Aerial Markerless Motion Capture

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät

der Eberhard Karls Universität Tübingen

zur Erlangung des Grades eines

Doktors der Naturwissenschaften

(Dr. rer. nat.)

vorgelegt von

Nitin Saini

aus Muzaffarnagar, Indien

Tübingen

2024

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Eberhard Karls Universität Tübingen.

Tag der mündlichen Qualifikation:	05.11.2024
Dekan:	Prof. Dr. Thilo Stehle
1. Berichterstatter:	Prof. Dr. Michael J. Black
2. Berichterstatter:	Jun.-Prof. Dr. -Ing. Aamir Ahmad
3. Berichterstatter:	Prof. Dr. -Ing. Hendrik P. A. Lensch

Abstract

Human motion capture (mocap) is important for several applications such as healthcare, sports, animation etc. Existing markerless mocap methods employ multiple static and calibrated RGB cameras to infer the subject's pose. These methods are not suitable for outdoor and unstructured scenarios. They need an extra calibration step before the mocap session and cannot dynamically adapt the viewpoint for the best mocap performance. A mocap setup consisting of multiple unmanned aerial vehicles with onboard cameras is ideal for such situations. However, estimating the subject's motion together with the camera motions is an under-constrained problem. In this thesis, we explore multiple approaches where we split this problem into multiple stages. We obtain the prior knowledge or rough estimates of the subject's or the cameras' motion in the initial stages and exploit them in the final stages.

In our work AirCap-Pose-Estimator, we use extra sensors (an IMU and a GPS receiver) on the multiple moving cameras to obtain the approximate camera poses. We use these estimates to jointly optimize the camera poses, the 3D body pose and the subject's shape to robustly fit the 2D keypoints of the subject. We show that the camera pose estimates using just the sensors are not accurate enough, and our joint optimization formulation improves the accuracy of the camera poses while estimating the subject's poses.

Placing extra sensors on the cameras is not always feasible. That is why, in our work AirPose, we introduce a distributed neural network that runs on board, estimating the subject's motion and calibrating the cameras relative to the subject. We utilize realistic human scans with ground truth to train our network. We further fine-tune it using a small amount of real-world data. Finally, we propose a bundle-adjustment method (AirPose+), which utilizes the initial estimates from our network to recover high-quality motions of the subject and the cameras.

Finally, we consider a generic setup consisting of multiple static and moving cameras. We propose a method that estimates the poses of the cameras and the human relative to the ground plane using only 2D human keypoints. We learn a human motion prior using a large amount of human mocap data and use it in a novel multi-stage optimization approach to fit the SMPL human body model and the camera poses to the 2D keypoints. We show that in addition to the aerial cameras, our method works for smartphone cameras and standard RGB ground cameras.

This thesis advances the field of markerless mocap which is currently limited to multiple static calibrated RGB cameras. Our methods allow the user to use moving RGB cameras and skip the extrinsic calibration. In the future, we will explore the usage of a single moving camera without even needing camera intrinsics.

Zusammenfassung

Die Erfassung menschlicher Bewegungen (Mocap) ist für verschiedene Anwendungen wie Gesundheitswesen, Sport, Animation usw. wichtig. Bestehende markerlose Mocap-Methoden verwenden mehrere statische und kalibrierte RGB-Kameras, um die Pose der Person zu bestimmen. Diese Methoden sind für Außenaufnahmen und unstrukturierte Szenarien nicht geeignet. Sie erfordern einen zusätzlichen Kalibrierungsschritt vor der Mocap-Sitzung und können den Blickpunkt nicht dynamisch anpassen, um die beste Mocap-Leistung zu erzielen.

Ein Mocap-Setup, das aus mehreren unbemannten Luftfahrzeugen mit Onboard- Kameras besteht, ist für solche Situationen ideal. Allerdings ist die Schätzung der Bewegung des Motivs zusammen mit den Kamerabewegungen ein unterbestimmtes Beschränkungs- problem. In dieser Arbeit untersuchen wir mehrere Ansätze, bei denen wir dieses Problem in mehrere Phasen aufteilen. In den ersten Phasen erhalten wir das Vorwissen oder grobe Schätzungen der Bewegungen der Person oder der Kameras, welche wir in den weiteren Phasen ausnutzen.

In unserer Arbeit AirCap-Pose-Estimator verwenden wir zusätzliche Sensoren (eine IMU und einen GPS-Empfänger) an den mehreren sich bewegenden Kameras, um die ungefähren Kamerapositionen zu ermitteln. Wir verwenden diese Schätzungen zur gemeinsamen Optimierung der Kamerapositionen, der 3D-Körperposition und der Form der Person, um die 2D-Keypoints der Person robust anzunähern. Wir zeigen, dass die Schätzungen der Kameraposition, welche nur die Sensoren verwendet nicht genau genug ist, und dass unsere kombinierte Optimierungsformel die Genauigkeit der Kamerapositionen verbessert und gleichzeitig die Posen der Person bestimmt.

Die Anbringung zusätzlicher Sensoren an den Kameras ist nicht immer machbar. Deshalb führen wir in unserer Arbeit AirPose ein verteiltes neuronales Netz ein, das an Bord läuft, die Bewegung der Person schätzt und die Kameras relativ zur Person kalibriert. Wir verwenden realistische menschliche Scans mit Ground-Truth, um unser Netzwerk zu trainieren. Wir nehmen eine weitere Feinabstimmung anhand einer kleinen Menge realer Daten vor. Schließlich präsentieren wir eine Methode zur Bündelanpassung (AirPose+), welche die anfänglichen Schätzungen unseres Netzwerks nutzt, um qualitativ hochwertige Bewegungen des Subjekts und der Kameras zu ermitteln.

Schlussendlich betrachten wir ein allgemeines Setup, das aus mehreren statischen und beweglichen Kameras besteht. Wir schlagen eine Methode vor, die die Posen der Kameras und des Menschen relativ zur Bodenebene schätzt, wobei nur 2D-Keypoints des Menschen verwendet werden. Wir lernen einen Bewegungsprior mit einer großen Menge an menschlichen Mocap-Daten und verwenden ihn in einem neuartigen mehrstufigen

Optimierungsansatz, um das SMPL-Körpermodell und die Kamerapositionen an die 2D-Keypoints anzupassen. Wir zeigen, dass unsere Methode nicht nur für Luftbildkameras, sondern auch für Smartphone-Kameras und Standard-RGB-Bodenkameras funktioniert.

Diese Arbeit ist ein Fortschritt auf dem Gebiet der markerlosen Mocap, welche bisher auf mehrere statische, kalibrierte RGB-Kameras beschränkt war. Unsere Methode ermöglicht es dem Benutzer, bewegliche RGB-Kameras zu verwenden und die extrinsische Kalibrierung zu überspringen. In Zukunft werden wir die Verwendung einer einzigen beweglichen Kamera erforschen, die nicht einmal eine Kamera-Intrinsik benötigt.

Acknowledgements

First and foremost, I thank my supervisors, Michael J. Black and Aamir Ahmad, for their invaluable guidance, support, and insightful feedback throughout my PhD journey. Their expertise and encouragement have been instrumental in shaping this thesis. Michael has created an amazing environment for cutting-edge scientific research, and I am glad I was part of it. I learned a lot from him and aspire to gain his leadership qualities. Aamir has been a great mentor, not just during my PhD but in my personal life too. I cannot imagine tackling those conference deadlines alone. His quick plans just a few days before the deadlines made them possible. At the same time, every piece of advice he gave me for my personal life made it very easy, be it for my driving license, marriage registration, preparations before having a kid and several other occasions.

I thank Prof. Hendrik Lensch for providing an additional review of this thesis. I also thank Prof. Andreas Zell for agreeing to be an examiner for my thesis defense.

I express my appreciation to my collaborators, Rahul Tallamraju, Eric Price, Elia Bonetto, Raffi Enficiaud, Chun-Hao Paul Huang, Ro man Ludwig, Yu-Tang Liu, Abhinanda Ranjit Punnakkal and Igor Martinovic. Even the pressure of conference deadlines became enjoyable when working with these wonderful people.

Special thanks to Nima Ghorbani and Arjun Chandrasekaran for their friendship, stimulating discussions, and insightful perspectives on engineering, science and research.

I am deeply grateful to Jonas Wulff, Joachim Tesch, Priyanka Patel, Ahmed Osman, Vassilis Choutas, Yan Zhang, Nikos Kolotouros, Georgios Pavlakos, Shoubhik Sanyal, and Parth Ghosh for their various types of help and insightful discussions. Those small talks during coffee breaks enabled me to move forward whenever I was stuck working on a problem.

I extend my gratitude to Raffi Enficiaud, Jean-Claude Passy, and Talha Zaman, from whom I learned much about programming. Their guidance has significantly enhanced my technical skills.

A heartfelt thanks to Melanie Feldhofer, Nicole Overbaugh, and Johanna Werminghausen for efficiently managing all administrative matters. Their support has allowed me to focus on my research without worrying about administrative tasks. I would also like to thank Rocko for bringing joy and a positive atmosphere to the office. His presence was a constant source of comfort and motivation.

Special thanks to Jun Saito and Ruben Villegas for their invaluable guidance during my internship at Adobe Research.

Finally, I thank my family and friends who have supported and encouraged me along this journey. Their belief in me has been a constant source of strength and inspiration.

I thank my wife Priyanka for taking care of everything at home so that I can focus on my PhD. Her ability to balance work, home, and personal growth is truly inspiring. I am incredibly grateful for her unwavering support and love. I thank my Sisters Preeti and Parul for their immense love. Growing up with them was the best time of my life, and their influence has been instrumental in shaping me into the person I am today. In the end, I thank my parents who are my source of inspiration. They are my early educators and I am proud of the values they instilled in me. I cannot thank them enough for their love and support in all of my efforts. Their influence on my life is immeasurable and for that, I am eternally grateful.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Motion capture (mocap) is the process of recording the 3D motion of a subject. 3D human mocap is important for various applications in fields such as animation, sports, healthcare, etc. Conventional mocap methods rely on active (IMUs) or passive IR (Infrared) markers placed on the body of the subject. These markers make the mocap process inconvenient and time-consuming as the markers need to be carefully placed on the subject and the cameras/markers need to be calibrated before the session. Additionally, most of these systems are not suitable for long-range outdoor mocap. For example, active IMU-based systems suffer from drift, due to which they are unable to get the global motion correctly. Because of the drift, they need to be calibrated again after a short duration of mocap, making them unsuitable for longer mocap sessions. The passive IR-based systems need static cameras, which makes them unsuitable for long-range mocap. That is why, recent mocap methods are markerless (12; 13; 14; 15). They do not need any markers or IMUs on the subject's body to reconstruct his/her 3D motion. A typical markerless mocap setup consists of multiple static and calibrated RGB cameras. The calibration is a separate step that needs to be done before the mocap session after which the cameras cannot be moved. However, in practice, cameras need to be moved during the mocap session because of multiple reasons, such as, avoiding direct sunlight outdoors, or to get a better viewpoint for mocap etc. In this thesis, we focus on a unique markerless mocap setup where the RGB cameras are mounted on multiple unmanned micro-aerial vehicles (UAVs). This setup allows the cameras to be moved during the mocap session and, since it uses RGB cameras, there is no need for subject preparation such as placing markers on the body. However, reconstructing human motion using such a setup is extremely challenging, and we discuss these challenges in further sections. First, we discuss the perspective camera projection model, and then, we discuss a simple triangulation method to recover the 3D point from the corresponding 2D point in two camera views. We further build upon this to discuss the challenges of recovering a 3D human body given the corresponding images in multiple views. After this, we present our approach to handle these challenges. Later in this chapter, we also discuss related works.

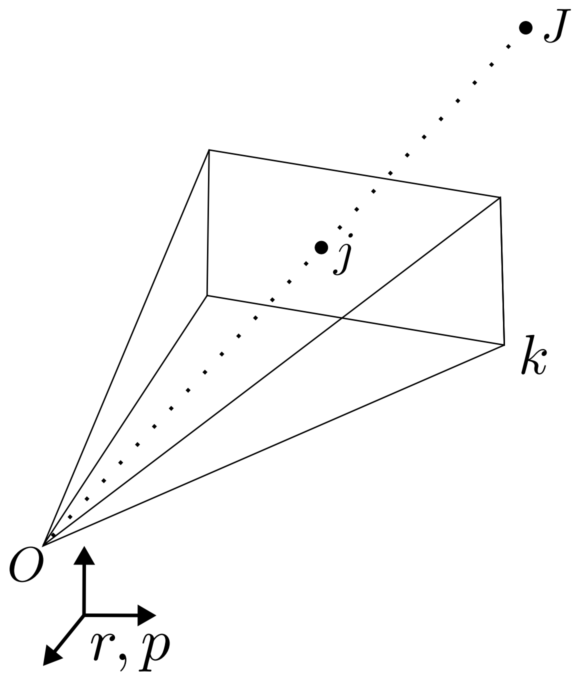


Figure 1.1: Perspective camera projection model.

