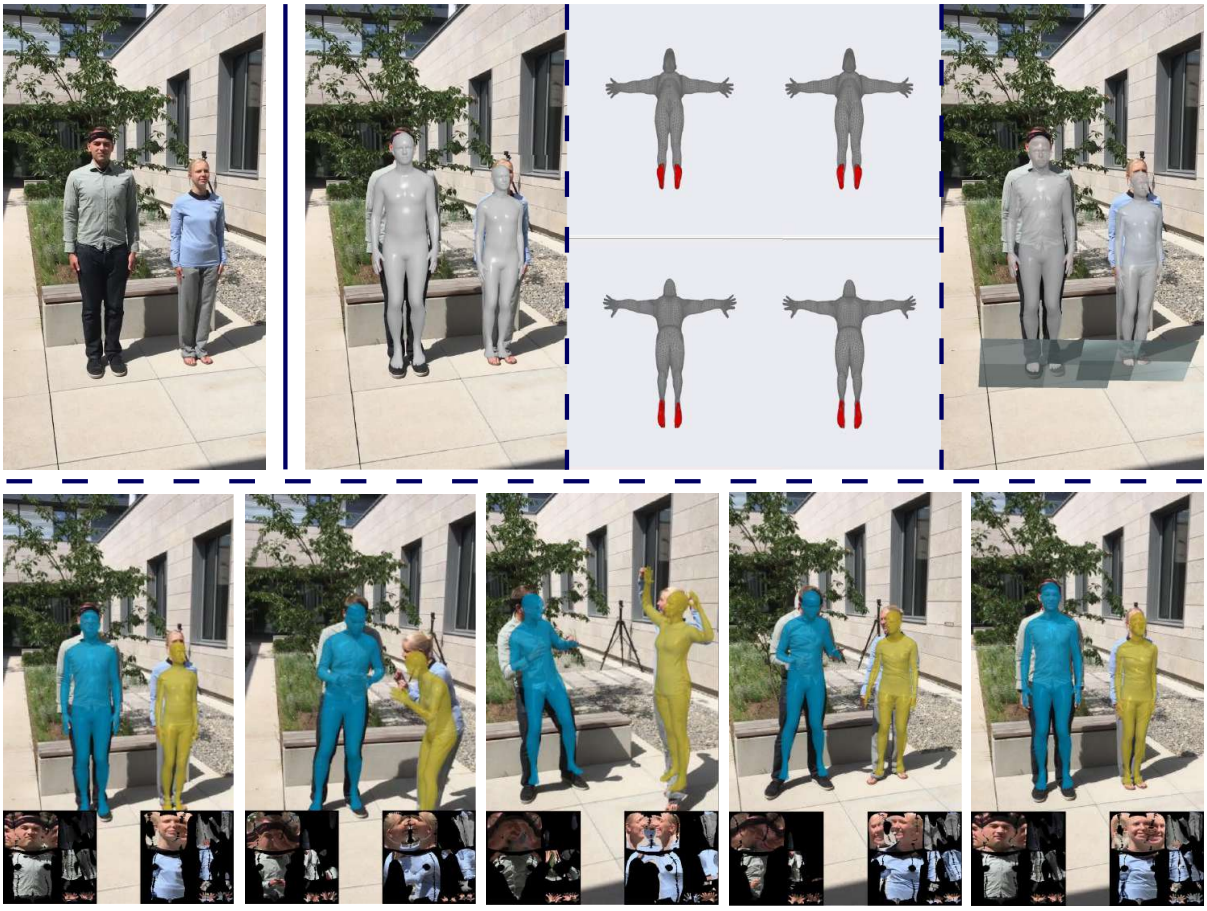


Figure 4.7: Output of third-level processing nodes for an input frame of *3DPW*. Top, from left to right: input, SMPL mesh fitting, contact estimation (front and back views for each subject), and ground estimation. Note the grounds are regressed per subject, i.e. they constitute local ground portions. Bottom: appearance aware tracking and textures visualization (different colors correspond to IDs).

texture). PHALP employs a CNN for regressing projected SMPL mesh textures and comprises 23.3M learnable parameters. Figure 4.7 visualizes tracking for the same frame as in Figure 4.4, also demonstrating the regressed UV textures. Importantly, we observed that disabling texture-based matching in PHALP increased the likelihood of identity switches and, crucially, identity swaps. The latter occurs when two subjects are in proximity, and the tracker exchanges their IDs permanently.

Contact Estimation

As introduced in 2.3.4, we employ a monocular SMPL contact estimator, BSTRO [21], to derive the probability of each mesh vertex being in touch with scene surface. BSTRO's CNN backbone, HRNet [58], and transformer mesh encoder, METRO [34], encapsulate 243.1M parameters for processing cropped patches resized to $224 \times 224px$. We

exemplify regressed contact information for the meshes of the input frame in Figure 4.7. We employ mesh contact estimates in two major ways: first we fit ground planes to the vertices in contact, and second we pool contact information around feet and hand centroid vertices to regress coarse foot-hand contact information. The latter is performed by max-pooling a radius of 10 vertices around pre-defined vertex centroids associated with foot and hand joints. Both are used in global motion optimization of the regressed human tracks.

Ground Estimation

Global motion optimization, and, in particular, HuMoR model [45] presented in 3.2.2, needs ground information for converting SMPL parameters to a ”canonical” representation, i.e. normalized with respect to distance from the ground, body height, and camera parameters. We follow the steps below to estimate ground planes per detected human and frame:

1. We aggregate all vertices with contact probability greater than 0.5 from all SMPL meshes in a given frame.
2. The vertices are clustered based on vertex density using the DBSCAN algorithm [12] into groups of spatially close vertices in contact.
3. For each cluster, a plane is fitted using least-squares estimation.
4. Each SMPL is associated with a plane by minimizing the distance of its vertices in contact. If feet belong to those vertices, they are examined prior.
5. The estimated planes are merged per-frame and filtered in time, so that for every SMPL mesh the most recent plane with its feet in contact is kept. This is done to prevent non-physical plane associations for instance when feet are not in touch with the ground.

In 4.7 we show regressed planes for the detected humans and corresponding regressed meshes.

4.1.4 Track-Level Processing (fourth level)

After completing appearance-aware human tracking across frames and estimating essential 3D properties (like vertex contacts and ground planes) for the regressed

meshes, we are ready to assemble complete human tracks. Since the SMPL meshes were estimated in camera coordinates, these initial tracks are camera-local. In this subsection, we'll outline the steps to convert these local human motions to the world frame, as introduced in the previous chapter.

Local Human Tracks Creation

To create the initial human tracks, we first gather outputs from all parent nodes and associate them using tracking information. Additionally, we must address potential detection or tracking misses (e.g., due to occlusions) by infilling missing observations. We start by extracting the track length and visibility from the tracking node, which we denote as:

$$
\begin{aligned}
t_s^{(p)} &: \text{start frame of p-th track} \\
t_e^{(p)} &: \text{end frame of p-th track} \\
V_i^{(p)} &: \text{1 if p-th track is visible in frame i, 0 otherwise} \\
& \quad 0 \leq i \leq N_{frames}
\end{aligned}
$$

Therefore, the track corresponding to the p-th detected person spans times (i.e., frames) $T^{(p)} = \{t : t_s^{(p)} \leq t \leq t_e^{(p)}\}$, which we denote with T for simplicity with the person index p implied by the context. The tracking node provides the start and end times of each track, which we determine by examining the first and last occurrence of each track ID. Based on the frames where the tracked person is detected, we gather the outputs of the following nodes:

- **2D Keypoints**: We collect a sequence of $T \times 17 \times 2$ COCO-2017 keypoints for the frames where the person is visible. We use linear interpolation to fill in any missing frames for each keypoint independently.
- **SMPL Parameters**: We collect the SMPL parameters - shape ($\vec{\beta}$), pose ($\vec{\theta}$), global translation ($\vec{\tau}$), and global orientation ($\vec{\Gamma}$) - for the frames where the person is visible. We use linear interpolation for missing translations, and SO3 interpolation for pose and orientation (following the SLAHMR method [63]). While there are DNN-based methods for more realistic SMPL parameter infilling during long-term occlusions [65], we opt for the simpler approach of interpolation since our focus is on capture techniques. Subsequent stages in our process will

handle enhancing occluded motions.

- **Camera Poses**: From the VO system, we obtain a sequence of camera translations and rotations relative to the world frame (conventionally set as the first frame's camera location). The output is a $T \times 4 \times 4$ sequence of transformation matrices in homogeneous coordinates. DPVO's graph-based bundle adjustment handles missed frames by extrapolating the pose difference from the last known frame [54].

- **Ground Parameters**: or each SMPL mesh, we fit a local ground plane to the feet vertices (or other contact vertices). We collect these as $T \times 3$ parameters for normals and $T \times 3$ for offsets. We'll use these to estimate canonical mesh coordinates during global optimization with the HuMoR motion prior. Missing values are linearly interpolated.

Figure 4.8 (2nd row) visualizes input frames and corresponding 3D bodies for two human tracks. Note that camera information, though present, has not been used yet to position the trajectories in the world frame. Also, since SMPL parameters are estimated using human-centered image patches, the meshes initially appear overlapped (Figure 4.9). We address this by following common practice in global HMR solutions: estimating frame-wide camera translations with CLIFF's camera conversion [33], and then computing relative translations of all tracks to the first one. This also reduces the impact of camera movement on the local 3D human motions.

Regression-based Global Track Lifting

To transform trajectories from camera coordinates to a fixed world frame, we employ recurrent regression network. Specifically, we utilize the WHAM architecture [50], presented in 3.2.1, configuring it with 47.7M parameters in its recurrent and convolutional modules. Figure 4.10 illustrates how we leverage WHAM's 2D keypoint encoder, global decoder, and refiner modules. Notably, we provide the following inputs:

1. Motion Encoder: 2D poses estimated by ViTPose.
2. Global Decoder: Camera angular velocities estimated by DPVO.
3. Global Refiner: Feet contact probabilities estimated by BSTRO.

Figure 4.8: Visualization of human tracks recovery. Left to right: input frames from scene `courtyard_goodNews_00` of 3DPW [57], camera-local motion recovery, global motion lifting using regression, and using hybrid regression and optimization in world frame.

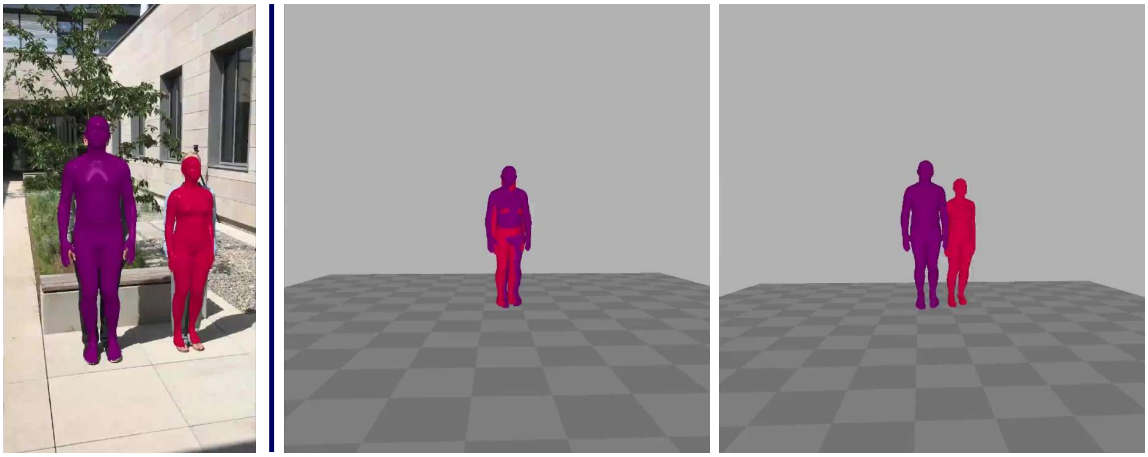


Figure 4.9: During camera-local mesh recovery, every subject is treated independently of the others, assuming it centered in its cropped image patch. As a result, we may get accurate re-projections (left), but all human meshes appear centered around in the origin in the world frame (middle). By maintaining relative offsets from the frame-wide regressed camera (right), we achieve much better global mesh placement.

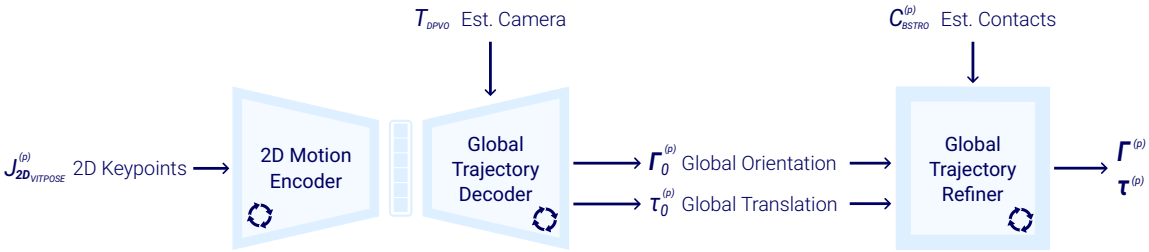


Figure 4.10: Our WHAM-based global trajectory regression network. We provide the network with the following inputs derived from ancestor nodes: keypoint sequences, camera poses, and contact probabilities. The regressor then estimates both initial and refined global mesh pose sequences, which are used to position the SMPL meshes of the p-th person within the world coordinate frame.

To generate the remaining SMPL parameters (excluding global translation and orientation), we rely on HMR2.0, making image input unnecessary for our WHAM-based global trajectory regressor. As demonstrated in Figure 4.8 (3rd row), WHAM effectively reduces jitter and produces more realistic human motions within a fixed frame. Additional visualizations can be found in the project's code repository.

Optimization-based Global Track Refinement

As can be seen in 4.8 (3rd row), our regressor produces globally plausible yet locally inconsistent meshes. More specifically, the projected mesh silhouettes deviate from the depicted ones whereas in camera-local meshes (2nd row) this is not the case. This is caused from WHAM modules weighting more the smoothness and realness

of the motion than its re-projected appearance. The first step we follow, therefore, is eliminate this by minimizing the euclidean distance of the projected SMPL joints and the detected pose keypoints from ViTPose. Then, following SLAHMR, we smooth-out the SMPL parameters and regressed joints so to reduce motion jitteriness in world frame, and finally we maximize the likelihood of SMPL transitions using HuMoR and the estimated local grounds. The optimization recipe we follow in this project closely follows that presented in SLAHMR [63], and has as follows:

(i) $[0 \leq t_{opt} \leq 250]$ **2D Keypoints**: minimize SMPL joint re-projection distance from detected keypoints by optimizing the global translation $\vec{\tau}$, orientation $\vec{\Gamma}$, and offset from camera (similar to $\vec{\pi}$). We start by optimizing $\vec{\tau}$ for 50 steps to provide a better initialization point for the joint optimization with the other parameters.

(ii) $[0 \leq t_{opt} \leq 50]$ **Body Shape**: discourage unlikely body shapes by minimizing the KL-divergence of the shape PCA coefficients, $\vec{\beta}$ to the unit Gaussian. After each step, we compute the average shape and use this for the entire trajectory, effectively minimizing shape jittering.

(iii) $[50 \leq t_{opt} \leq 150]$ **3D Joints**: smooth-out the SMPL joints, by optimizing the global translation $\vec{\tau}$, orientation $\vec{\Gamma}$, and body pose $\vec{\theta}$ parameters for minimal first order difference in joint locations.

(iv) $[200 \leq t_{opt} \leq 250]$ **Body Pose**: discourage unlikely body poses by minimizing the KL-divergence of the body pose latents, $\vec{\theta_{\mathbf{enc}}}$ (i.e. $\vec{\theta}$ encoded with VPoser [42]), to unit Gaussian, as well as the temporal smoothness of the latents.

(v) $[250 \leq t_{opt} \leq 400]$ **Motion Prior**: discourage motion ”jumpiness”” by maximizing the likelihood of each SMPL transition in HuMoR [45] encoder. During this stage, we optimize body pose latents $\vec{\theta_{\mathbf{enc}}}$ (and therefore body pose), global pose $\vec{\Gamma}, \vec{\tau}$ and offset from camera. It is important to note that since HuMoR relies on an explicit ground to transform meshes to/from a normalized space, we observed that it lead to failure in approximately 41% of the scenes. SLAHMR reports similar findings. In case of failure (expressed with NaNs in the likelihood), we revert to the last successful step and end the optimization).

where t_{opt} denotes the step index of the optimization process. As described in [63], we opt not to optimize all desired parameters jointly as due to the intrinsic ambiguation the optimizer usually finds a short-path that yields non-meaningful human poses or

motions. Differently from SLAHMR, we use the Adam [26] optimizer with learning rate 0.05. An example output of global optimization is visualized in 4.8 (4th row).

4.2 Experimentation Datasets

HMR datasets typically contain both 2D and 3D annotations for each person in the input frames or scenes. These annotations can include image-based detections, ground-truth human attributes (e.g., gender, age), and 3D human body shape information captured via markers or other methods. Additionally, some datasets provide global mesh placements relative to a fixed world reference frame. While this information is beneficial for end-to-end training of local-to-global trajectory lifting networks, it is not strictly essential. Models can still learn to predict realistic motions within an implicitly defined global frame. Moreover, standard evaluation metrics involve a mesh alignment stage, making the explicit global ground truth data unnecessary.

For the experiments in this project, we focus on the widely-used monocular HMR dataset *3D Poses In-The-Wild (3DPW)*, which is described next.