1.1 Background

1.1.1 Perspective projection

The image formation process on a camera plane is mathematically modelled using the perspective camera projection model as shown in Fig. 1.1. Consider a camera in the 3D space with optical center O, intrinsic matrix k, rotation matrix r and translation p of the extrinsic parameters.

$$r = \begin{bmatrix} r11 & r12 & r13 \\ r21 & r22 & r23 \\ r31 & r32 & r33 \end{bmatrix}, \quad p = \begin{bmatrix} p1 \\ p2 \\ p3 \end{bmatrix}, \quad k = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}.$$

The projection, $j = [x, y]^T$, on the camera plane of a point $J = [u, v, w]^T$ in the 3D space is given as:

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r11 & r12 & r13 & p1 \\ r21 & r22 & r23 & p2 \\ r31 & r32 & r33 & p3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix}. \tag{1.1}$$

The camera extrinsic matrix $\left[\begin{smallmatrix} r & p \\ 0 & 1 \end{smallmatrix}\right]$ transforms J from a global coordinate frame to the camera frame. The matrix k further transforms it from 3D space to the image plane to

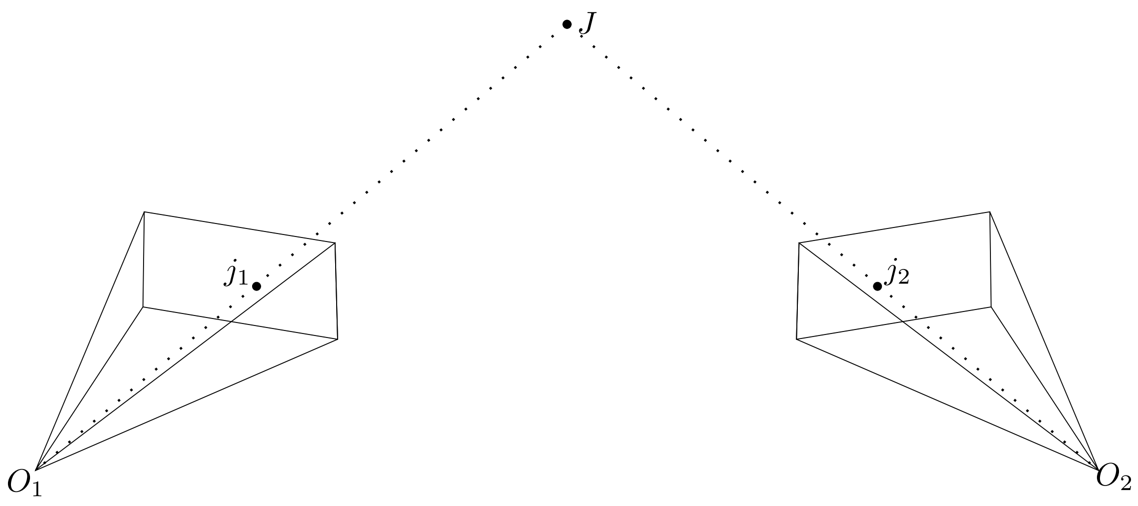


Figure 1.2: Triangulation of two 2D points (j_1 and j_2) to get the 3D point J.

give j.

1.1.2 Triangulation

The projection of a 3D point to the image plane results in the loss of depth information. It is not possible to uniquely reconstruct the 3D point J given the 2D projection j and camera extrinsic parameters r and p. All the points on the ray connecting the camera's optical center and the 2D point j are valid solutions. Eq. (1.1) can be written in the form of two linear equations and estimating three parameters $[u, v, w]$ given k, r, p and j will lead to infinite solutions. That is why it is an under-constrained problem.

The depth ambiguity can be resolved by exploiting information from another camera view. If the projection of the same 3D point J is available in another camera, it will give another two linear equations and estimating three parameters using a set of four linear equations will give a unique solution. As discussed in the next section, triangulation is the key principle used by passive marker based methods and markerless methods to estimate the 3D point given the 2D points in multiple views.

1.1.3 IR marker vs Markerless Motion Capture

An IR-based mocap system consists of multiple IR cameras and IR reflective markers placed on the subject's body. The cameras capture the IR light reflected by the marker and get the 2D projection of these 3D positions on the 2D image plane. In Fig. 1.2, we see the 2D projection of a 3D body marker onto the image plane of each camera in a 2-camera setup. The rotation and position components of the extrinsic parameters of camera c are r_c and p_c respectively. j_c is the 2D projection of 3D point J on camera c. The projection parameters (intrinsics) of these cameras are provided by the manufacturer and the extrinsic parameters are obtained before the mocap session using a camera calibration step. From a mocap recording, the mocap algorithm uses the 2D projections (see sample image in Fig. 1.3a) j_c on each image and recovers the 3D point by finding the intersection of rays $O_1 j_1$ and $O_2 j_2$.

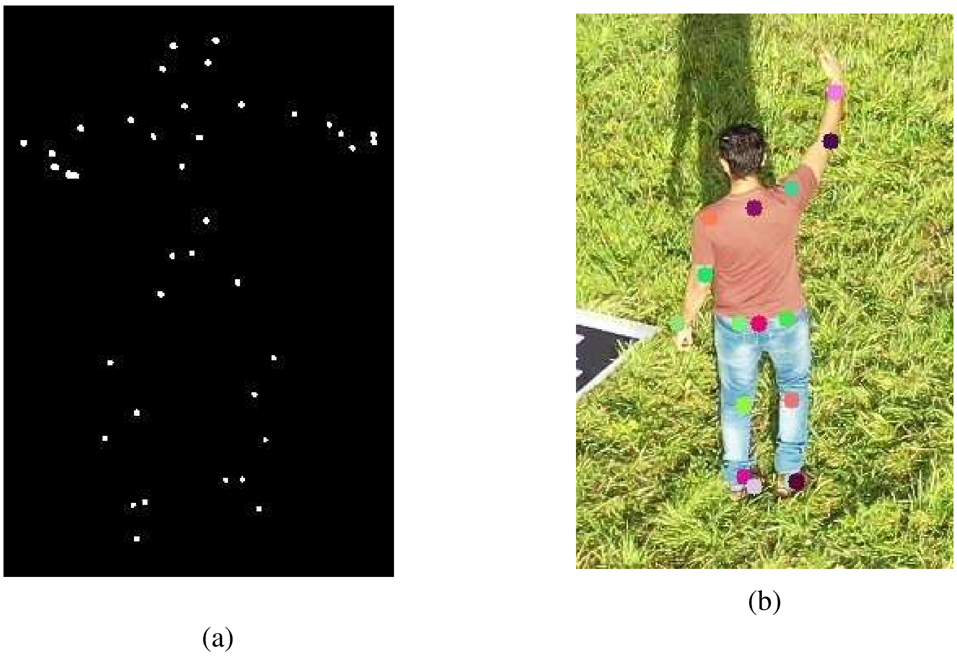


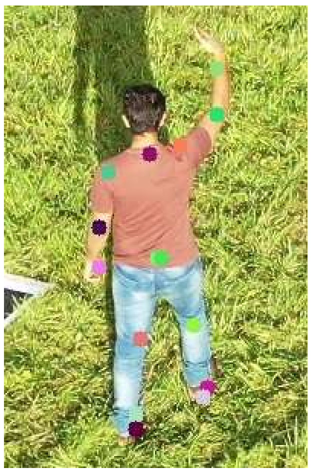
Figure 1.3: (a) Image showing 2D projections of IR markers on an IR camera. (b) 2D keypoints detected by a keypoint detector overlaid on top of the original image.

Markerless mocap systems are inspired by passive IR marker-based systems and work on similar principles. However, there are a few differences. There are no physical markers on the subject's body and the 2D image is RGB instead of Black and White. The markerless mocap algorithms rely on computer vision techniques to detect keypoints on the 2D images. Modern deep learning-based detectors can estimate the 2D human body keypoints on an RGB image, (16; 17). In Fig. 1.3b, we show a sample image with the 2D keypoints overlaid on top. This makes the placement of the passive markers on the subject's body unnecessary. Similar to the IR-based systems, the 3D keypoints on the subject's body are then recovered by triangulation. RGB-based markerless mocap systems can work outdoors and have low setup time because no subject preparation is required. Furthermore, they are cheaper because of the easy availability and low cost of the RGB cameras.

1.2 Challenges

1.2.1 Noisy 2D keypoints

Even though modern deep learning based detectors are quite good at estimating the 2D keypoints on the human body, they can be noisy and erroneous. A common type of error they make is swapping the keypoints from the left side of the body with the right side. An example is shown in Fig. 1.4a. Note the colors of the keypoints compared to Fig. 1.3b. Another example is shown in Fig. 1.4b, where the network also detects the 2D keypoints over the shadow of the person. Such errors result in a completely wrong estimate of the 3D points after the triangulation. Since they are trained using human-annotated data, they are not perfect and include human annotation errors. For example, the annotation of a particular keypoint (e.g. shoulder) on an image by a human annotator can be slightly different from another annotator. That is why, even for two similar images, the estimated keypoint can differ a little. Additionally, these detectors run on each image frame independently, and that is why their estimates are temporally incoherent. During triangulation, this small noise on the 2D plane can result in high noise in the 3D world.


(a)

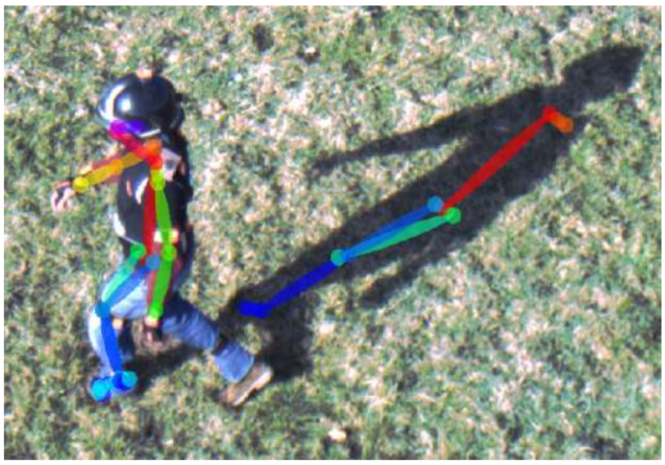

(b)

Figure 1.4: (a) Wrong 2D keypoints detected by a detector where left side keypoints are swapped with right side. (b) 2D keypoint detector detects the keypoints on the shadow of the person.

1.2.2 Estimating poses of a moving camera

To accurately get a 3D point from multiple 2D keypoints, we also need accurate poses (r_c, p_c) of the cameras. Existing methods usually obtain the camera poses beforehand using calibration patterns (ArUco markers(6)) and the cameras are not allowed to move once this calibration is done. This method works fine for static cameras, however, using this for moving cameras is extremely challenging. The pattern should be visible in all the camera frames at all times during the mocap. Additionally, the pattern position should not change because the camera calibration happens relative to the pattern. As a result, the cameras cannot move freely to keep the moving person in the field of view. Their motion becomes heavily restricted to keep the pattern and the freely moving person in the field of view. Additionally, the calibration from these patterns is not completely reliable. Since these patterns are planar, their orientation cannot be uniquely determined if they are tilted in the image (see (18)).

Another way of estimating the motion of a moving camera is to use a SLAM method (Simultaneous Localization And Mapping). SLAM methods use static textured surfaces in the video to estimate the motion of a single camera relative to its starting position. This allows the pose tracking of each moving camera relative to their starting points. However, for triangulation, we need the poses of all the cameras in a common reference frame. Also, the background should have sufficient texture for SLAM algorithms to work. Another way of estimating the pose of the cameras is by using extra sensors on the cameras such as GPS, IMU etc. The observations of these sensors can be fused together to estimate the camera poses relative to the GPS coordinate system. However, the noise and drift in these sensors are quite significant for the purpose of triangulation. In Chapter 2, we use such sensors to approximate the poses of RGB cameras.

1.2.3 Representing Human Motion

The 3D human motion is usually represented as a series of 3D human poses at different time steps over time. However, a human pose can be defined in multiple ways. The traditional and most common way is by using a human skeleton consisting of 3D positions of human joints. A better way to represent human pose is by using a 3D human body model. We discuss both of them in the following sections.

Skeletal Representation

The skeletal representation of the human pose is the 3D position of the human body joints. However, the actual human joints are not used because the real human skeleton is a complicated structure. In practice, fewer important body joints (14-24) are used to represent a human body pose (fingers, jaw etc are excluded). The connection between the joints can be treated as the bones. Most human poses can be represented using this representation, but there are various human poses which have the same joint positions but different joint angles. Such poses involve rotation change along a bone (e.g. wrist rotation). Another significant drawback of this representation is the independence of all the joints. In reality, human joints cannot be positioned independently. While the joints in this representation can move independently and assume any position, the real human bones have fixed lengths, which constrains the distance between two connected joints. That is why the skeletal representation is susceptible to varying bone lengths.

SMPL Body Model

SMPL (Skinned Multi-Person Linear model) (19) is a widely used parametric human body model. It is learned by using thousands of high-quality body scans of people with a wide variety of body types. It is parameterized by two latent parameters: pose and shape. Given the pose and shape parameters, it outputs the 3D human mesh and the 3D locations of the body joints. It uses 24 body joints, and the pose parameters are the axis angles between the two connecting bones at all these joints. These are also called joint angles. The pose parameter is represented by θ. It is a 69×1 vector, i.e. 3 axis-angle values for each of the 23 joints. The rotation of the root joint is ϕ, which is again represented by 3 axis-angle values. The position of the root joint is τ, which is represented by 3 coordinate values in the 3D world. The shape parameter is a vector of 10 or more real coefficients corresponding to the eigen shapes learned from a large dataset of human scans. The SMPL shape parameter β is a (10×1) vector whose elements are weights of the 10 most significant eigen shapes (refer (19) for details). The pose parameters θ determine the articulated pose of the human body, the root rotation ϕ and position τ determine the rotation and position of the body in 3D world, and the shape parameters decide the body's shape features such as height, girth, etc. Since the joint angles are defined relative to the parent bone, the joint angles at all the joints are independent of each other. The shape parameters decide the underlying skeleton of the SMPL body model. Thus, using this representation solves the problem of unrealistic variations in the bone lengths. SMPL-X (20) is the successor of the SMPL body model which retains the qualities of the SMPL model (the pose and shape parameters) and includes the hands and face parameters too. However, in this thesis, we only use the pose and shape parameters of SMPL-X. We use the same representation θ, ϕ, τ and β for both SMPL and SMPL-X. The dimensions of these parameters remain the same, but their values for SMPL and SMPL-X are not directly interchangeable. Both SMPL and SMPL-X models have a vertices regression function ($\mathcal{S}$) and a 3D joint regressor function ($\mathcal{J}$). Both of them take in the pose and the shape parameters and outputs the posed 3D mesh and the 3D location of the joints, respectively.

1.2.4 Active viewpoint optimization

Some viewpoints are better for estimating the subject's pose compared to others. One reason for this is the self-occlusion of the subject's body parts. If a body part is occluded in a view, the 2D keypoint estimator is most likely to fail, resulting in a bad estimate of the subject's 3D pose. While the viewpoint of a static camera cannot be changed, a freely moving camera can adjust its viewpoint to optimize for the best mocap results. Therefore, multiple flying UAVs with onboard RGB cameras are an ideal solution for outdoor markerless mocap. However, to actively change the viewpoint, the mocap algorithm should give out results in real time such that the motion of the UAVs can be actively planned. This is a challenging problem considering the limited payload capacity generally available on UAVs. More computation power requires heavy and power-hungry hardware onboard, which results in either a very short flight duration or a heavier and higher-capacity battery. Even with such hardware, the amount of computation cannot match the conventional desktop computers. Therefore, the mocap algorithm should work on embedded hardware in real-time. In the next paragraph, we discuss such a method HMR (4) which has proved to be capable of running on an embedded GPU in real-time.