4.2.1 3D Poses In-The-Wild (3DPW)

Introduced by Marcard et al. [57], the 3D Poses In-The-Wild (3DPW) dataset is a leading benchmark for evaluating human pose estimation algorithms in uncontrolled environments. This dataset features indoor and outdoor videos captured with a moving mobile phone, along with corresponding 2D annotations of human pose and shape. It is split in three sets, train containing, validation, and test, containing 25, 12, and 24 scenes respectively. The 3D poses are meticulously estimated through a process that optimizes SMPL body model alignment against 2D pose detections, enhanced by Inertial Measurement Unit (IMU) readings attached to human joints. A frame of 3DPW with the provided ground truth annotations is give in Figure 4.11.

3DPW presents a particularly demanding evaluation scenario. It comprises approximately 51K frames recorded at 30Hz, depicting common actions like walking, conversing, and stair climbing. The moving camera, natural clothing, and cluttered backgrounds introduce significant occlusions and complex motions, making it a robust test for monocular motion capture in dynamic settings, that is at the core of the current work. Therefore, we use exclusively 3DPW for evaluating our developed pipeline both

Figure 4.11: Sample frame from 3DPW (left) and visualization of 2D poses alongside camera-accurate, textured SMPL 3D meshes. Global placement is achieved indirectly via IMU-based camera localization.
Source: Refactored from *Recovering accurate 3d human pose in the wild using imus and a moving camera*, Marcard et al. [57]

in terms of camera-local and global mesh trajectory estimation. In the dataset's benchmark guidelines, it is allowed for methods to employ ground truth annotations of human genders and tracking IDs; we only use ground-truth genders as MiVOLO exhibited significant bias when inference was performed on 3DPW.

Chapter 5

Results

In this chapter the quantitative and qualitative results of applying the developed methodology to the datasets under consideration are given. We begin with a brief recap of the pipeline, and the proceed by describing the evaluation process. The chapter is completed by results in tabular and image forms as well as relative discussion.

5.1 Methodology in a Nutshell

Our approach to monocular human motion recovery (HMR) for videos shot with dynamic cameras leverages a multi-stage pipeline to deliver accurate 3D motion representations of individuals in complex environments. The detailed process is repeated below in summarized form:

- **Video Preprocessing**: The input video is first carefully preprocessed. Individual frames are extracted (optionally) padded to have square aspect ratios, to facilitate subsequent analysis.
- **Image-based Detection**: State-of-the-art 2D human attribute estimation techniques are employed to identify individuals and detect key joints and landmarks within each frame. Alongside pose, body attributes and additional features, such as genders and segmentation masks, are extracted. These 2D detections form the basis for subsequent 3D reconstruction.
- **Mesh Estimation**: We fit SMPL models to the detected 2D attributes. The SMPL model provides a parametric representation of human body shape and pose.

By optimizing SMPL parameters against the 2D detections, we obtain a local 3D representation of individuals in each frame, capturing their body shape and articulation relative to the camera.

- **Appearance-Aware Tracking**: To maintain consistent reconstructions across frames and mitigate challenges like occlusions or complex movement, we implement an appearance-aware tracking mechanism. This tracker associates detected individuals between frames, incorporating visual features beyond just box-based cues. This ensures robustness and temporal coherence in the 3D reconstructions, and was found to be a crucial aspect of the overall performance.
- **Local Track Creation**: From the sequence of estimated camera-local SMPL parameters, we derive 3D human motion tracks within the camera's coordinate system. These tracks depict the trajectory of each individual's motion relative to the camera over time. We infill missing observations caused by the detector and/or the tracker by interpolating on the present ones.
- **Global Trajectory Regression**: To recover the 3D trajectory of each subject within a world reference frame, we employ a regression based on WHAM that enables us to lift the camera-local trajectory to a global one using regression. Those are natural realistic global motions, though most times are over-smoothed and tend to deviate from visual observations.
- **Global Optimization**: The regressed global trajectory is further refined using a targeted global optimization strategy. This step enforces desired characteristics like smoothness, physical plausibility, and re-projection consistency gleaned from the video. Here, we follow the optimization recipe derived from SLAHMR, ensuring a refined trajectory that adheres to realistic motion patterns.

5.2 Evaluation Procedure

To evaluate our method's performance, we employ a suite of standard metrics used for motion recovery analysis. These metrics directly compare regressed 3D meshes with their ground-truth counterparts at the vertex and skeleton levels. In what follows, we define the used metrics, and subsequently, outline the process of aligning and matching ground-truth and generated tracks to ensure accurate and fair comparison. Followingly, we present our pipeline's performance on the 3DPW dataset both on per-

scene basis and the average metrics across each subset of the dataset.

5.2.1 Evaluation Metrics

Mean Per-Joint Position Error (MPJPE) is a fundamental metric that measures the average euclidean distance (mm) between predicted and ground-truth joint positions. It is perhaps the most common evaluation metric for 3D human pose estimation. MPJPE is calculated after the alignment of the root joint (typically the pelvis) between the ground-truth and predicted body poses. Equation 5.1 shows the MPJPE calculation for one frame after root joint alignment:

$$E_{MPJPE} = \frac{1}{N_J} \sum_{i=1}^{N_J} \left\| J_i^{gt} - J_i^{pred} \right\|^2 \tag{5.1}$$

where N_J is the number of joints, J_i^{gt} is the 3D location of the i-th ground-truth joint and J_i^{pred} is that of the joint predicted by running SMPL with the estimated parameters. The number of SMPL joints used to compute joint-related metrics in this project is $N_J = 14$ (see Figure 4.5 right) following MPII benchmark [1].

Procrustes Aligned Mean Per-Joint Position Error (PA-MPJPE) PA-MPJPE accounts for potential rigid transformations between the prediction and ground truth by applying Procrustes Analysis (PA) before calculating the error. Procrustes analysis estimates the ”best” similarity transformation (translation, scaling, and rotation) so the projection of one mesh to the other results in minimized vertex offsets [16]. This isolates pose-specific errors from misalignment in mesh pose or scale. As such, it is the de facto metric used to benchmark HMR methods.

Mean Per-Vertex Error (PVE) PVE calculates the average position error (mm) of the vertices of the generated SMPL mesh and the ground truth ones, after root joint alignment. As it operates on the generated mesh vertices, PVE captures the fit fidelity of both the human body and shape. Equation 5.2 shows the MPVE calculation for one frame after root joint alignment:

$$E_{PVE} = \frac{1}{N_V} \sum_{i=1}^{N_V} \left\| V_i^{gt} - V_i^{pred} \right\|^2 \tag{5.2}$$

where N_V is the number of joints, V_i^{gt} is the 3D location of the i-th ground-truth joint and V_i^{pred} is that of the vertices predicted by running SMPL with the estimated parameters. The number of vertices regressed in the SMPL model that we use in this project is 6980, each having xyz coordinates.

Acceleration Error Acceleration error, introduced by Kanazawa et al. in [23], is a metric for evaluating the temporal smoothness (m/s²) of the estimated pose sequence. Acceleration vectors are computed for the 3D joint sequence, and the acceleration error is calculated as the average difference between the estimated and ground-truth acceleration vectors. Equations 5.3 show the per-joint and the final accelaration error calculation after root joint alignment:

$$\begin{aligned} A_j^{gt} &= J_{j,1:T_p-2}^{gt} - 2 \times J_{j,2:T_p-1}^{gt} + J_{j,3:T_p}^{gt} \\ A_j^{pred} &= J_{j,1:T_p-2}^{pred} - 2 \times J_{j,2:T_p-1}^{pred} + J_{j,3:T_p}^{pred} \\ Accel &= \frac{1}{N_J} \sum_{i=1}^{N_J} \left\| A_i^{gt} - A_i^{pred} \right\|^2 \times fps^2 \end{aligned} \tag{5.3}$$

where T is the total timesteps (number of frames) and fps are the frames per second used while recording the video. As we focus on 3DPW, $fps = 30$ throughout this work, and T_p is the number of frames from the first up to and including the last occurrence of person on the video (as derived via tracking). Note that a total of $T - 2$ timesteps are used to compute acceleration error, as Equation 5.3 is the second order central difference on discretized mesh samples at time offsets equal to $1/fps = 1/30sec.$

5.2.2 Track Alignment

Ground tracks usually contain at least as many time steps as the ones found in the estimated tracks. Therefore, in order to correctly evaluate an inferred track an alignment step is necessary. During this step, both tracks are trimmed to minimum common set of time steps, which usually requires disregarding ground-truth measurements before the inferred track's first and after last visible frame. We remind the reader, that intermediate steps are already present for both track under consideration at this point, as they are either present (ground truth), inferred, or infilled during the local track creation stage.

5.2.3 Bipartite Matching

Given that the assigned IDs by the tracker may differ from the ones given by the dataset authors as ground-truths we cannot rely on them to associate tracks. There have been a number of ways to achieve this association problem; we opt for a simplified bipartite matching algorithm. In particular, we compute the evaluation metrics for every possible pair of original-estimated tracks, we rank those relative to the ground truth track ID (i.e. for every such ID we list the metrics with respect to all inferred tracks), and then select the pairs that minimize the average PA-MPJPE metric of the association. This process is heavily inspired by SLAHMR, but we allow for more flexibility by matching based on the average metric across all ground-truth tracks. For this, we employ the Munkres algorithm[1] [40].