HMR (4) is the first method to regress the SMPL body and shape parameters from a single RGB image. Its neural network consists of a ResNet50 (21) backbone and an autoregressor network. ResNet50 backbone is a neural network consisting of 48 convolutional layers, 1 maxpool and 1 average pool layer. It extracts features from the input RGB image and passes them to the autoregressor network which consists of three fully connected layers. The autoregressor network takes in these features along with mean pose (θ, ϕ) and shape β, and predicts the pose and shape offsets. These offsets are added to the input pose and shape values to get the output of the autoregressor network. The resultant values along with the ResNet50 features are again fed back to the autoregressor network to improve the estimates. Three such iterations are done to get the final output from the autoregressor network. This network architecture is adopted with minor modifications in multiple following works (15; 22; 23; 24; 25), including our work AirPose (Chapter 3).

1.2.5 Local or undesired minima

A straightforward way to fit a body model to 2D keypoints on an image is by minimizing their squared distances from the 2D projections of 3D keypoints on the body. This process is very sensitive to initialization. All the joints move independently during the fitting process, which can result in implausible poses. Ideally, the 3D pose change from initial to target pose should happen over a manifold of plausible human poses. However, integrating such a constraint in the fitting process is not straightforward. Bogo et al. (1) addressed the problem of self-intersection of body parts by approximating the body model with capsules and penalizing the self-intersection. We can see in Fig. 1.5, even though the 2D keypoints are correct, the resulting pose has the left hand penetrating the body. Putting self-intersection constraints on capsules gives a more accurate resulting pose. The capsule approximation is made because calculating intersection using simple shapes (such as cylindrical capsules) is much easier than with a complicated body mesh. In some cases, a single view is not sufficient to resolve the 3D pose of the person. Multiple valid 3D poses can result in the same 2D keypoints. An example is shown in Fig. 1.6 where two different 3D poses results in the same 2D projection. Since both the poses are valid human poses, a new view of the same person is needed to disambiguate between the poses.

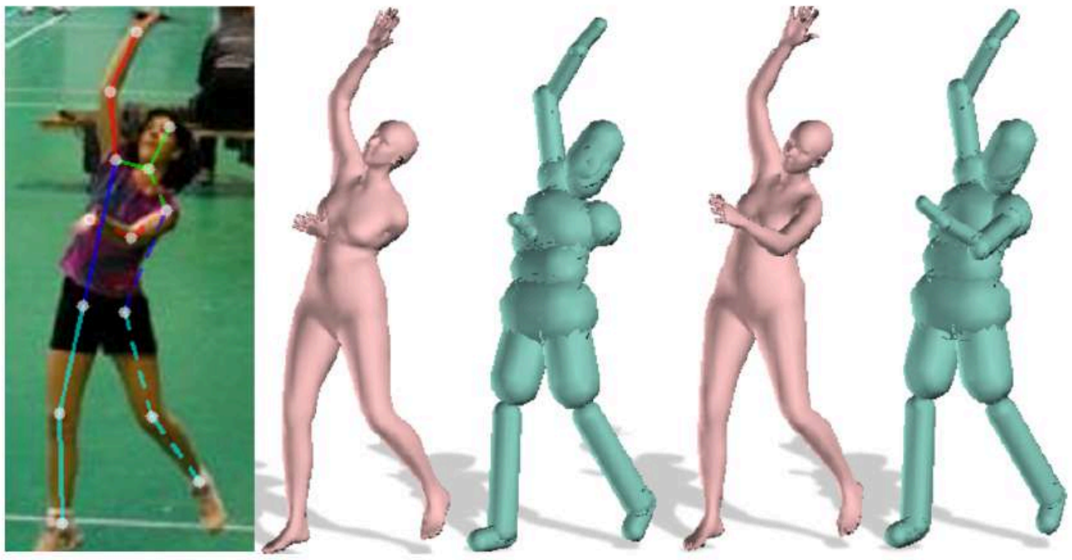


Figure 1.5: Body model cannot avoid self inter-penetrations; Bogo et. al. (1) puts explicit constraints by approximating body with capsules (image taken from (1)).

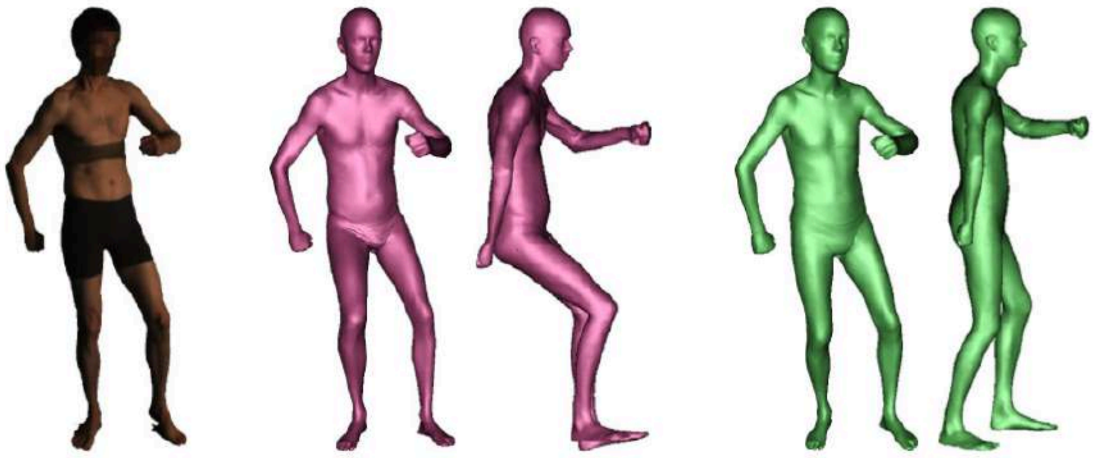


Figure 1.6: The same silhouette/keypoints can be explained by two combinations (magenta and green) of body shapes and poses (image taken from (2)).

1.2.6 Temporal coherency

Fitting a human body model to 2D keypoints in a single frame might look fine, however, when visualized for a video, they look jittery and unnatural. This is because the 2D keypoint detector does inference on each video frame independently, making the detections temporally incoherent. Independently fitting to these incoherent detections results in temporally incoherent body poses. In reality, the human body is driven by muscular forces, resulting in smooth and coherent motion. The ideal way of modelling human motion is by modeling these forces. However, there are 320 pairs of skeletal muscles and modelling their dynamics is an extremely challenging task. Existing works avoid muscle modelling and use some post-processing step to smoothen the per-frame fitting results. For example, (12) uses a DCT based human motion smoothing. Other methods try to learn the human motion models from mocap datasets and use them either for fitting (8) to the 2D keypoints or integrated in a deep neural network for human motion regression from videos (26).

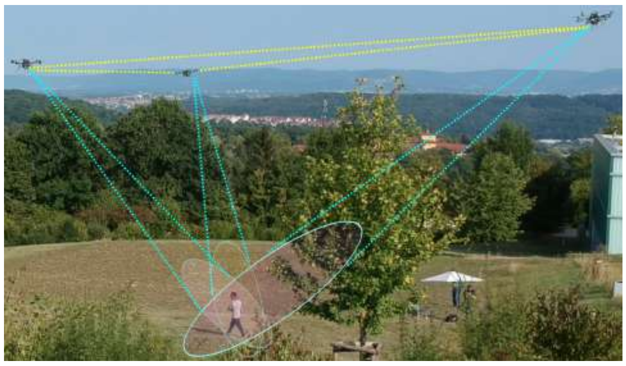
1.3 Our approach

Our goal is to estimate the motion of a freely moving human subject using multiple RGB cameras, each mounted on a flying unmanned aerial vehicle. As opposed to the conventional approach of first calibrating the cameras using calibration patterns and then estimating the subject's pose, in this thesis, we explore multiple approaches to combine the two steps, i.e. estimating the subject's pose and the camera's poses simultaneously. To do so, we explore three different approaches.

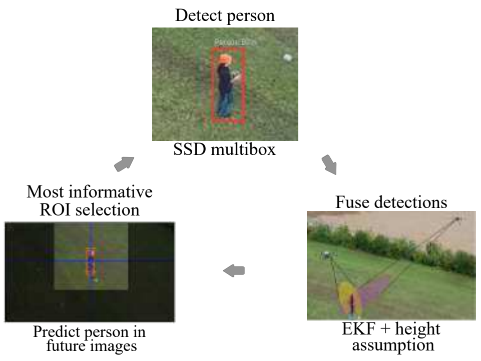
1.3.1 AirCap-Pose-Estimator

A simple aerial mocap system would consist of multiple UAVs, each with an onboard camera to capture video of the subject and onboard computing capability to estimate the subject's pose. However, estimating the camera poses using such a simple setup is not sufficient, and we need additional hardware onboard. In our first approach AirCap-Pose-Estimator, we use a system of autonomously flying UAVs each mounted with an RGB camera, a CPU, a GPU, onboard storage, wifi module and other sensors such as IMUs and GPS. Our full method consists of two phases, 1) online data acquisition, and 2) offline human pose and shape estimation.

In the first phase, the subject moves in an open area and the UAVs fly autonomously tracking and following him/her (see Fig. 1.7a). This is achieved using existing cooperative detection and tracking (CDT) and formation control methods (27; 28). The CDT utilizes the onboard extra sensors along with the information from videos to estimate the approximate pose of the UAVs and the position of the subject (see Fig. 1.7b). The CDT first detects the person in a video frame, then projects the 2D bounding box with



(a)



(b)

Figure 1.7: (a) Our online data acquisition setup consists of three UAVs tracking and following the subject. (b) Overview of the CDT algorithm.

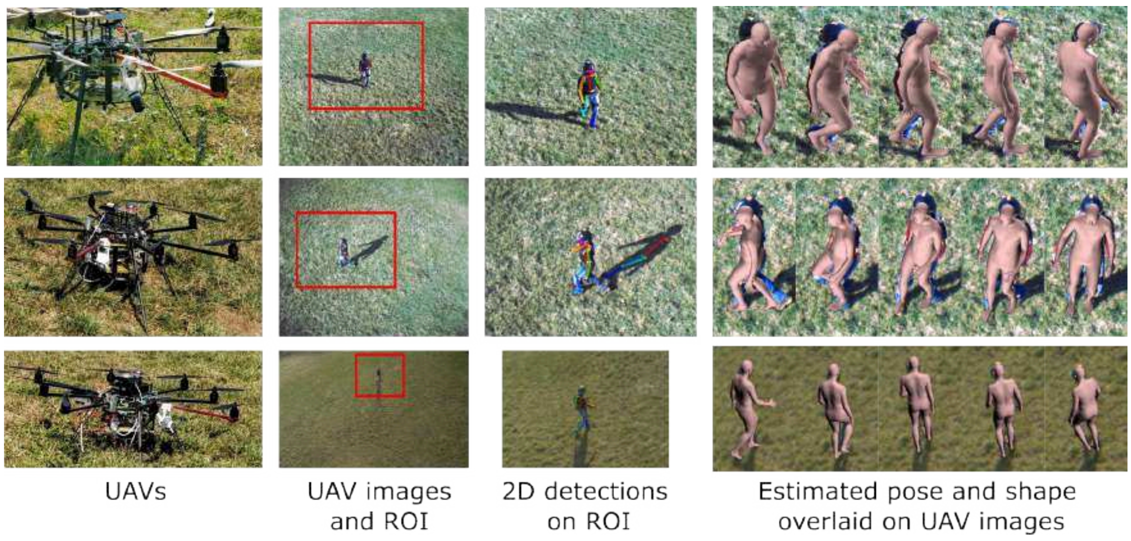


Figure 1.8: Overview of our offline 3D body fitting method. For every time instant, we crop the ROI provided by the CDT, detect the 2D keypoints and fit the SMPL body model simultaneously to all the views.

uncertainty in the 3D world using the camera poses from the sensors and an assumed height of the person. This 3D information from all the views is fused by an extended Kalman filter (EKF) to predict the person in every image. This prediction is further used to crop the region of interest (ROI) for the person detection in the next frame. Since the cameras are rigidly attached to the UAVs, we indirectly get the poses of the cameras. We get approximate poses because IMUs and GPS both have drift, and we cannot get precise camera poses using only these sensors. During the acquisition phase, all the UAVs try to keep the subject centered in the camera frame while avoiding obstacles and maintaining a formation such that the subject is best viewed. All the image frames, sensor data, and pose estimates of the cameras and the subject get stored on the onboard storage and then processed in the second phase.

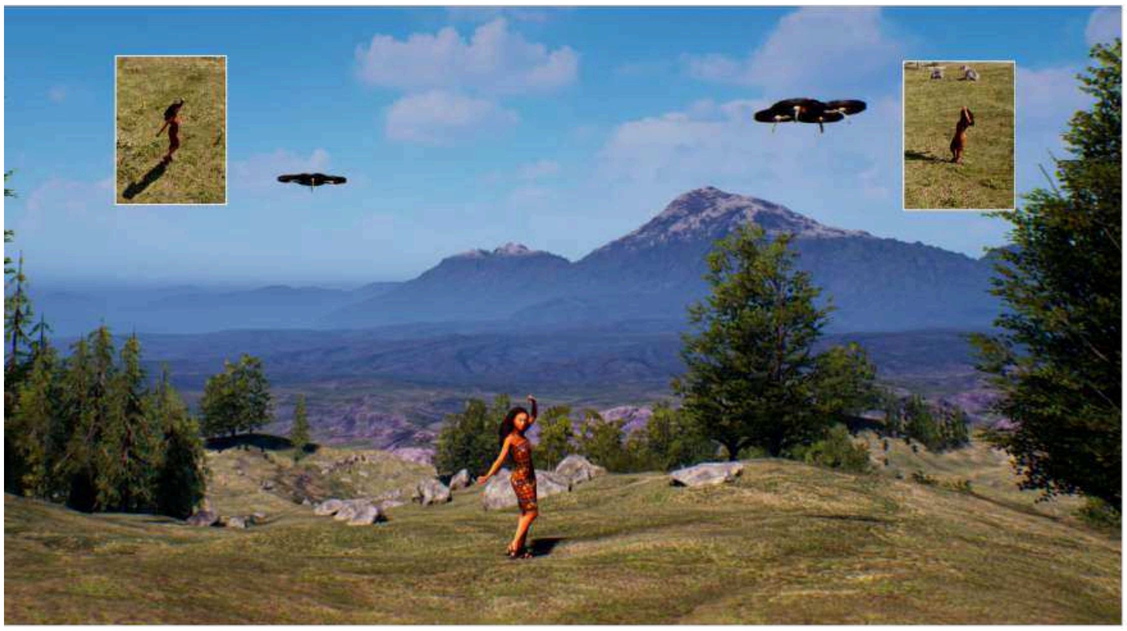
In the second phase, we use the collected data in the first phase to estimate the global and articulated 3D poses of the person and the UAVs. In the first phase, the CDT cannot run at a high frame rate because of the limited computation available onboard. We run this method again offline on the raw sensor data to get more accurate estimates of the UAV poses and the subject's position and orientation. Then we run 2D keypoint detectors on the collected RGB frames. We use multiple detectors to avoid the common problems with 2D keypoint detectors, as discussed in Sec. 1.2.1. Finally, we fit the SMPL body model to these 2D keypoints at each time step on all the camera views simultaneously (see Fig. 1.8). We initialize the camera poses and the subject's position and orientation using the estimates from the offline CDT. During the fitting, we optimize for the 3D global and articulated pose of the subject and the poses of the cameras. To avoid the implausible subject poses as discussed in Sec. 1.2.5, we use a human pose prior model VPoser (20) trained on a large dataset of real human poses. Even though we optimize for the camera poses, we restrict them to not deviate much from the CDT estimates.

To evaluate our method, we compare with reference human pose data obtained using an IMU-based method (29). We collect data from an IMU suit worn on the subject's body while collecting data in the first phase of our method. We process this data offline using (29) to get the reference poses of the subject.

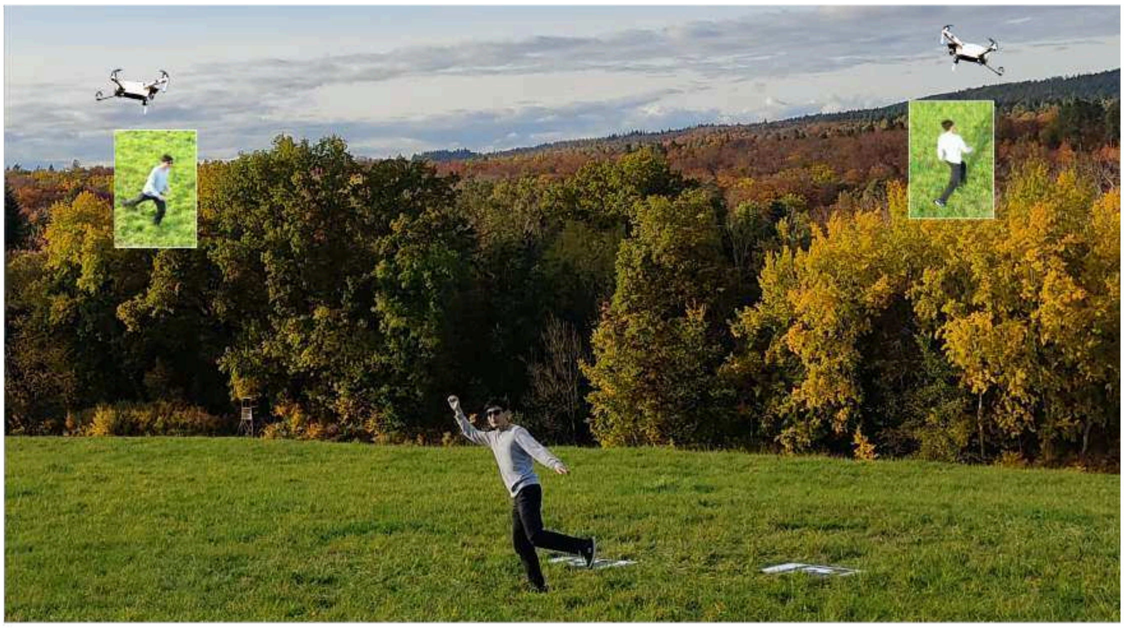
1.3.2 AirPose

A major limitation of the AirCap-Pose-Estimator is that the subject's pose is estimated offline after the data collection. It cannot run in real-time because it is an optimization-based method which requires many optimization iterations for each frame, including gradient computations for each frame. Additionally, the onboard compute hardware is limited, thus unable to handle such heavy processing.

In our work AirPose, we present a method that can estimate the subject's 3D global and articulated pose onboard in real-time. As opposed to the optimization-based method which refines the subject's pose in multiple iterations, we use a light-weight neural network AirPoseNet, which takes in an image and regresses the 3D global and articulated pose of the subject relative to the camera. AirPoseNet is designed such that the overall computation can run in a distributed and decentralized way. To achieve this, identical instances of AirPoseNet run on each UAV independently while sharing minimal information across views. A major challenge in training such a network is the availability of training data. We address it by creating a synthetic dataset using scans of real people rendered in a realistic virtual environment using Unreal Engine (see Fig. 3.4). We use the ground truth (GT) body fitting for the scans from the AGORA dataset (7) and train AirPoseNet using the paired images and the GT data. To improve its performance on real-world images, we fine-tune it on a small amount of real-world data. We collect this data by flying two DJI UAVs manually in the wild while the subject performs various



(a)



(b)

Figure 1.9: Data samples from (a) synthetic data, and (b) real-world data showing the scene and the corresponding captured images from two viewpoints (shown in the inset). (a) We place 3D human scans in realistic environments in Unreal Engine and render them from multiple viewpoints. We use the synthetic data along with the 3D ground truth to train AirPose. (b) We fly two DJI UAVs in the wild to collect real-world data which is used along with the 2D keypoints to fine-tune AirPose.

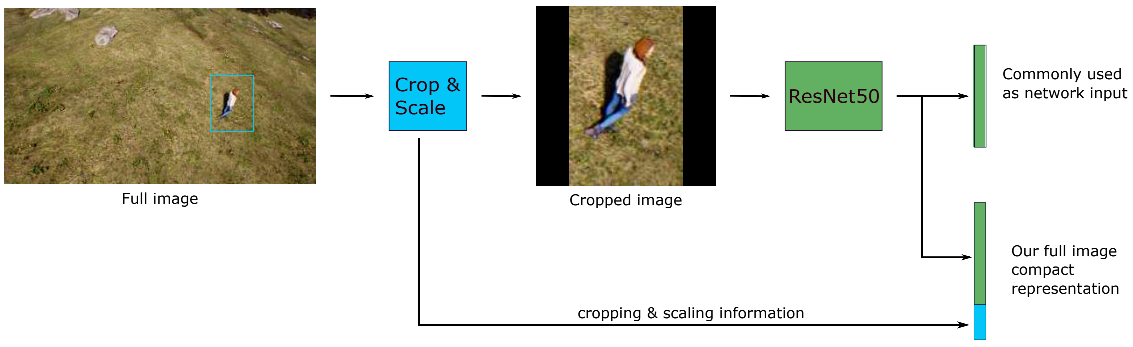

Full image
Crop & Scale
Cropped image
ResNet50
Commonly used as network input
Our full image compact representation
cropping & scaling information

Figure 1.10: Our compact representation for full image, which improves the estimation of the global pose of the person from a single view.

actions (see Fig. 1.9b). Since the GT fittings are not available for this data, we use the 2D keypoints obtained using a 2D human keypoint detector to supervise the finetuning process.

We take inspiration from the neural network introduced in HMR to design our decentralized and distributed neural network. AirPoseNet consists of a ResNet50 (21) backbone which extracts the image features, followed by multiple stages of an autoregressor module which takes in a fixed initial pose and outputs the correctives that are added to the input pose to get the estimated pose. The information exchange between the views happens after each autoregressor stage output. As opposed to the existing methods (4; 22), which process the cropped region around the subject, we use the full image, which improves the global position estimate of the subject. However, a small image region is usually occupied by the subject, and processing the full image becomes computationally expensive for the neural network. Therefore, we introduce a compressed novel representation of the full image resulting in the processing of only the region of interest and significantly reducing the computation overhead. As shown in Fig. 1.10, the full-size image is cropped and scaled to a fixed size which is fed to the feature extractor (ResNet50) and the output features are further fed to the autoregressor network to estimate the subject's pose. This representation ignores the cropping and scaling information, which is crucial for the estimation of the global pose of the person. We append the cropping and scaling parameters to the ResNet50 features, which result in a complete yet compact representation of the full image.

We show qualitative results and compare them with a baseline method, which is an adaptation of (22). We implement AirPose on the same hardware as AirCap-Pose-Estimator and evaluate its performance by hardware-in-the-loop implementation and a simple synchronization strategy. We found that AirPose itself can run at 20 FPS while the full pipeline including image acquisition, human detection, preprocessing etc. runs at 3 FPS. Finally, we also introduce an optimization-based post-processing method, AirPose$^+$ to improve the 3D human motion by utilizing the heavy compute available offline (see Fig. 1.11). It takes in the AirPose estimates and fits them to the 2D keypoints while

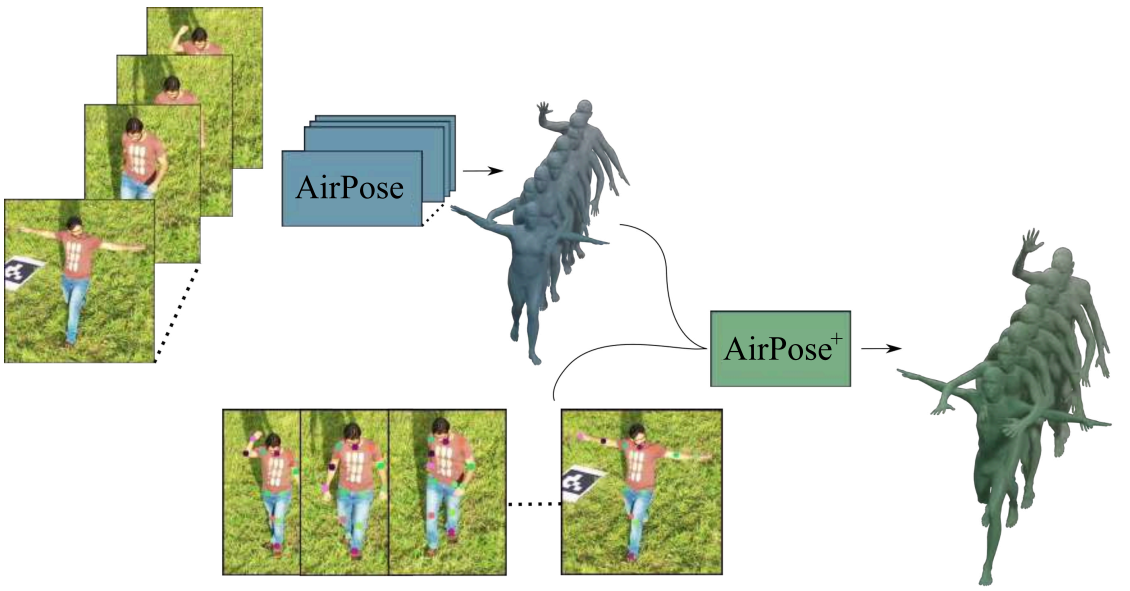


Figure 1.11: Overview of our full pipeline. AirPose takes in a single frame and estimates the articulated and global pose of the person relative to the camera for that frame. AirPose$^+$ is an offline bundle adjustment method, which takes in the AirPose estimates and fits them to the 2D keypoints for the whole sequence.

introducing temporal constraints, resulting in smooth and temporally coherent motions.

AirPose can estimate the pose of the person and the cameras relative to each other using only the RGB cameras without needing any extrinsic camera calibration or extra hardware. While the AirCap-Pose-Estimator can estimate the pose of the subject and the cameras relative to a static global reference frame (GPS frame), AirPose estimates the subject's pose relative to each camera. This is fine for applications such as gesture-controlled UAVs, however, global and articulated poses relative to a ground plane are required for motion capture applications where a 3D character needs to be placed in a 3D scene. That is why, we introduce SmartMocap to estimate the 3D articulated and global poses of the subject and the cameras.