5.2.4 Fair Comparison

To ensure fairness, we do not use pre-computed meshes when comparing tracks, e.g. ones that may have been generated during the optimization process. Instead, we use the final sequence of SMPL parameters for each track, $\left\{\vec{\boldsymbol{\theta}}_t, \vec{\boldsymbol{\beta}}_t, \vec{\boldsymbol{\Gamma}}_t, \vec{\boldsymbol{\tau}}_t\right\}_{t=t_s^{(p)}}^{t_e^{(p)}}$, to generate the corresponding sequence of mesh vertices and 3D joints used for comparison. Via this way, comparison is actually done for the same set of mesh parameters while meshed bodies are derived in identical way.

5.3 Results on 3DPW

We are now able to present the results of applying our developed pipeline on the 3DPW dataset. In Table 5.1 we list the aggregated metrics throughout the entire dataset consisting of the train, validation, and test subsets. We provide the computed joint and vertex errors during the three stages of our pipeline: camera-local, based on HMR2.0 [15], global using regression, based on WHAM [50], and global using regression and optimization with the latter being based on SLAHMR [63]. As can be seen in Table 5.1, WHAM achieves the best pose reconstruction error (Procrustes-aligned MPJPE), both in its original form and in our re-implementation. Our findings suggest that using it for global trajectory regression results in naturally smooth motions especially

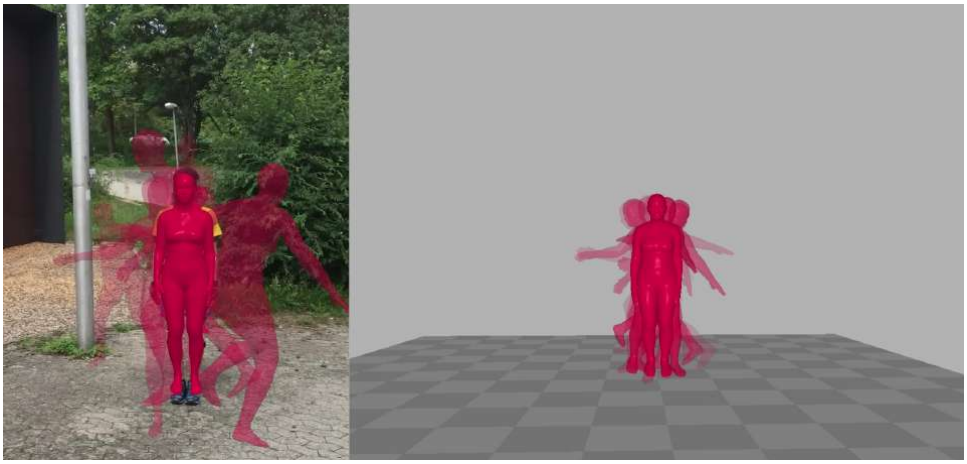
[1]The Munkres (also known as the Hungarian) algorithm finds the minimum-cost assignment between workers and jobs in a cost matrix by iteratively manipulating rows and columns, identifying zero-cost assignments, and adjusting the matrix until an optimal solution is reached.

Models	3DPW			
	PA-MPJPE	MPJPE	PVE	Acc
HMR2.0 [15] *	44.4	69.8	82.2	18.1
WHAM [50] *	37.2	59.4	71.0	6.9
SLAHMR [63] *	55.9	-	-	-
Local	48.1	70.2	90.8	17.8
Global Regr	**43.1**	74.5	101.1	**7.2**
Global Opt	49.7	**59.9**	**74.4**	9.1
Global Regr + Opt	43.4	64.1	74.3	7.6

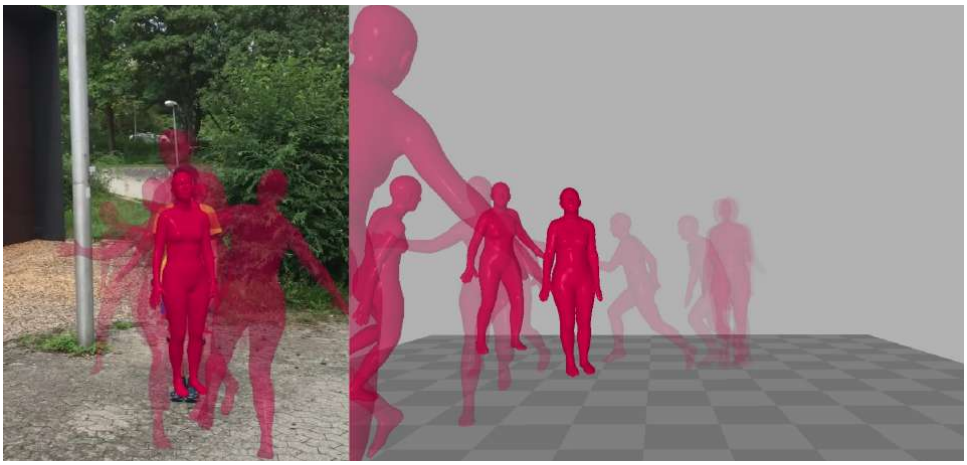
Table 5.1: Global motion estimation metrics on 3DPW [57] aggregated across all subsets, i.e. considering all dataset scenes. The metrics denoted with * are taken from the original papers and are given here for reference.

when feet are in touch with the ground, as was mostly the case with AMASS [39], its training dataset. But overall, our global optimization recipe, based on SLAHMR, resulted in better alignment with the ground truth in terms of global mesh orientations, and therefore yielded the lowest errors without using PA. Finally, and most importantly, chaining regression with optimization for global trajectory lifting showcased viable results, exhibiting joint reconstruction error on par with that of regression method but non-PA errors on par with optimization ones. This signifies that combining regression with optimization for local to global motion lifting, is meaningful and effective, aligning with similar findings in literature regarding camera-local motion regression [28].

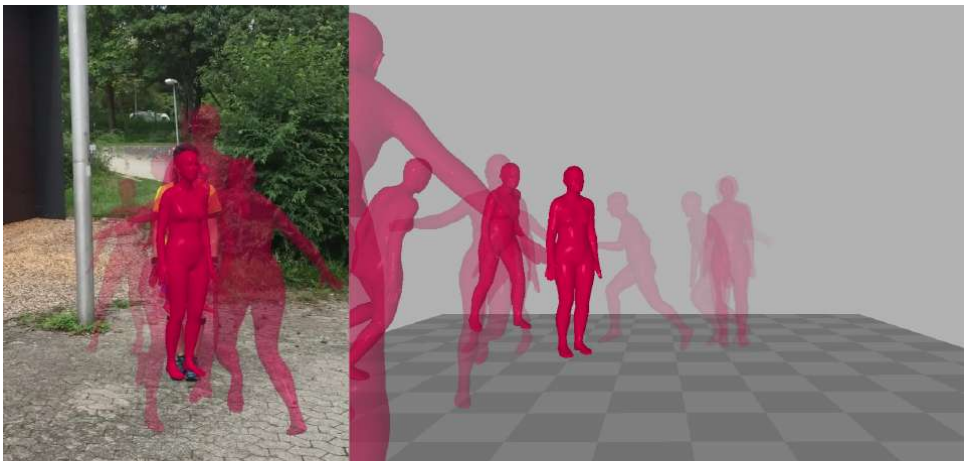
In Table 5.2 we list the (per-)joint reconstruction errors for each scene of 3DPW separately, while also noting the number of humans that are in focus in each scene (or equivalently for whom there are ground truth data). One can readily notice the increased errors on the test compared to the one train split, which mainly has to do with the increased number of humans appearing as well as the overall complexity of the scenes in the former. In addition, it is apparent that ”downtown” scenes containing, tend to have higher MPJPE than ”courtyard” ones. That should be attributed towards increased cases of occlusion and more complex interactions happening on the former, that make the tracker and contact estimator to underperform. Finally, scenes with wide depth range (such as `downtown_sitOnStairs`) often lead to poor camera-local reconstruction error, as local HMR models struggle to disambiguate depth and height or shape [51].



(a) Local

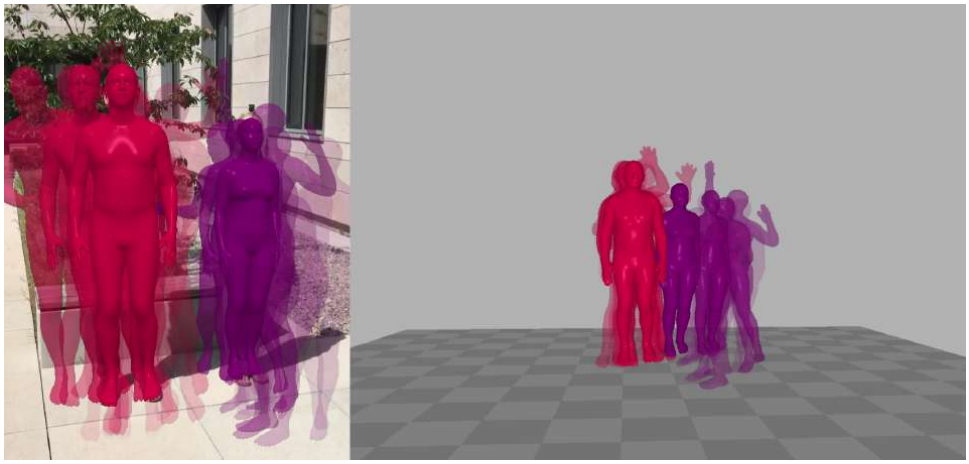


(b) Global Regresion

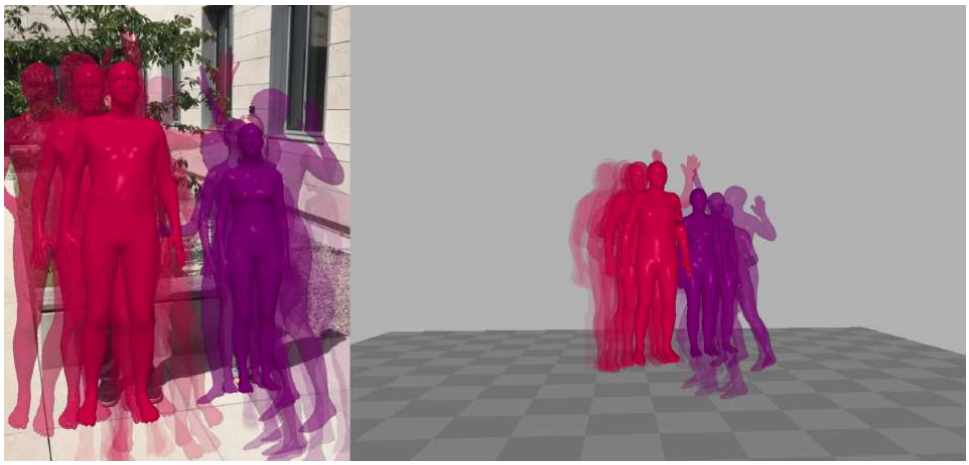


(c) Global Regresion + Optimization

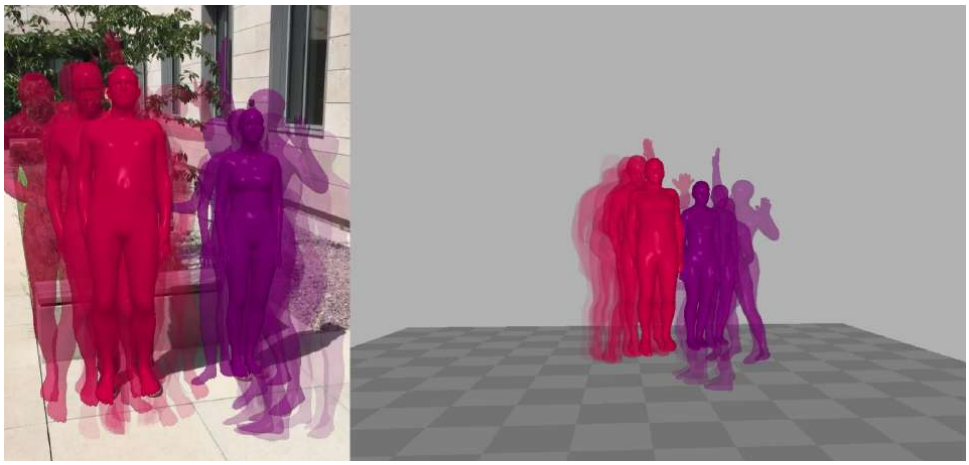
Figure 5.1: Visualization of the recovered human motion from a a scene of 3DPW [57] training set, `outdoor_slalom`. Note that camera-local estimation from HMR2.0 [15] (a) results in consistent re-projections yet quite limited motions in 3D, whereas the globally-lifted tracks have more diverse motions (depicted as mesh locations). Global regression using WHAM [50] (b) results in smooth motions sacrificing consistency with visual cues. Further optimization of the global trajectory (c) results in better projection consistency and smoother mesh translations relative to the world frame. The backgrounds corresponding to the last frame are used, while transparency indicates track age. Frames are sampled exponentially.



(a) Local



(b) Global Regresion



(c) Global Regresion + Optimization

Figure 5.2: Visualization of the recovered human motion from a a scene of 3DPW [57] training set, `courtyard_goodNews`. Note that camera-local estimation from HMR2.0 [15] (a) results in consistent re-projections yet quite limited motions in 3D, whereas the globally-lifted tracks have more diverse motions (depicted as mesh locations). Global regression using WHAM [50] (b) results in smooth motions sacrificing consistency with visual cues. Further optimization of the global trajectory (c) results in better projection consistency and smoother mesh translations relative to the world frame. The backgrounds corresponding to the last frame are used, while transparency indicates track age. Frames are sampled exponentially.

	Local		Global Regr.		Global Opt.		Global Regr. + Opt.	
Scenes	PA-MPJPE	MPJPE	PA-MPJPE	MPJPE	PA-MPJPE	MPJPE	PA-MPJPE	MPJPE
Train Set								
courtyard_arguing[2]	24.49	41.57	15.49	34.30	19.08	25.09	18.89	23.90
courtyard_backpack[1]	30.09	50.76	17.00	44.75	21.47	31.94	19.43	24.46
courtyard_basketball[2]	38.65	72.12	31.27	85.10	38.84	61.35	35.46	50.28
courtyard_bodyScannerMotions[1]	96.01	110.28	74.77	103.65	92.95	74.57	70.04	57.07
courtyard_box[1]	28.86	64.04	22.80	63.07	28.66	45.03	31.27	39.60
courtyard_capoeira[2]	37.66	80.47	26.75	75.88	32.95	54.50	28.41	43.10
courtyard_captureSelfies[2]	66.24	92.33	89.95	107.25	112.02	77.99	85.14	101.27
courtyard_giveDirections[2]	22.28	45.19	12.00	36.18	15.40	25.25	16.88	25.88
courtyard_golf[1]	29.13	56.28	19.10	52.20	23.61	37.59	27.53	30.26
courtyard_goodNews[2]	23.97	46.48	13.74	31.23	17.88	22.72	14.26	23.26
courtyard_jacket[1]	74.53	95.19	55.53	103.50	69.47	74.56	57.44	56.10
courtyard_laceShoe[1]	96.32	98.40	71.96	87.61	90.27	63.94	72.22	88.27
courtyard_rangeOfMotions[2]	35.47	46.53	31.82	53.01	40.55	38.07	32.67	32.26
courtyard_relaxOnBench[1]	28.91	56.77	15.03	97.55	19.02	70.46	19.73	42.85
courtyard_shakeHands[2]	28.88	55.55	15.41	81.45	20.36	59.40	21.15	45.57
courtyard_warmWelcome[1]	36.58	62.76	39.27	83.08	48.64	59.86	44.08	51.49
outdoors_climbing[1]	26.92	36.51	21.38	123.74	26.47	89.56	25.54	56.47
outdoors_freestyle[1]	25.97	45.58	16.05	29.28	20.61	20.39	20.02	16.99
outdoors_slalom[1]	31.22	37.59	20.59	59.24	25.15	42.46	24.47	24.30
Test Set								
downtown_arguing[2]	29.39	41.17	28.23	42.35	32.86	33.31	27.65	47.48
downtown_bar[2]	52.07	56.10	70.33	101.44	80.13	78.72	69.73	98.91
downtown_bus[2]	54.28	92.14	66.02	89.25	75.34	70.22	66.50	95.84
downtown_cafe[2]	75.82	99.54	60.72	88.39	68.77	69.40	57.79	87.37
downtown_car[2]	60.86	72.62	61.86	100.68	70.70	78.61	62.34	99.63
downtown_crossStreets[2]	56.28	60.69	53.59	84.71	61.21	65.82	51.02	78.86
downtown_downstairs[1]	46.75	82.11	63.32	126.02	72.34	98.23	65.17	129.77
downtown_enterShop[1]	68.22	101.18	61.10	118.85	69.81	91.86	61.47	115.71
downtown_rampAndStairs[2]	29.91	83.66	26.54	59.31	30.74	46.69	25.45	69.66
downtown_runForBus[2]	61.98	81.88	54.11	76.77	61.62	59.92	54.31	77.43
downtown_sitOnStairs[2]	107.25	123.98	55.88	106.25	63.93	83.16	55.01	110.62
downtown_stairs[1]	66.21	101.00	63.61	70.24	72.41	54.54	63.02	68.21
downtown_upstairs[1]	23.87	46.34	28.92	20.53	33.42	16.18	30.29	23.62
downtown_walking[2]	36.32	36.13	33.60	48.09	38.12	37.62	32.51	47.76
downtown_walkUphill[1]	57.15	87.29	52.54	108.22	60.58	83.96	54.87	109.07
downtown_warmWelcome[2]	42.54	103.69	38.13	64.33	43.56	49.98	37.84	69.20
downtown_weeklyMarket[1]	27.90	27.50	31.79	63.73	36.37	49.64	30.81	36.93
downtown_windowShopping[1]	29.73	40.95	30.92	76.31	35.66	58.76	24.53	51.77
flat_packBags[1]	51.61	66.68	26.92	61.84	31.76	48.39	29.91	64.85
office_phoneCall[2]	61.03	106.69	60.29	102.87	68.94	79.46	62.20	107.39

Table 5.2: Joint errors (mm) on 3DPW [57] train and test sets. Superscripts denote number of humans in focus on input videos. Grayscale heatmap has been applied.