1.3.3 SmartMocap

In SmartMocap, we consider a general mocap system that consists of single/multiple moving/static extrinsically uncalibrated RGB cameras. Our goal is to estimate the motion of the cameras, articulated motion and the global motion of the subject relative to a static reference frame. This is a challenging problem, and we address this using the following insights (also shown in Fig. 1.12). First, we use the ground plane as the reference frame of our mocap system. Next, we represent the subject's pose using the SMPL body model, where the subject's motion is represented as a trajectory of human poses such that the

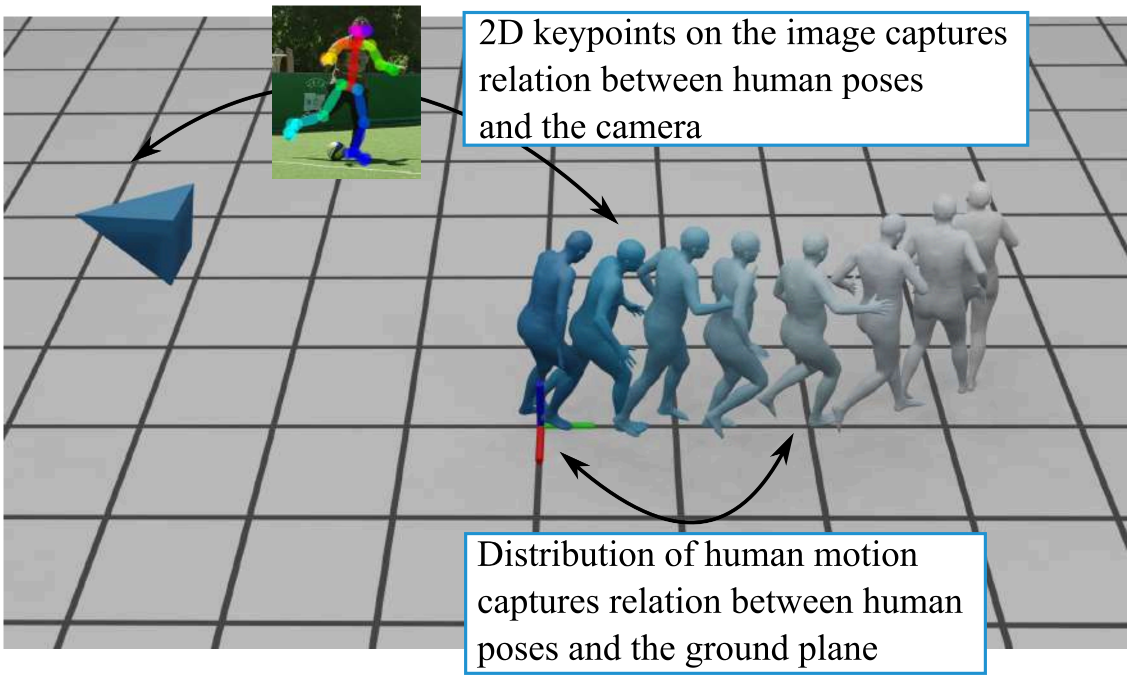

2D keypoints on the image captures relation between human poses and the camera
Distribution of human motion captures relation between human poses and the ground plane

Figure 1.12: SmartMocap uses a motion prior network to encode 3D human motion relative to the ground plane. At the same time, it fits this human motion representation to the 2D human keypoints and estimates the 3D human motion and the camera poses relative to the ground plane.

origin is the ground projection of SMPL's root joint in the first frame. Next, we train a variational auto-encoder model (motion prior) that encodes and learns the distribution of human motions in relation to the ground plane. Finally, we use the 2D human body keypoints to resolve the human articulated pose and global pose relative to the camera. In summary, our motion prior network encodes the human body relative to the ground plane, while the 2D keypoints resolves the camera pose relative to the 3D human body. We utilize both of them together in an optimization problem to estimate the 3D human motion and the camera motion relative to the ground plane.

The motion prior VAE network is a crucial component of our method. We use the AMASS dataset (3) to train this network. We resample the motion sequences to a fixed frame rate of 30 FPS, select 25 consecutive frames and represent all the frames relative to the ground projection of the root joint in the first frame to get a training sample.

We use our insights to formulate a novel multi-stage optimization problem and jointly estimate all the motions relative to our set origin. Such optimization is complicated and very likely to converge to an undesired minimum. That is why, we first initialize the articulated pose of the subject and poses of the cameras at each time frame using a monocular regression method PARE (30). The subject's global pose at every frame is initialized to the mean starting pose in the latent space of our motion prior network. Because of the fixed length of the training sample, we optimize for the poses of the cameras and the global and articulated poses of the subject in chunks of 25 frames. Next, we stitch together the sequences such that the subject pose at the last frame of a chunk is aligned with the pose at the first frame of the next chunk. We again optimize the full sequence using the same loss function to get the final estimate of the camera and subject's poses.

We show quantitative results on the RICH dataset (31) and compare them with an existing monocular method. We also show results on the AirPose dataset. To show the ease of use of SmartMocap, we collect a new dataset using 4 smartphones (2 static and 2 moving) and show qualitative results on this dataset.

1.4 Related works

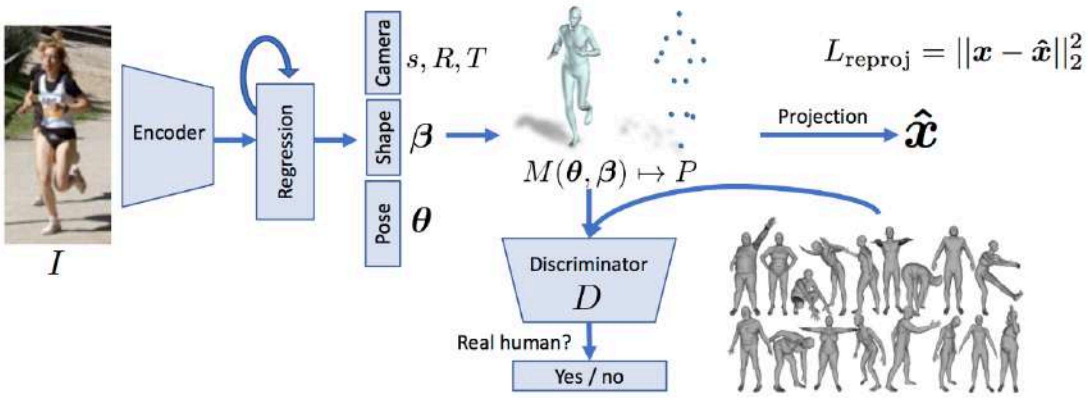
1.4.1 Monocular methods

Early monocular methods (32; 33; 34) use 3D human skeleton models to estimate human pose from a single RGB image. However, as we discussed in Sec. 1.2.3, 3D skeleton representation has various drawbacks. To overcome them, they impose physical constraints such as bone length consistency and joint angle constraints to reason about the 3D structure of the human skeleton. For supervision, they need the 2D keypoints or edges from a human annotator to reconstruct the pose of the subject in a single frame. Since they process all the frames independently, it is very difficult to put explicit constraints on the body shape and pose.

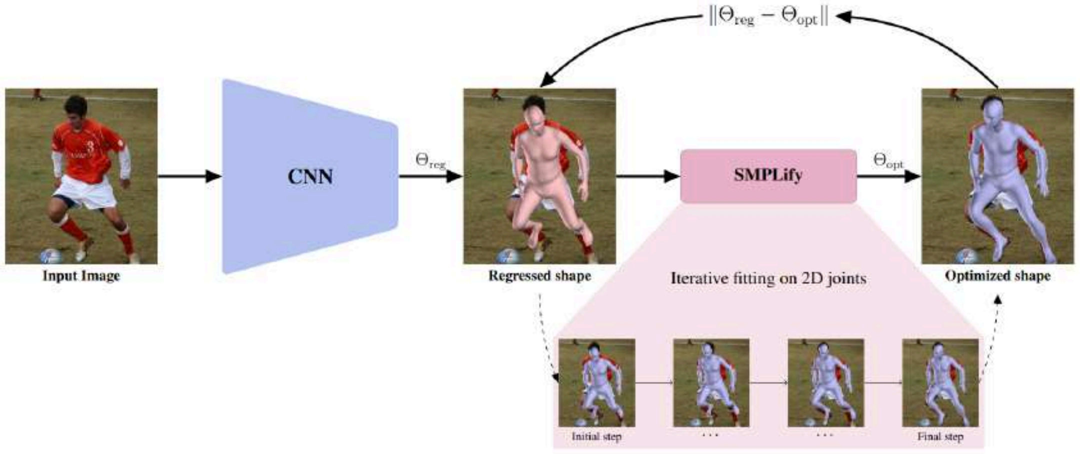
Guan et al. (2) use a human body model (35) which is learned from human scans to constrain the body shape. Their method takes in the 2D keypoints from a human annotator and the 2D silhouette of the subject to estimate the pose and shape of the person in a single image. They fit the body model to the 2D keypoints and body silhouette on the image plane to get the posed and shaped body model. They also show that even with the 2D keypoints and silhouette as input, the subject's pose and shape cannot be uniquely determined. Multiple combinations of body shapes and poses can explain the same silhouette and the keypoints (see Fig. 1.6). Bogo et al. (1) automate the process of obtaining the 2D keypoints using deep neural-network based methods. They also address the problem of self-interpenetration of the estimated body parts, which happens because of no constraint on the body pose (see Fig. 1.5). They approximate the human body using 3D capsules, which enables them to efficiently compute the self-intersection of the body parts and penalize it. Additionally, they learn a pose prior distribution of the human poses using a mixture of Gaussians and utilize it during the optimization. Pavlakos et al. (20) introduce a variational auto-encoder (VAE) based pose prior network, VPoser, and use it in their fitting method to restrict the resulting pose to be a valid 3D human pose. VPoser is trained using AMASS dataset (3) which is a large collection of 3D human motions in SMPL format.

Modern methods use deep neural networks to regress the parameters of a human body model directly from the RGB image (4; 15; 22; 24; 36; 37; 38; 39) or video (26; 40; 41; 42). Training a neural network to predict 3D human pose from an RGB image requires a lot of 2D-to-3D paired data, which is extremely difficult to collect. HMR (4) overcomes this by training a human pose discriminator using a large dataset of 3D human poses. They train a network to regress 3D human pose and weak perspective camera parameters from a tightly cropped image of a person (refer Fig. 1.13a). They use a 2D keypoint re-projection loss along with a discriminator, which helps avoid unnatural poses while satisfying the 2D re-projection constraints. SPIN (5) develops upon HMR and utilizes more in-the-wild images (refer Fig. 1.13b). They propose an auto-corrective optimization step and trains the regressor in the loop. They initialize the 3D human pose using HMR, fit it to the 2D keypoints and use the resultant 3D pose as supervision to further refine the network. Both the HMR and SPIN work on a single image. VIBE (26) extends the method used by HMR and trained a human motion regressor, which takes in a sequence of RGB images and gives the sequence of 3D human poses. They use a large amount of 3D human mocap data to train their discriminator, along with the generator network. Choi et al. (42) propose a neural network architecture combining the temporal information from the past and future frames, which results in smooth and temporally consistent 3D human motion.

All the above methods estimate the human pose/motion relative to the camera or relative to some local coordinate system on the human body. Recent methods (8; 9; 14; 43) try to estimate the human motion in a world reference frame using a monocular video. HuMoR (8) learns the distribution of human motion transitions, including global motion, and uses it to fit the SMPL model to the 2D keypoints on a monocular video from



(a)



(b)

Figure 1.13: (a) HMR regressor takes in the cropped and scaled image I of a person to give the SMPL pose and shape parameters. The Discriminator D is trained using AMASS dataset (3) (image taken from (4)). (b) Auto-corrective optimization step for improved supervision to train the HPS regressor (image taken from (5)).

a static camera. However, encoding only the motion transitions is very sensitive to noisy observations (2D keypoints) and even a little noise can lead to unrealistic human motion estimates. Yuan et al. (9) and Park et al. (44) try to address the challenging problem of motion estimation using a single moving camera. Both of these methods use a monocular pose regressor to get the articulated pose of the person and then either a motion infiller (9) network or an inverse kinematic solver along with the contact points (44) to predict the global motion. Since they do not use the RGB images after the first step and the pose regressor methods are not accurate enough, the global motion based on these inaccurate articulated poses is also not of good quality. Furthermore, they cannot estimate the camera motion due to the bad quality of global motion estimates. Ye et al. (14) and Henning et al. (45) use the SLAM methods to decouple the motion of the camera and the person from the monocular RGB video. However, these methods are dependent on the performance of the SLAM methods, which gets worse when the background in the video does not have enough texture. Sun et al. (43) introduce a regression-based method to estimate the camera motion and the motion of multiple people in the scene from a single video. Since this is a regression-based method, it is faster than the previous optimization-based methods, however, the quality of the reconstructed motion is not very good. While getting the global human and camera motion using only a single moving RGB camera is an extremely challenging problem, our method SmartMocap (Chapter 4) shows that adding just one static camera allows us to estimate the motion of the human and cameras in the world frame.

1.4.2 Multi-view methods

Most of the existing markerless human mocap methods utilize videos from multiple calibrated and time-synchronized cameras. They first detect 2D features (keypoints, silhouette, etc.) on the image plane and then use the camera calibration parameters to either project them into the 3D space (13; 46; 47; 48; 49; 50; 51; 52; 53) or fitting a 3D human body to the 2D features (12; 54; 55; 56; 57; 58; 59).

Early multi-view methods use the Pictorial Structure Model (PSM) (60) to represent the human skeleton. Amin et al. (61) are the first ones to use PSM in multi-view scenarios. They estimate 2D PSM for each view and then triangulate to get the 3D pose. Pavlakos et al.(47) use a similar approach to first get the 3D human pose, then they use this data to train a model to estimate 3D human pose from a single image. They further fine-tune their 2D keypoint estimation network in an unsupervised manner, using only multi-view constraints. Similarly, Rhodin et al. (48) use the multi-view estimates to improve the performance of the monocular 3D pose estimator. Since the PSMs require discretization of the 3D space, they are limited by the speed vs accuracy tradeoff. Discretizing the 3D space in finer bins leads to better accuracy but makes the method slow. To overcome this, Qiu et al. (49) introduce recursive PSM which first estimates a coarse 3D position of a body joint, and then recursively discretizes the 3D space to finer bins to get a more accurate position. Tome et al. (62) train a lightweight neural network that iteratively refines the 3D pose of the person by projecting the 3D joints back to all the views. While these methods try to use the 2D heatmaps as a confidence score, Hua et al. (63) extract the 2D keypoint from the heatmaps, triangulate them to get a 3D skeleton and then refine them using a graph convolutional network to get the final 3D skeleton. He et al. (50) argue that computing 2D features without any 3D context and fusing them to get the 3D pose is suboptimal. That is why they introduce epipolar transformer networks that can leverage the epipolar geometry and fuse a 2D feature in one view to a better matching feature in another view along the epipolar line. Zhang et al. (51) avoid explicit modelling of the feature fusion across multiple views, thus employing transformer networks to learn it. Iskakov et al. (13) introduce a simple yet effective approach where they directly triangulate 2D heatmaps to get a 3D heatmap which is converted to 3D joints by a Voxel-to-voxel network (64). Tu et al. (52) propose a similar approach but extend it to multiple people. Since these approaches are learned from data, they cannot generalize to new datasets where the camera configuration is changed. Bartol et al.(65) introduce an approach which generalizes to any multi-view scenario, thus, removing the need for training for every dataset or camera configuration. For every time instant, they generate multiple 3D human pose hypotheses by triangulating 2D keypoints from randomly selected views. Finally, they train a network to give a score to a pose hypothesis without needing to know the spatial camera arrangement.