Chapter 6

Conclusions

6.1 Discussion

This project explored various methods for recovering human motion from monocular cues. We began with image-based detection, followed by camera-local mesh and motion estimation. Qualitative results, presented in Figures 4.8, 5.2, and 5.1, highlight the necessity of lifting local tracks to a fixed global frame. Additionally, these results demonstrate the powerful combination of regression-based and hand-crafted optimization approaches for achieving this goal. Regression enables 3D world reconstruction of human motion, while hand-crafted global optimization reduces non-PA reconstruction errors by minimizing re-projection and motion criteria.

We extensively evaluated our pipeline against standard HMR metrics, emphasizing mesh vertex and skeletal joint reconstruction. Table 5.1 demonstrates that our implementation performs comparably to, or even surpasses, state-of-the-art methods on the 3DPW dataset. Importantly, this highlights the effectiveness of adapting hybrid approaches from camera-local literature to global motion recovery. Specifically, when global optimization is applied on top of global regression, the recovered motions exhibit superior performance characteristics.

6.2 Future Work

This work offers several avenues for improvement and extension. Firstly, applying the pipeline to larger and more recent datasets like RICH [21] and EMBD [24]

could provide deeper insights into its effectiveness and limitations. Additionally, more comprehensive ablation tests should be conducted to evaluate the individual contributions of each pipeline node. For example, while DPVO was used for visual odometry, the qualitative impact of our interventions remains to be measured. Finally, an exciting direction would be to develop an end-to-end local-to-global lifting network. This network could incorporate hand-crafted optimization objectives as supervision signals or regularization terms during training, potentially leading to further performance gains.

Bibliography

[1] Andriluka, Mykhaylo, Pishchulin, Leonid, Gehler, Peter, and Schiele, Bernt. “2d human pose estimation: New benchmark and state of the art analysis”. In: *Proceedings of the IEEE Conference on computer Vision and Pattern Recognition*. 2014, pp. 3686–3693.

[2] Autodesk, INC. *Maya*. Version 2019. Jan. 15, 2019. URL: `https://%20autodesk.com/maya`.

[3] Ba, Jimmy Lei, Kiros, Jamie Ryan, and Hinton, Geoffrey E. “Layer normalization”. In: *arXiv preprint arXiv:1607.06450* (2016).

[4] Bewley, Alex, Ge, Zongyuan, Ott, Lionel, Ramos, Fabio, and Upcroft, Ben. “Simple online and realtime tracking”. In: *2016 IEEE international conference on image processing (ICIP)*. IEEE. 2016, pp. 3464–3468.

[5] Bogo, Federica, Kanazawa, Angjoo, Lassner, Christoph, Gehler, Peter, Romero, Javier, and Black, Michael J. “Keep it SMPL: Automatic Estimation of 3D Human Pose and Shape from a Single Image”. In: *Computer Vision – ECCV 2016*. Lecture Notes in Computer Science. Springer International Publishing, Oct. 2016.

[6] Brown, Duane. “The bundle adjustment-progress and prospect”. In: *XIII Congress of the ISPRS, Helsinki, 1976*. 1976.

[7] Carion, Nicolas, Massa, Francisco, Synnaeve, Gabriel, Usunier, Nicolas, Kirillov, Alexander, and Zagoruyko, Sergey. “End-to-end object detection with transformers”. In: *European conference on computer vision*. Springer. 2020, pp. 213–229.

[8] Chen, Kai, Wang, Jiaqi, Pang, Jiangmiao, Cao, Yuhang, Xiong, Yu, Li, Xiaoxiao, Sun, Shuyang, Feng, Wansen, Liu, Ziwei, Xu, Jiarui, et al. “MMDetection: Open mmlab detection toolbox and benchmark”. In: *arXiv preprint arXiv:1906.07155* (2019).

[9] Community, Blender Online. *Blender - a 3D modelling and rendering package*. Blender Foundation. Stichting Blender Foundation, Amsterdam, 2018. URL: http://www.blender.org.

[10] Deng, Boyang, Lewis, John P, Jeruzalski, Timothy, Pons-Moll, Gerard, Hinton, Geoffrey, Norouzi, Mohammad, and Tagliasacchi, Andrea. "Nasa neural articulated shape approximation". In: *European Conference on Computer Vision*. Springer. 2020, pp. 612–628.

[11] Ellson, John, Gansner, Emden, Koutsofios, Lefteris, North, Stephen C, and Woodhull, Gordon. "Graphviz—open source graph drawing tools". In: *Graph Drawing: 9th International Symposium, GD 2001 Vienna, Austria, September 23–26, 2001 Revised Papers 9*. Springer. 2002, pp. 483–484.

[12] Ester, Martin, Kriegel, Hans-Peter, Sander, Jörg, Xu, Xiaowei, et al. "A density-based algorithm for discovering clusters in large spatial databases with noise". In: *kdd*. Vol. 96. 34. 1996, pp. 226–231.

[13] Evans, Clark, Net, Ingy döt, and Ben-Kiki, Oren. *The official YAML web site*. 2001. URL: https://yaml.org/.

[14] Girshick, Ross. "Fast r-cnn". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1440–1448.

[15] Goel, Shubham, Pavlakos, Georgios, Rajasegaran, Jathushan, Kanazawa, Angjoo, and Malik, Jitendra. "Humans in 4D: Reconstructing and Tracking Humans with Transformers". In: *arXiv preprint arXiv:2305.20091* (2023).

[16] Gower, John C. "Generalized procrustes analysis". In: *Psychometrika* 40 (1975), pp. 33–51.

[17] Hartley, Richard and Zisserman, Andrew. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[18] He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[19] Hirshberg, David A, Loper, Matthew, Rachlin, Eric, and Black, Michael J. "Coregistration: Simultaneous alignment and modeling of articulated 3D shape". In: *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision,*

Florence, Italy, October 7-13, 2012, Proceedings, Part VI 12. Springer. 2012, pp. 242–255.

[20] Hochreiter, Sepp and Schmidhuber, Jürgen. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[21] Huang, Chun-Hao P., Yi, Hongwei, Höschle, Markus, Safroshkin, Matvey, Alexiadis, Tsvetelina, Polikovsky, Senya, Scharstein, Daniel, and Black, Michael J. “Capturing and Inferring Dense Full-Body Human-Scene Contact”. In: *Proceedings IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*. June 2022, pp. 13274–13285.

[22] Imambi, Sagar, Prakash, Kolla Bhanu, and Kanagachidambaresan, GR. “PyTorch”. In: *Programming with TensorFlow: Solution for Edge Computing Applications* (2021), pp. 87–104.

[23] Kanazawa, Angjoo, Zhang, Jason Y, Felsen, Panna, and Malik, Jitendra. “Learning 3d human dynamics from video”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 5614–5623.

[24] Kaufmann, Manuel, Song, Jie, Guo, Chen, Shen, Kaiyue, Jiang, Tianjian, Tang, Chengcheng, Zárate, Juan José, and Hilliges, Otmar. “EMDB: The Electromagnetic Database of Global 3D Human Pose and Shape in the Wild”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2023, pp. 14632–14643.

[25] Ke, Bingxin, Obukhov, Anton, Huang, Shengyu, Metzger, Nando, Daudt, Rodrigo Caye, and Schindler, Konrad. “Repurposing Diffusion-Based Image Generators for Monocular Depth Estimation”. In: *arXiv preprint arXiv:2312.02145* (2023).

[26] Kingma, Diederik P and Ba, Jimmy. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[27] Kocabas, Muhammed, Yuan, Ye, Molchanov, Pavlo, Guo, Yunrong, Black, Michael J, Hilliges, Otmar, Kautz, Jan, and Iqbal, Umar. “PACE: Human and Camera Motion Estimation from in-the-wild Videos”. In: *arXiv preprint arXiv:2310.13768* (2023).

[28] Kolotouros, Nikos, Pavlakos, Georgios, Black, Michael J, and Daniilidis, Kostas. “Learning to reconstruct 3D human pose and shape via model-fitting in the loop”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 2252–2261.

[29] Kolotouros, Nikos, Pavlakos, Georgios, and Daniilidis, Kostas. “Convolutional mesh regression for single-image human shape reconstruction”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4501–4510.

[30] Kolotouros, Nikos, Pavlakos, Georgios, and Daniilidis, Kostas. “Convolutional mesh regression for single-image human shape reconstruction”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4501–4510.

[31] Kuprashevich, Maksim and Tolstykh, Irina. “Mivolo: Multi-input transformer for age and gender estimation”. In: *arXiv preprint arXiv:2307.04616* (2023).

[32] Li, Jiefeng, Xu, Chao, Chen, Zhicun, Bian, Siyuan, Yang, Lixin, and Lu, Cewu. “HybrIK: A Hybrid Analytical-Neural Inverse Kinematics Solution for 3D Human Pose and Shape Estimation”. en. In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Nashville, TN, USA: IEEE, June 2021, pp. 3382–3392. ISBN: 978-1-66544-509-2. DOI: 10.1109/CVPR46437.2021.00339. URL: https://ieeexplore.ieee.org/document/9577652/ (visited on 06/20/2023).

[33] Li, Zhihao, Liu, Jianzhuang, Zhang, Zhensong, Xu, Songcen, and Yan, Youliang. “Cliff: Carrying location information in full frames into human pose and shape estimation”. In: *European Conference on Computer Vision*. Springer. 2022, pp. 590–606.

[34] Lin, Kevin, Wang, Lijuan, and Liu, Zicheng. “End-to-end human pose and mesh reconstruction with transformers”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 1954–1963.

[35] Lin, Tsung-Yi, Maire, Michael, Belongie, Serge, Hays, James, Perona, Pietro, Ramanan, Deva, Dollár, Piotr, and Zitnick, C Lawrence. “Microsoft coco: Common objects in context”. In: *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer. 2014, pp. 740–755.

[36] Liu, Ze, Lin, Yutong, Cao, Yue, Hu, Han, Wei, Yixuan, Zhang, Zheng, Lin, Stephen, and Guo, Baining. “Swin transformer: Hierarchical vision transformer using shifted windows”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 10012–10022.

[37] Liu, Zhuang, Mao, Hanzi, Wu, Chao-Yuan, Feichtenhofer, Christoph, Darrell, Trevor, and Xie, Saining. “A convnet for the 2020s”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 11976–11986.

[38] Loper, Matthew, Mahmood, Naureen, Romero, Javier, Pons-Moll, Gerard, and Black, Michael J. “SMPL: A Skinned Multi-Person Linear Model”. In: *ACM Trans. Graphics (Proc. SIGGRAPH Asia)* 34.6 (Oct. 2015), 248:1–248:16.

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

[40] Munkres, James. “Algorithms for the assignment and transportation problems”. In: *Journal of the society for industrial and applied mathematics* 5.1 (1957), pp. 32–38.

[41] Osman, Ahmed AA, Bolkart, Timo, and Black, Michael J. “Star: Sparse trained articulated human body regressor”. In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VI 16*. Springer. 2020, pp. 598–613.

[42] Pavlakos, Georgios, Choutas, Vasileios, Ghorbani, Nima, Bolkart, Timo, Osman, Ahmed A. A., Tzionas, Dimitrios, and Black, Michael J. “Expressive Body Capture: 3D Hands, Face, and Body from a Single Image”. In: *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[43] Rajasegaran, Jathushan, Pavlakos, Georgios, Kanazawa, Angjoo, and Malik, Jitendra. “Tracking people by predicting 3D appearance, location and pose”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 2740–2749.

[44] Ravi, Nikhila, Reizenstein, Jeremy, Novotny, David, Gordon, Taylor, Lo, Wan-Yen, Johnson, Justin, and Gkioxari, Georgia. “Accelerating 3d deep learning with pytorch3d”. In: *arXiv preprint arXiv:2007.08501* (2020).

[45] Rempe, Davis, Birdal, Tolga, Hertzmann, Aaron, Yang, Jimei, Sridhar, Srinath, and Guibas, Leonidas J. “Humor: 3d human motion model for robust pose estimation”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 11488–11499.

[46] Rodrigues, Olinde. “Des lois géométriques qui régissent les déplacements d’un système solide dans l’espace, et de la variation des coordonnées provenant de ces déplacements considérés indépendamment des causes qui peuvent les produire”. In: *Journal de mathématiques pures et appliquées* 5 (1840), pp. 380–440.

[47] Rombach, Robin, Blattmann, Andreas, Lorenz, Dominik, Esser, Patrick, and Ommer, Björn. “High-resolution image synthesis with latent diffusion models. 2022 IEEE”. In: *CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 10674–10685.

[48] Romero, Javier, Tzionas, Dimitrios, and Black, Michael J. “Embodied hands: Modeling and capturing hands and bodies together”. In: *arXiv preprint arXiv:2201.02610* (2022).

[49] Saini, Nitin, Price, Eric, Tallamraju, Rahul, Enficiaud, Raffi, Ludwig, Roman, Martinovic, Igor, Ahmad, Aamir, and Black, Michael J. “Markerless outdoor human motion capture using multiple autonomous micro aerial vehicles”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 823–832.

[50] Shin, Soyong, Kim, Juyong, Halilaj, Eni, and Black, Michael J. “WHAM: Reconstructing World-grounded Humans with Accurate 3D Motion”. In: *arXiv preprint arXiv:2312.07531* (2023).

[51] Sun, Yu, Liu, Wu, Bao, Qian, Fu, Yili, Mei, Tao, and Black, Michael J. “Putting People in their Place: Monocular Regression of 3D People in Depth”. In: *CVPR*. 2022.

[52] Tallamraju, Rahul, Saini, Nitin, Bonetto, Elia, Pabst, Michael, Liu, Yu Tang, Black, Michael J, and Ahmad, Aamir. “AirCapRL: autonomous aerial human motion capture using deep reinforcement learning”. In: *IEEE Robotics and Automation Letters* 5.4 (2020), pp. 6678–6685.

[53] Teed, Zachary and Deng, Jia. “Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras”. In: *Advances in neural information processing systems* 34 (2021), pp. 16558–16569.

[54] Teed, Zachary, Lipson, Lahav, and Deng, Jia. “Deep patch visual odometry”. In: *Advances in Neural Information Processing Systems* 36 (2024).

[55] Tripathi, Shashank, Müller, Lea, Huang, Chun-Hao P., Omid, Taheri, Black, Michael J., and Tzionas, Dimitrios. “3D Human Pose Estimation via Intuitive Physics”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023, pp. 4713–4725. URL: https://ipman.is.tue.mpg.de.

[56] Vaswani, Ashish, Shazeer, Noam, Parmar, Niki, Uszkoreit, Jakob, Jones, Llion, Gomez, Aidan N, Kaiser, Łukasz, and Polosukhin, Illia. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).

[57] Von Marcard, Timo, Henschel, Roberto, Black, Michael J, Rosenhahn, Bodo, and Pons-Moll, Gerard. “Recovering accurate 3d human pose in the wild using imus and a moving camera”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 601–617.

[58] Wang, Jingdong, Sun, Ke, Cheng, Tianheng, Jiang, Borui, Deng, Chaorui, Zhao, Yang, Liu, Dong, Mu, Yadong, Tan, Mingkui, Wang, Xinggang, et al. “Deep high-resolution representation learning for visual recognition”. In: *IEEE transactions on pattern analysis and machine intelligence* 43.10 (2020), pp. 3349–3364.

[59] Wojke, Nicolai, Bewley, Alex, and Paulus, Dietrich. “Simple online and realtime tracking with a deep association metric”. In: *2017 IEEE international conference on image processing (ICIP)*. IEEE. 2017, pp. 3645–3649.

[60] Woo, Sanghyun, Debnath, Shoubhik, Hu, Ronghang, Chen, Xinlei, Liu, Zhuang, Kweon, In So, and Xie, Saining. “Convnext v2: Co-designing and scaling convnets with masked autoencoders”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 16133–16142.

[61] Xu, Hongyi, Bazavan, Eduard Gabriel, Zanfir, Andrei, Freeman, William T, Sukthankar, Rahul, and Sminchisescu, Cristian. “Ghum & ghuml: Generative 3d human shape and articulated pose models”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 6184–6193.

[62] Xu, Yufei, Zhang, Jing, Zhang, Qiming, and Tao, Dacheng. “Vitpose: Simple vision transformer baselines for human pose estimation”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 38571–38584.

[63] Ye, Vickie, Pavlakos, Georgios, Malik, Jitendra, and Kanazawa, Angjoo. “Decoupling human and camera motion from videos in the wild”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 21222–21232.

[64] Yuan, L, Hou, Q, Jiang, Z, Feng, J, and Yan, S. “Volo: Vision outlooker for visual recognition. arXiv”. In: *arXiv preprint arXiv:2106.13112* (2021).

[65] Yuan, Ye, Iqbal, Umar, Molchanov, Pavlo, Kitani, Kris, and Kautz, Jan. “GLAMR: Global occlusion-aware human mesh recovery with dynamic cameras”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 11038–11049.