A common limitation of the above-discussed methods is that they estimate the 3D skeleton of the subject per frame, which is why they cannot enforce constraints such as bone length consistency. Chen et al. (53) propose a closed-form solution for triangulating 2D keypoints with known camera calibration and constraints on bone-length consistency. Because of a closed-form solution, their method execution is very fast and capable of running in real-time on a CPU. Using explicit constraints on the skeleton does not guarantee a unique skeleton for the sequence. A better way is to use a parametric human body model such as SMPL. Huang et al. (12) extend the method from (66) and fit the SMPL model (19) to the 2D keypoints and silhouettes on multiple views simultaneously. For temporally coherent motion estimation, they use DCT-based motion prior in a separate step after the per-frame fitting. Similar to SPIN (22), Li et al. (54) trains a regressor for human pose and shape estimation from multiple views. First, they use the regressor to initialize the SMPL parameters, then fit it to the 2D keypoints on all the views and finally use the results to train the regressor.

Getting the full 3D pose of the subject (including the global pose) needs calibrated cameras. Since calibrating the cameras in a separate step is not always possible (e.g. in-the-wild scenarios), (67; 68; 69; 70) utilize the human body to calibrate the static cameras. Jiang et al.(68) build upon (13) and extend the learnable triangulation to work for uncalibrated static cameras. In place of volumetric triangulation proposed by (13), they sample camera pose from a distribution, triangulate 2D heatmaps using them and then update the camera pose distribution based on the cross-view reprojection error. Takahashi et al. (67) use the skeleton representation but put explicit constraints on the body model, such as fixed bone lengths across the time and smooth motion constraints. They roughly initialize the camera parameters using structure-from-motion and triangulate the 2D keypoints to get a rough initialization of the 3D joints. Then they optimize all the parameters using the bundle adjustment approach for the full sequence. In contrast to other approaches, they use unsynchronized cameras and estimate the time shift along with the other parameters. Dong et al. (70) utilize the person's pose to synchronize the cameras. They take multiple videos of the same action being performed at different time instances, for example, serving a tennis ball in different matches. They use an off-the-shelf 3D human pose estimator to get the articulated 3D pose of the subject and use it for camera synchronization. They further improve the pose by optimizing the 2D projection error. Huang et al. (69) propose to fit the SMPL body model to the multi-view videos of multiple people interacting. They do not assume calibrated cameras but estimate the poses of the cameras relative to the first camera. They learn a human motion prior and use it during the optimization procedure. The motion prior helps in correcting the errors due to the noisy 2D keypoints. While the noise in the 2D keypoints results in a bad 3D pose of the person, it can result in a huge error in the camera poses.

The static cameras cannot actively change the viewpoint to get a better view of the person, therefore, Elhayek et al. (71) use moving cameras along with the static cameras as their mocap setup. They use sound to synchronize the video stream across multiple cameras. They require a 3D template of the subject's body and some user interaction to get the initial camera calibration using features and bundle adjustment. Then they jointly estimate the body pose and camera calibration parameters. However, all the above methods for uncalibrated cameras estimate the human motion relative to one static camera, which should be calibrated relative to the world such that the estimated human motion can be transformed relative to the world reference frame. Calibrating cameras is particularly hard if all the cameras are moving. Hasler et al. (72) use multiple handheld smartphone cameras and calibrate them relative to a static background using a structure-from-motion (SFM) method. However, such calibration is not reliable and only works for a static background with suitable texture. It may fail when there are moving objects in the background or when the texture is not sufficient for SFM.

1.4.3 Aerial mocap methods

While multiple cameras help in handling occlusions and result in a better mocap, static cameras cannot capture long-range motions. For such motions, cameras should be moving to keep the person in their field of view. Multiple cameras moving around the person is a promising solution for unrestricted and flexible motion capture. Therefore, Multiple works place the cameras on UAVs (73; 74; 75; 76; 77; 78; 79) to capture long-range human motions.

However, early works are primarily restricted to indoor lab environments. Flycon (73) and Drocap (74) use a single UAV with a camera. Flycon needs the subject to wear LED markers leveraging mature IR-based mocap algorithms. Drocap is a markerless system, however, it uses a high-latency fitting-based method to compute the subject's 3D skeleton and the UAV poses for the complete sequence. Another method presented in (76) uses a depth camera along with an RGB camera, which results in lower accuracy in outdoor daylight conditions. Other methods (77; 78; 79) focus on motion planning of the motion planning of the UAVs and use simple triangulation method to estimate the 3D pose of the subject. Flycap (75) uses RGB-D cameras on multiple UAVs in indoor environments to reconstruct a 3D point cloud over time. It requires a template scanning step in which the subject must be static while a UAV scans them. Similarly, ActiveMoCap (79) needs to run a calibration mode for each subject, where it estimates the shape (bone lengths) of the subject. Since all these methods need some kind of subject preparation for every mocap session, we address this problem and present multiple methods to capture the motion of a human subject without needing any preparation. Our method AirCap-Pose-Estimator (Chapter 2) uses multiple autonomously flying UAVs to collect data and process it offline to compute the mocap. Our method AirPose (3), does not need camera extrinsics to estimate the subject's poses onboard in real-time, which can be used to plan the motion of the UAVs. Our method SmartMocap (Chapter 4) estimates the motion of the subject and the cameras relative to the ground plane, and it can be used with any type of RGB cameras (aerial, smartphone, etc.).

1.5 Thesis organization

This thesis is organized as follows:

In Chapter 2, we introduce AirCap-Pose-Estimator. First, we present the hardware setup of the UAVs and the online and offline phases of our method. Then we present our experiments with outdoor data collection, including the data for the reference method. Finally, we show the quantitative and qualitative results of our AirCap-Pose-Estimator on the collected data.

In Chapter 3, we introduce AirPose. First, we present the motivation for AirPose, a baseline method, and our insights for AirPose and AirPose$^+$. Then we discuss our training pipeline, synthetic data generation, real-world data collection and evaluation methods. Finally, we present our experiments and discuss the results on different datasets.

In Chapter 4, we introduce SmartMocap. First, we discuss human motion prior network, its training data and training pipeline. Then we present our optimization problem formulation to get the human and camera motions. Finally, we present the results on multiple datasets, evaluation metrics, and comparison with a reference method.

In Chapter 5, we conclude our thesis and discuss the future directions of our work. We summarize each of our individual methods, along with their limitations and motivation for the next method. Finally, we discuss possible future directions of our work, briefly discussing an ideal markerless mocap system.

The following works (80; 81; 82) form part of this thesis.

- N. Saini, E. Price, R. Tallamraju, R. Enficiaud, R. Ludwig, I. Martinovic, A. Ahmad, and M. Black. Markerless outdoor human motion capture using multiple autonomous micro aerial vehicles. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 823–832, 2019

- N. Saini, E. Bonetto, E. Price, A. Ahmad, and M. J. Black. Airpose: Multi-view fusion network for aerial 3d human pose and shape estimation. *IEEE Robotics and Automation Letters*, 7(2):4805–4812, 2022
- N. Saini, C.-H. P. Huang, M. J. Black, and A. Ahmad. Smartmocap: Joint estimation of human and camera motion using uncalibrated rgb cameras. *IEEE Robotics and Automation Letters*, 8(6):3206–3213, 2023

The following work (83) was done during my PhD, but does not form a part of this thesis.

- R. Tallamraju, N. Saini, E. Bonetto, M. Pabst, Y. T. Liu, M. J. Black, and A. Ahmad. Aircaprl: Autonomous aerial human motion capture using deep reinforcement learning. *IEEE Robotics and Automation Letters*, 5(4):6678–6685, 2020

Chapter 2

AirCap Pose Estimator

Our goal is to capture a freely moving human outdoors, running, jumping, etc., with no markers and full freedom of movement. To do so, we propose to capture human movement using RGB cameras mounted on multiple unmanned aerial vehicles (UAVs); that is, a flying motion capture system. As discussed in Sec. 1.4.3, this idea is not new, but no previous methods achieve our goal. They are either restricted to indoors (75; 76), use simple triangulation of 3D keypoints (77; 78; 79) or require specialized markers to be worn on the body (73).

This problem has remained unsolved because it combines several technologies, each of which, on their own, is a major technical challenge. First, we need multiple aerial vehicles that can coordinate, self-locate, identify a subject, hold them in view, avoid obstacles, etc. Second, motion capture requires calibrated cameras, where the extrinsic parameters are known with high accuracy. Achieving this outdoors with moving vehicles in real settings is a significant technical challenge. Third, while deep learning methods have made 2D human joint detection reliable, accurate 3D human pose from images remains an active research problem.

To address these issues, we present an outdoor markerless human motion capture system using a team of UAVs, called AirCap-Pose-Estimator. Each UAV consists of only an RGB camera to detect and track the subject. Each UAV also has an on-board IMU, GPS and barometer used only for its self-pose estimation in the global (GPS) coordinate frame. Note that these sensors alone are not sufficient to achieve the accuracy of camera calibration necessary for human mocap. Consequently, like (84), we formulate the calibration problem together with the human pose estimation problem. The overall functioning of our motion capture system is split into two phases, namely, i) an online data acquisition phase using autonomous UAVs, and ii) an offline pose and shape estimation phase. This is summarized in Fig. 2.1.

During the online data acquisition phase, the UAVs cooperatively detect and track the 3D position of a subject using the approach presented in (85) and follow them in formation using the perception-driven method from (28). This formation allows the UAVs to i) keep the subject in their camera's field of view and centered on the image plane, ii) maintain a threshold distance from the subject, and iii) not collide with each other or any other static obstacle in the environment. The data acquired in this phase consists of

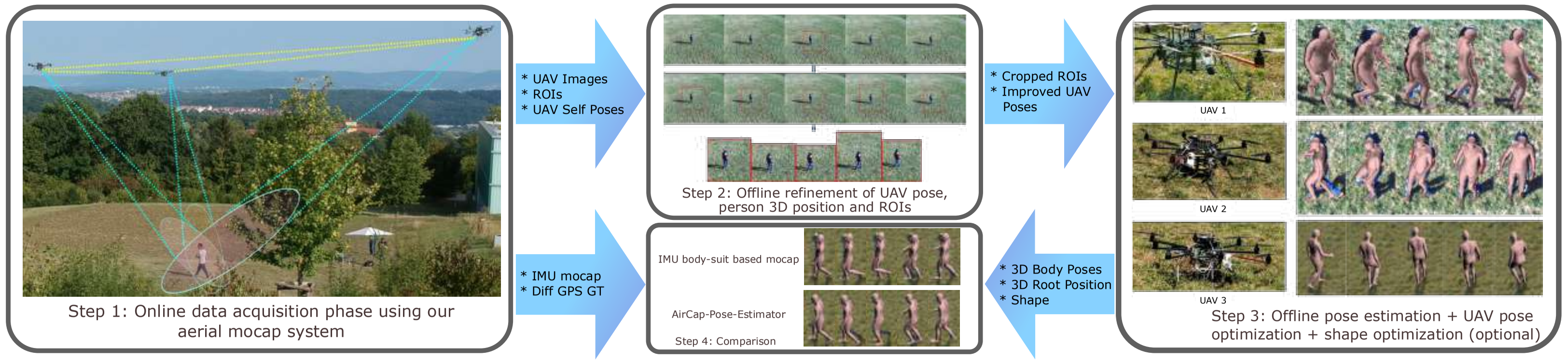

Step 1: Online data acquisition phase using our aerial mocap system
* UAV Images
* ROIs
* UAV Self Poses
* IMU mocap
* Diff GPS GT
Step 2: Offline refinement of UAV pose, person 3D position and ROIs
IMU body-suit based mocap
AirCap-Pose-Estimator
Step 4: Comparison
* Cropped ROIs
* Improved UAV Poses
* 3D Body Poses
* 3D Root Position
* Shape
UAV 1
UAV 2
UAV 3
Step 3: Offline pose estimation + UAV pose optimization + shape optimization (optional)

Figure 2.1: AirCap-Pose-Estimator Overview. Step 1 is the online phase, while Steps 2–4 are parts of the offline phase, as described in the text.

images captured by all UAVs and their camera extrinsic and intrinsic parameters. The data also contains the person's approximate 3D location in the world coordinates (not the detailed pose). Note that the camera extrinsics from this phase are approximate and, as we will see, not sufficiently accurate for human mocap.

In the second phase, which is offline, human pose and shape are estimated using only the acquired RGB images and the UAV's self-localization poses (the camera extrinsics). Our approach relies on 2D joint detections in each camera; current methods like AlphaPose (86) and OpenPose (16; 87) are quite accurate even with aerial imagery. To fuse these 2D detections into a 3D pose estimate, we formulate an objective function in which we simultaneously solve for body shape, 3D pose, and 3D camera positions. We use the SMPL body model (discussd in 1.2.3) to fuse the noisy estimates. SMPL captures the shape of the human body and this constrains the possible solutions. We project the joints of SMPL onto each of the images (using the estimated camera parameters) and compute the error (robustly) between the predictions and the observed 2D detections. Since the 2D pose detections may be noisy, we regularize the 3D fitting using VPoser (see Sec. 1.3.1). We solve for camera parameters jointly and constrain them to be similar to those estimated by the UAVs.

In summary, AirCap-Pose-Estimator addresses the following key challenges: (1) Detection and tracking of a person by multiple UAVs fully autonomously. (2) Estimation of the camera extrinsics and the 3D location of the person. (3) Fitting a 3D body model robustly to 2D joint detections from multiple flying cameras. (4) We show, for the first time, that it is possible to capture 3D human movement fully autonomously from aerial vehicles. (5) We compare our 3D poses with reference data computed from a multi-IMU suit and the SIP method for pose estimation (29). While the accuracy is not yet on par with commercial marker-based systems, this is a practical step towards a solution that addresses each piece of technology in an integrated whole. Our code and dataset are available at `https://github.com/robot-perception-group/Aircap_Pose_Estimator`. A video explaining our method and results is available at `https://tinyurl.com/aircapposeest`.

2.1 AirCap-Pose-Estimator

We first describe our motion capture hardware and the online phase. Then we discuss our system pipeline in detail by introducing mathematical symbols and notation, followed by the algorithm. The pipeline consists of four steps.

2.1.1 Step 1 : mocap system setup and online data acquisition phase

Step 1 in Fig. 2.1 shows our UAV-based outdoor motion capture system tracking and following a person. It consists of a team of self-designed 8-rotor UAVs (see Step 3 in Fig. 2.1 inset). Each UAV is equipped with a 2MP HD camera, a computer with an Intel i7 processor, an NVIDIA Jetson TX1 embedded GPU, and an OpenPilot Revolution[1] flight controller board. We use the flight controller's position and yaw controller, as well as its GPS and IMU-based self-pose (position and orientation) estimation functionalities.