[66] Zhu, Wentao, Ma, Xiaoxuan, Liu, Zhaoyang, Liu, Libin, Wu, Wayne, and Wang, Yizhou. “Motionbert: A unified perspective on learning human motion representations”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 15085–15099.

[67] Zong, Zhuofan, Song, Guanglu, and Liu, Yu. “Detrs with collaborative hybrid assignments training”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2023, pp. 6748–6758.

Appendix - Contents

A Execution Framework **76**

B Code Structure **78**

Appendix A

Execution Framework

To enable efficient single-GPU inference of multiples compute-heavy models and high-definition inputs, the developed system employs ETL constructs and filesystem-based caching mechanisms. Every node's output is cached in disk and retrieved before execution, thus enabling atomic filesystem-based caching mechanism and fault recovery. In addition, when a node is executed its corresponding modules are allocated before and de-allocated after so as to free up GPU memory. This paradigm, though decreasing throughput, ensures that only one visual learner resides at the GPU at a given time, thus enabling singe-GPU inference while using multiple memory-hungry models.

As the number of nodes and modules grow the need for a system-wide configuration becomes apparent; our execution graph, consisting of nodes that implement the `IPipelineNode` interface's `forward` and `visualize` methods (among others), is configured in YAML format [13], to enable flexibility and ease of use. At runtime, the nodes are initialized from a developed `ConfigReader`, that it is in-turn initialized from a pipeline configuration file. Therein, the class name of each node is resolved by recursively searching all the files inside `src` directory of the project as well as classes defined by PyTorch [22] library. If the node contains modules that allocated GPU memory then a `ConfigConcrete` instance is created instead, that handles the (de-)allocation logic of the enfolded modules. Model checkpoints are transformed and loaded automatically; in practice the model weights are extracted from each checkpoint and then re-associated based on shape and module name matching relative to current model state dictionary.

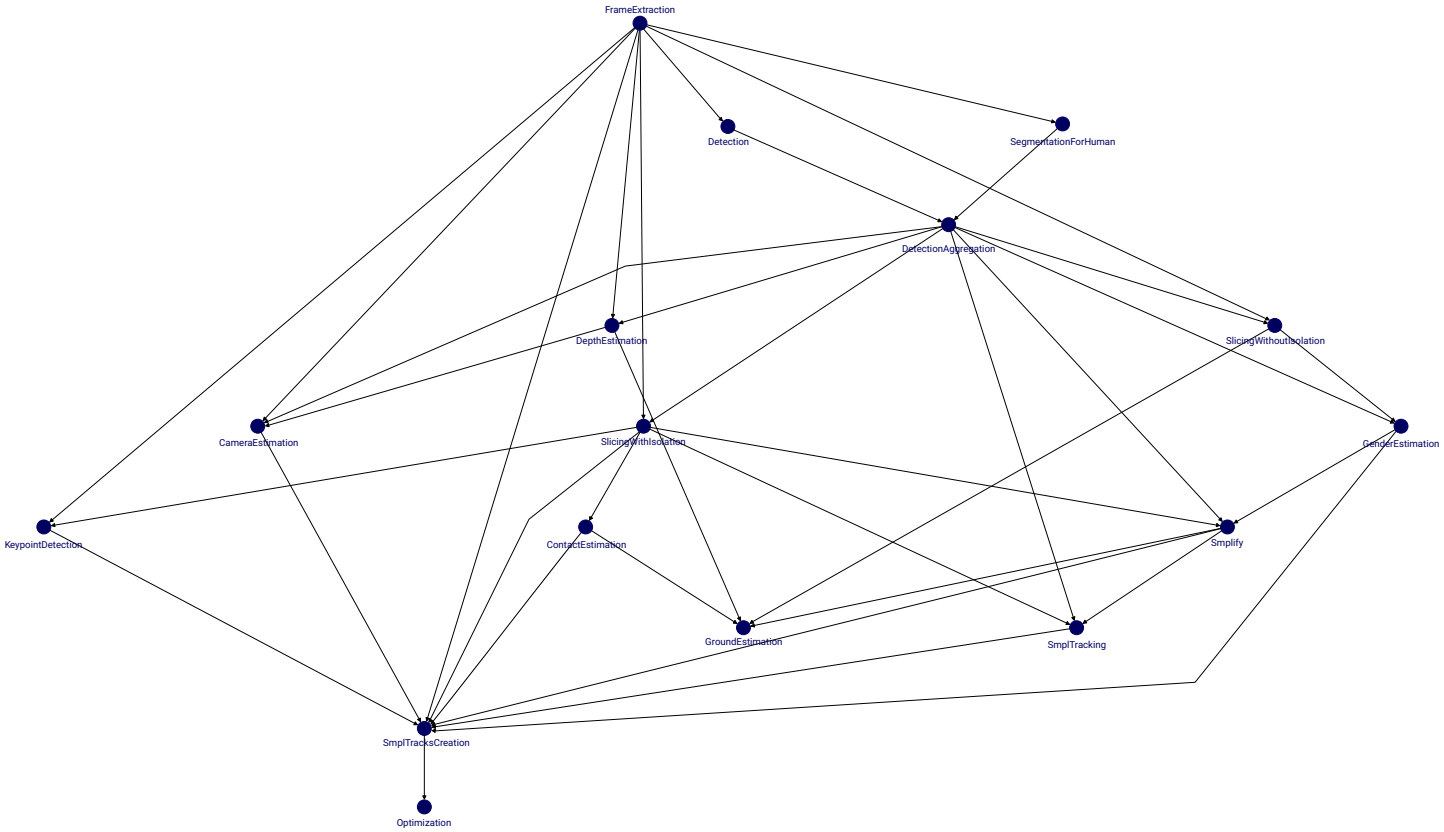


Figure A.1: Full pipeline graph visualized directly from the code using the `graphviz` library [11]. Class names of the defined work nodes are shown in navy blue, while dependencies are given as arrows. Note that the execution framework is essentially based in directed acyclic graph (DAG). This figure is best seen in zoom.

Since multiple nodes require access to the SMPL PyTorch module for generating the meshes, we have coded from scratch this module accounting for flexibility and adaptability. In particular, using only the Linear Blend Skinning (LBS) function from [38], we incorporate all the major variants of SMPL model family, i.e. SMPL [38], SMPL-H [48], SMPL-X [42], and STAR [41], by only requiring passing the corresponding string argument during initialization. In addition, this class supports multiple extra (virtual) joint regressors, defined by the various methods used in this work, effectively unifying all SMPL calls and enabling more robust evaluation and method comparison. To enable reproducibility and also contribute to the software infrastructure of the computer vision community, a special Docker image comprising major deep learning (e.g. PyTorch 2.1.0) and 3D graphics (e.g. PyTorch3D) on top of CUDA 12.1 has been created and available online at `https://hub.docker.com/repository/docker/thanosch/mmtorch`.

Appendix B

Code Structure

The repository of the developed codebase as part of this work is available online at `https://github.com/charisoudis/M3C`. The code is structured as follows:

- `config`: Configuration files YAML format. Entire pipeline configuration is stored under the `pipeline` subpath wherein each the execution graph is described and import to individual models are used to include other config files in a nested manner.
- `data`: All datasets' files as well as `TORCH_HOME` path (such as PyTorch's hub model files) should reside or (symbolically) linked inside this directory. This way the application can access training/evaluation data and PyTorch dependencies in a clear and streamlined manner.
- `src`: Root path of all source code written or included as part of this work. This sources path is divided into three high-level fancily-named modules:
  - `ants`: Containing exclusively cloned repositories for codes of models we use in the project. Those are grouped by release year and each is forked form the original GitHub repository and edited in terms of import paths. Some, such as the DPVO's repo [54], have more modifications; we exclude all paths inside `ants` when running code statistics (e.g. LOC counting).
  - `aria`: Core interfaces, building blocks, and utility classes and functions all reside in this directory. In particular, interfaces and code structuring stuff can be found inside `aria/artifacts`, while the main utility classes such config readers, renderers, and detection processors are placed in

Statistic	Value	Description
LOC	15596	Total number of code lines in .py files
sLOC	10831	Source code lines (without blank lines)
sLOC (%)	69.44	Source code lines percentage
Comment Lines	3104	Comment lines inside .py files
Comment Lines (%)	19.90	Comment lines percentage
Blank Lines	1661	Blank lines inside .py files
Blank Lines (%)	10.65	Blank lines percentage

Table B.1: Code statistics of the developed codebase.

`aria/bolts`. Neural network sub-modules and components are placed in `aria/flux`, whereas their full, higher-level, implementations are given in `aria/illuminate`. Finally, dataset pre-processors and custom dataloaders are jointly placed under `aria/providers`.

— **`artisan`**: This folder comprises logic code written as part of this project. This includes the implementation of all defined execution nodes, the trajectory creation, global regression and optimization executors, and others. More information can be found in the code repository's as well as in each of the aforementioned sub-directories `README.md` files.

Some central statistics of the codebase that was developed exceeding 15.5K LOC, are provided in Table B.1.