To detect, track and follow the person, we use a perception-driven formation approach (28; 85). Each copter runs a single shot detector (SSD) multibox detector (88) on the images acquired by its camera, using its on-board GPU to detect the person's bounding box on the image frames. A detection rate of $\sim$4 Hz is achieved during the online acquisition. The UAVs then share the person's 2D image bounding box positions and their 3D self-pose estimates wirelessly between each other. Subsequently, using a cooperative detection and tracking (CDT) filter (85) that runs on-board each UAV's CPU, they estimate the 3D position of the person's center of mass in a consistent world frame (GPS-frame). Using this method, the UAVs also improve their 3D self-pose localization. One key feature of the CDT filter is that it allows the detector to focus on the most informative region of interest (ROI) in future image frames, thereby making it computationally efficient. Note that even though the detections are obtained at $\sim$4 Hz, the CDT filter runs at $\sim$30 Hz, alternating between the standard prediction and update steps, except that the updates happen at a lower frequency.

In the online phase, the goal is to keep the person in the field of view and centered in each UAV's camera. Additional constraints include maintaining threshold distances to the other UAVs and static obstacles. To this end, each UAV runs a model predictive control (MPC)-based formation controller (28) on its on-board CPU. The MPC's objective is to maintain a threshold distance to the subject while adhering to the aforementioned formation constraints. Orienting the UAV towards the subject is achieved using an additional yaw controller (separate from the MPC). Further details regarding the CDT tracker and formation controller can be obtained from (85) and (28), respectively.

During the online phase, all UAVs save images on-board at $\sim$40 Hz and their self-pose estimates at $\sim$100 Hz. As the camera is rigidly mounted on each UAV, the extrinsics of the camera are obtained using a fixed and known transformation from the UAV's self-pose (position and orientation) in the world frame.

2.1.2 Step 2 : 2D region of interest and UAV self pose refinement

In this step, we run the CDT algorithm of Step 1 offline to improve the subject's tracked position estimate and each UAV's self pose estimates. The SSD Multibox detector runs on every frame in Step 2. The CDT filter leverages these every-frame observations to obtain the ROIs for every image and to improve the UAV self-pose estimates.

[1]OpenPilot: http://www.librepilot.org/site/index.html

2.1.3 Step 3 : Offline pose estimation

The rest of this section discusses Step 3 in which the person's pose and shape, as a function of time, is estimated using the data acquired in the online phase (Step 1) and refined in Step 2. Note that Step 4 concerns comparison with ground truth and is, therefore, discussed in the next section with experiments and results.

Preliminaries

Consider a system with C moving cameras. The intrinsic parameters of each camera are fixed. Since the cameras are moving in the world frame, their extrinsic parameters (rotation vector, translation vector) are changing over time. The rotation and the position vectors of camera c at any time instant t are represented as $r_{c,t} \in \mathbb{R}^3$ and, $p_{c,t} \in \mathbb{R}^3$ respectively. We use the SMPL body model to represent the pose and shape of the subject (see Sec. 1.2.3).

2D joint detections on the collected images can be highly noisy. We use multiple 2D joint detectors for robustness. Say we use D detectors and each detector gives the positions of N joints on the camera plane. The position of the n^{th} joint given by the d^{th} detector on the c^{th} camera plane at time instant t is represented as $j_{c,n,d,t} \in \mathbb{R}^2$. The detector also gives a confidence value in terms of probability for each detected joint. It is represented as $w_{c,n,d,t} \in \mathbb{R}$.

The SMPL pose vector θ is the collection of all the joint angles. However, human poses do not span the entire angle space. To restrict θ to the natural pose space, we use the latent space of VPoser. VPoser is a variational autoencoder (VAE) with encoder ($\mathcal{V}_E$) and decoder ($\mathcal{V}_D$). The new parameterization of the human pose is represented as $v \in \mathbb{R}^3 2$ in the latent space of VPoser with a Normal distribution. VPoser is trained on more than 1 million poses of multiple subjects and is capable of producing novel, realistic human poses. For more details on the data and actual training procedure, refer to (89). VPoser decoder $\mathcal{V}_D$ provides a mapping from the latent variable v to full pose variable θ given as

$$\theta = \mathcal{V}_D(v). \tag{2.1}$$

We exploit the known distribution of the latent variable as a prior in our optimization objective, by keeping its values close to the mean of the Normal distribution. This translates to a simple L2 norm on the new parameterization.

Algorithm

We use the detected 2D joints and intrinsic parameters to optimize the body model parameters along with camera extrinsics. Camera extrinsics are initialized with the refined estimates obtained in Sec. 2.1.2. This is done independently for each time step.

Per-frame fitting We minimize a cost function at each time step t, which can be decomposed into the following components:

$$\begin{aligned} E(r_{1,t}\cdots r_{c,t}, p_{1,t}\cdots p_{c,t}, v_t, \beta_t) = \\ E_{2D} + \lambda_{r,p}E_{r,p} + \lambda_v E_v + \lambda_\beta E_\beta, \end{aligned} \tag{2.2}$$

where $\lambda_{r,p}$, λ_z and λ_β are weights of the corresponding components.

The first term ensures that the 2D projection of the model's 3D joints remains close to the observed 2D joints. It is given as

$$\begin{aligned} & E_{2D}(v_t, \beta_t, r_{c,t}, p_{c,t}) = \\ & \sum_{c,n,d} w_{c,n,d,t}\rho_{\sigma_1}\left(\left\|\Pi\big(r_{c,t}, p_{c,t}, \mathcal{J}_n(\mathcal{V}_D(v_t), \beta_t)\big) - j_{c,n,d,t}\right\| \right), \end{aligned} \tag{2.3}$$

where $\mathcal{J}_n$ is the joint regressor function that gives the n^{th} joint position given the SMPL pose and shape parameters. Π is the projection function that projects the 3D point on the image plane, given camera parameters. ρ_{σ_1} is the Geman-McClure robust penalty function with a fixed parameter σ_1, written as

$$\rho_{\sigma_1}(e) = \frac{e^2}{e^2 + \sigma_1^2}. \tag{2.4}$$

As explained in Sec. 2.1.1, camera extrinsic parameters are obtained directly from the UAV's self-pose data saved during the flights made by the UAV formation. The self-pose estimates of the UAVs are prone to various sources of error, e.g., GPS and IMU drift and changing prevalent wind speeds causing fluctuations in the barometer measurements. This causes the camera extrinsic parameters to be noisy. Hence, we also optimize for the camera extrinsic parameters, with the objective of keeping them close to the values estimated online by the UAVs, by including another cost term,

$$E_{r,p} = \rho_{\sigma_2}(r_{c,t} - \tilde{r}_{c,t}) + \rho_{\sigma_3}(p_{c,t} - \tilde{p}_{c,t}), \tag{2.5}$$

where $\tilde{r}_{c,t}$ and $\tilde{p}_{c,t}$ are the rotation and position vectors of the camera c at any time t estimated online by the UAVs during the data acquisition phase.

E_v is a regularization term on the latent pose parameter v given as

$$E_v = \|v\|. \tag{2.6}$$

β is a vector of the 10 most significant eigen shapes of SMPL, which we regularize with E_β as

$$E_\beta = \|\beta\|. \tag{2.7}$$

2.2 Experiments and Results

2.2.1 Data Acquisition

Using our UAV-based motion capture system described in Sec. 2.1.1, we performed a data collection formation flight using 3 UAVs. Our on-board formation controller, UAV self-pose and the person's (3D position, not joint poses) state estimator, etc., are implemented as Robot Operating System (ROS) nodes, which makes it easy for UAVs to communicate with each other using standard message types. The UAV formation constraints of altitude and horizontal distance from the subject are set to 8m. The value is relatively high due to safety considerations. During the formation flight, the subject is requested to walk on a grassy field at slow to moderate speeds and later perform random motion sequences, such as jumping jacks, bending forward/backward, swaying arms, etc.

2.2.2 Dataset

All images and camera extrinsic and intrinsic parameters are saved on-board each UAV as ROS messages in a rosbag file. Each message has a Unix timestamp denoting the time of its acquisition. We receive images from each camera at approx. 30-40 frames per second (FPS). Even though both UAV cameras have the same frame rates, they are not synchronized. Meaning, they do not necessarily capture image frames simultaneously. For any image from the first UAV's camera, there might not exist an image from the other UAV's camera at the same instant. Also, as camera parameters are available at a much higher frequency than images, for each image in the system, camera intrinsic and extrinsic parameters are available. Later, we extract data from the saved bagfile, refine them and use them to estimate the shape and pose of the subject using the method described in Sec. 2.1.3.

2.2.3 Reference Data

We obtain reference (ref) data to evaluate our reconstructions from two different systems, i) a commercially available IMU mocap system (Xsens) (90) and ii) a pair of differential GPS modules. The IMU system is used to obtain reference data for body pose relative to the root joint. For reference SMPL parameters, we use a state-of-the-art IMU mocap method Sparse Inertial Poser (SIP) (29). It uses raw data from Xsens and gives SMPL parameters. However, the global root joint position and orientation from SIP are not reliable for reference comparison. To solve this issue, we use a pair of differential GPS modules, each one attached to the shoulder of the subject, to get the position of the root joint in the global coordinate system. The reference global root orientation remains unestimated as it is not directly measurable with these two systems.

The IMU data is collected using 17 sensors on the subject's body. These sensors measure data at the rate of 60Hz. The measurements do not have the information about the exact time they are recorded. They are sent to the base station using wireless communication. At the time of arrival, they are assigned the frame number (different from our system). Using SIP, we get SMPL parameters for each of these frames. We manually align this sequence to our system by matching a characteristic motion sequence. This way we get a Unix timestamp for one frame, and assuming the frame rate to be 60Hz, we get a timestamp for every frame. We re-sample the frames that are closest to our image data sequence and compare them for all the error calculations.

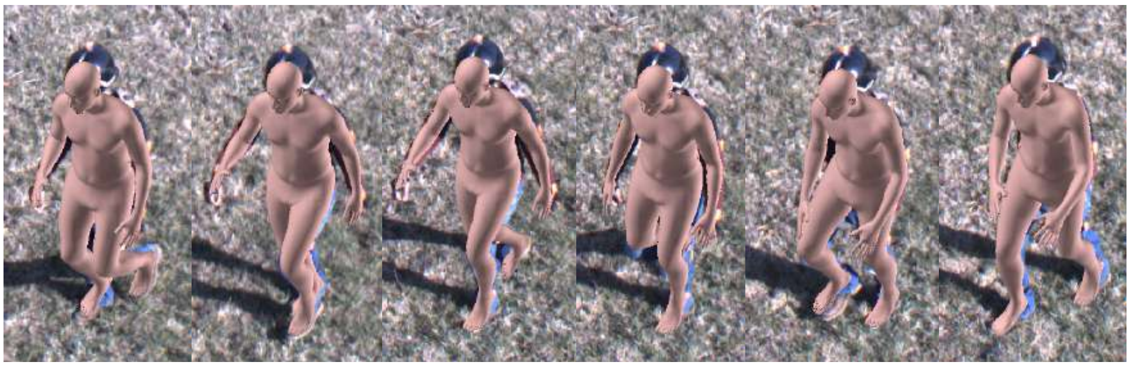
2.2.4 Implementation

Using the approach in (85), the UAVs autonomously maintain a formation around the person while following him/her and keeping him/her centered in their camera's field of view. During the formation flights, the UAVs detect the person in their camera image using single shot detector (SSD) multibox (88) and estimate his/her 3D world position (not the joint pose) and uncertainty associated, in order to maintain the formation. This also results in a cropped region of interest (ROI) which has the highest likelihood of having the person inside it. For every image, the UAVs also saves this corresponding ROI. The ROI data and UAV self pose estimates are then refined offline and saved. We crop the full images based on the provided ROIs and apply multiple joint detectors, each producing a set of 2D joints estimates. If the ROI goes outside the camera frame, we take the full image for 2D joint detection.

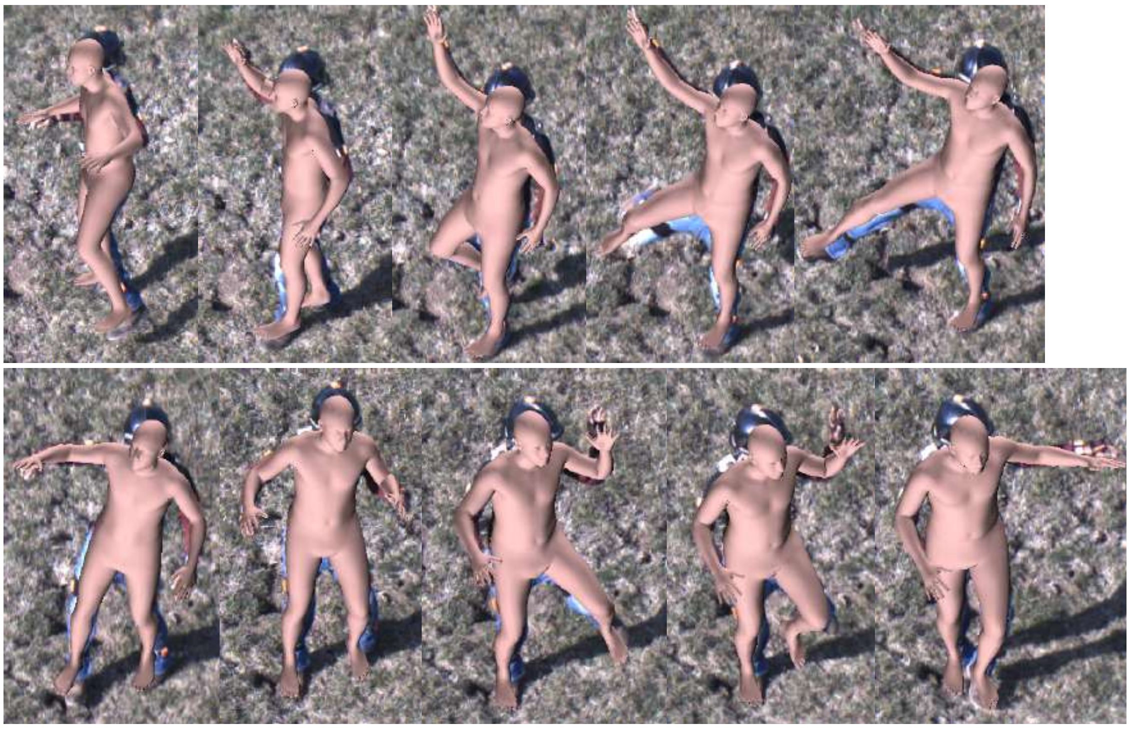
We then use two state-of-the-art 2D joint detectors: Alphapose (86; 91) and OpenPose (16; 87). All the dataset images are processed using these joint estimators, and their output is saved with the same timestamp as that of the image. We use these 2D joints along with the camera extrinsic and intrinsic parameters in our cost function as given in Eq. (2.2). Since the cameras are not synchronized, we use the closest frames in time from all the cameras for per-frame fitting. We use a PyTorch (92) implementation of SMPL to regress from SMPL parameters to 3D joint positions in Eq. (2.3). The total cost is sequentially minimized for each frame to get the optimized value of SMPL pose and camera extrinsic parameters. The value of σ_1 in Eq. (2.3), σ_2 and σ_3 in Eq. (2.5) are 40, 10 and 10 respectively. We found after trial and error that these values work well. After optimizing for a frame, the optimized parameter values are used as initial values for the next frame, except for the camera extrinsics. These are initialized with the ones obtained from Sec. 2.1.2. For optimization, we use the Adam optimizer (93) from PyTorch. The number of iterations for the first frame is 1000 with 0.25 learning rate and 100 with 0.1 learning rate for subsequent frames.

2.2.5 Results and Discussion

First, we compare our reconstructed pose with the reference pose. In this, we zero out the global position and rotation of the reconstructed SMPL and ref SMPL. In Table 2.1, we show the mean error in joint positions (e_{jp}) and mean error in joint angles (e_{ja}).



(a)



(b)

Figure 2.2: Pose and shape estimation results of our approach overlaid on some of the image sequences from one of the UAV's cameras. (a) A walking sequence. (b) A sequence with arbitrary arm and leg movement.

	GT shape		Shape estimation	
Joint	e_{jp}	e_{ja}	e_{jp}	e_{ja}
L_Hip	$0 \pm 0.$	6.73 ± 3.0675	$0 \pm 0.$	6.81 ± 2.9921
L_Knee	0.0767 ± 0.0384	8.60 ± 5.3021	0.0876 ± 0.0417	8.69 ± 5.2789
L_Ankle	0.1629 ± 0.0801	5.49 ± 2.3388	0.1904 ± 0.0909	5.50 ± 2.3235
L_Foot	0.1843 ± 0.0931	9.71 ± 2.1887	0.2157 ± 0.106	9.44 ± 2.1144
R_Hip	$0 \pm 0.$	6.62 ± 3.9518	$0 \pm 0.$	6.60 ± 3.9246
R_Knee	0.0680 ± 0.0457	9.59 ± 6.0983	0.0760 ± 0.0505	9.67 ± 6.1404
R_Ankle	0.1251 ± 0.0988	7.79 ± 3.0918	0.1448 ± 0.1156	7.73 ± 3.0527
R_Foot	0.1461 ± 0.1158	8.10 ± 5.6218	0.1693 ± 0.1358	7.86 ± 5.5202
Spine1	$0 \pm 0.$	5.32 ± 2.1159	$0 \pm 0.$	5.18 ± 2.0373
Spine2	0.0264 ± 0.0103	3.01 ± 1.3283	0.0290 ± 0.0111	2.96 ± 1.2894
Spine3	0.0397 ± 0.0147	1.61 ± 1.1054	0.0439 ± 0.0156	1.59 ± 1.1008
Neck	0.0931 ± 0.0306	6.25 ± 2.6598	0.1068 ± 0.033	6.11 ± 2.6312
Head	0.1237 ± 0.0372	5.13 ± 2.1265	0.1428 ± 0.0403	4.90 ± 2.0727
L_Collar	0.0683 ± 0.0241	4.09 ± 2.9901	0.0771 ± 0.0256	3.82 ± 3.0851
L_Shoulder	0.0779 ± 0.0338	13.15 ± 3.6292	0.0861 ± 0.0355	13.28 ± 3.583
L_Elbow	0.0863 ± 0.056	16.41 ± 8.5772	0.1023 ± 0.0642	16.15 ± 8.4229
L_Wrist	0.1689 ± 0.1231	10.46 ± 3.9089	0.1984 ± 0.1437	10.21 ± 3.8141
L_Hand	0.2045 ± 0.1524	2.34 ± 1.4662	0.2411 ± 0.1767	2.30 ± 1.4598
R_Collar	0.0694 ± 0.0241	5.55 ± 3.5196	0.0777 ± 0.0256	5.23 ± 3.5122
R_Shoulder	0.0919 ± 0.0372	10.96 ± 4.8626	0.0993 ± 0.0382	10.87 ± 4.709
R_Elbow	0.0987 ± 0.068	22.15 ± 7.5988	0.1075 ± 0.0761	21.65 ± 7.5291
R_Wrist	0.1781 ± 0.1375	11.41 ± 5.5971	0.2013 ± 0.1542	11.29 ± 5.6136
R_Hand	0.2134 ± 0.1697	3.19 ± 1.3718	0.2417 ± 0.1903	3.19 ± 1.3296
Pelvis	aligned with the ref			

Table 2.1: Mean error in joint positions (meters) and joint angles (degrees) (using GT body shape vs shape estimation). The pelvis joint is aligned with the ref. The position error for L_Hip, R_Hip and Spine1 becomes 0 because these joints are rigidly connected to the Pelvis.

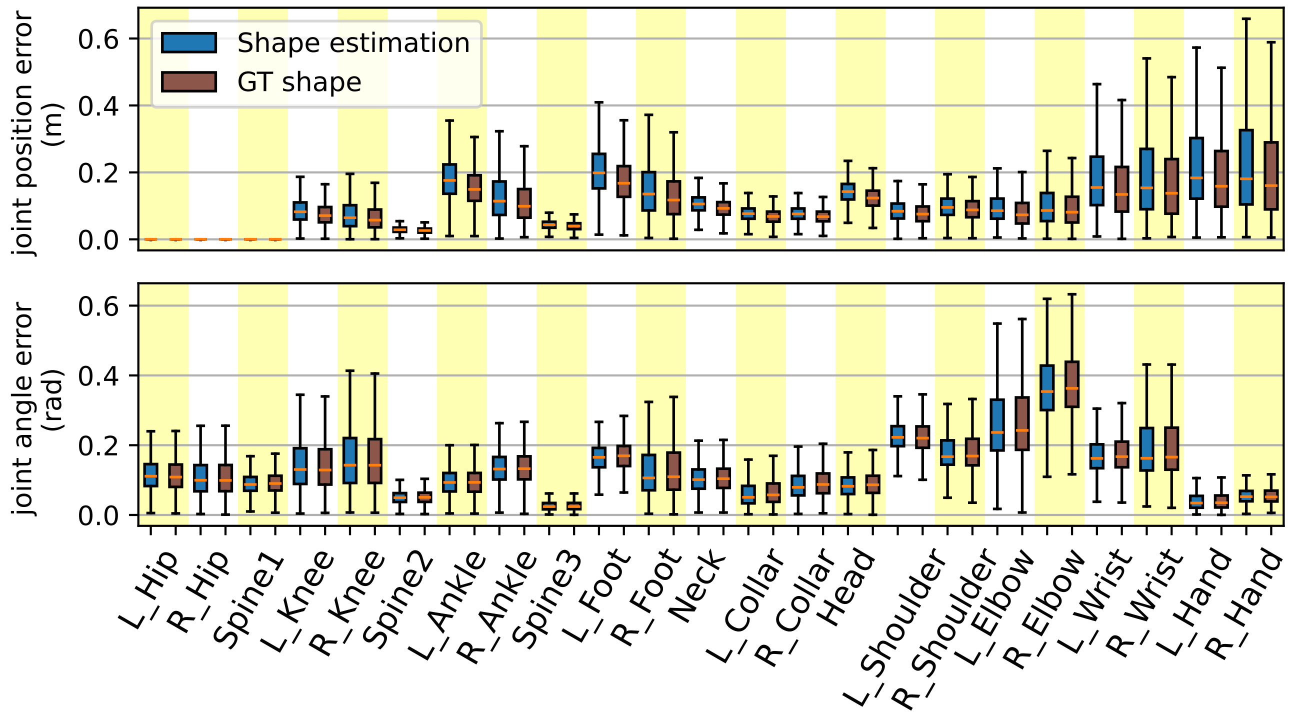


Figure 2.3: Bar plots showing the joint position error and joint angle error when GT shape is used (brown), and when shape is estimated (blue).

e_{jp} is calculated by taking the Euclidean distance between each estimated joint and the corresponding reference joint and then calculating its mean over the whole dataset. e_{ja} is calculated by taking the angle difference between the reconstructed joint angle and the corresponding reference joint angle and taking a mean over time and over all the 3 axes. We show these errors in two cases, 1) using the GT body shape, and 2) with the estimated subject shape. In case 1, we fix the shape to the GT shape obtained by scanning the person. In case 2, the shape is optimized in Step 3 of the system pipeline Eq. (2.2). In Table 2.1, we can see that the joint position estimates get better when GT shape is used, however, there is no significant difference in the joint angle error. This is clearer in the bar plots shown in Fig. 2.3. Since there is no significant change in the joint angles, the articulated pose estimate remains the same in both cases. However, higher joint position error in the case of shape estimation indicates that the estimated shape is not accurate. Since we are fitting the 3D body model to the 2D keypoints, the optimization algorithm can estimate a little bigger body while adjusting the body position a little further away from the camera. This can also happen because of noise in the 2D keypoints.

We see that the error is higher for the joints corresponding to the extremities. This is because we use a pose regularization in Eq. (2.6) which penalizes the distance from the mean pose of VPoser. Since the extreme poses have more variation in the extremity joints, these joints are penalized more. We notice that including shape estimation in Step 3 does not affect the error significantly. However, the per-frame fitting does not make sure that the shape remains the same for the whole sequence. For a better shape estimate, instead of the per-frame fitting, the complete sequence should be optimized with a constant shape for the whole sequence.

Note that our reference method is also not perfect, and its estimates cannot be treated as ground truth. In this procedure, we consistently assume that the IMU system has a frame rate of 60 Hz. However, this is not always the case. The communication delays and failures between the sensors and the base station can cause it to vary. Between 1:33 and 2:45 in the video (see Chapter introduction for video link), we see that sometimes both the meshes go out of sync. One possible reason for this is the variable frame rate of both our system and the IMU system. This also affects our quantitative results adversely. Our calculated error is more than what it actually should be.

For all the further results, we fix the shape to the actual shape of the subject, and all the errors presented are joint position errors in meters.

Pose Evaluation

In Fig. 2.2 we present qualitative results of our pose estimation. Quantitative results are presented in Fig. 2.4, where we show the mean joint position error with time. At every time instant, we compute a metric *NumDet*, which is the total number of views in which the subject is detected. Since we use two detectors, we take the maximum of the two. Thus, *NumDet* is computed as $\max_d \sum_c \mathcal{D}(d,c)$, where $\mathcal{D}(d,c)$ denotes the number

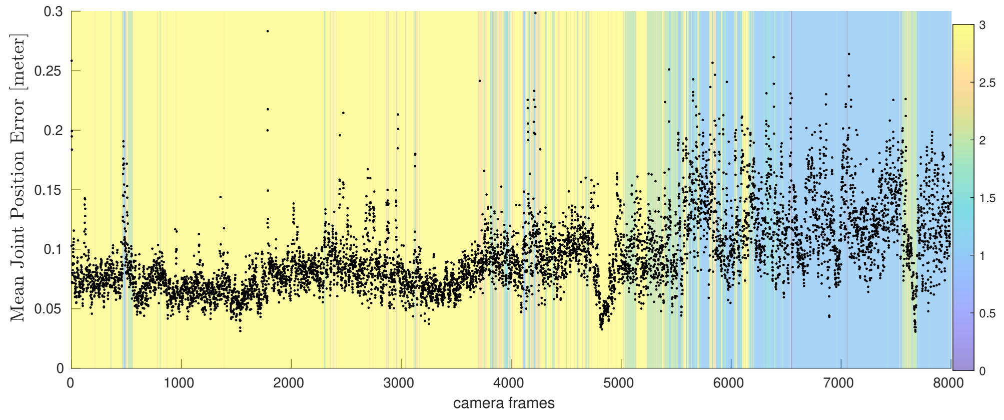


Figure 2.4: Mean joint position error in every frame. Background color denotes the *NumDet* at that frame, calculated as explained in Sec. 2.2.5.

of detections by the detector d for UAV camera c. In Fig. 2.4 we show *NumDet* in the background represented by a color scheme. We can see that the mean joint position error becomes high when the *NumDet* is low, which is expected from our approach. This shows that observations from multiple views add confidence to the estimated pose.

Global Position Evaluation

We use the absolute position from the differential GPS modules mounted on the subject's shoulders. These modules are in a coordinate system that has a constant offset to our system's coordinate system. We find this offset by taking a mean difference between estimated and ref position for the first 200 frames and correct it manually.

All the Steps 1, 2 and 3 provide the person's position. We denote these by τ_{s_1},τ_{s_2} and τ_{s_3} respectively, and the ref root position by τ_{ref}. We show the X, Y and Z components of these in Fig. 2.5. In the motion sequence shown in this experiment, the subject first moves on a zigzag trajectory over a sloped terrain with moderate speed. This can be seen in the ref plots for the initial 4700 frames. Then the subject performs various random body pose sequences like jumping jacks, punching, dancing etc., with small motion in global position. This is reflected in the plots, as there is not much variability in the ref position.

In the inset of Fig. 2.5, we show box plot of the Euclidean error of τ_{s_1}, τ_{s_2}, τ_{s_3} with respect to τ_{ref}. We see that the estimate improves in Step 2 and further in Step 3. If we do not optimize for camera parameters in Step 3 our person position estimates are unchanged from that of Step 2. This indicates that the optimization of camera parameters improves the person's global pose estimate. However, looking at the outliers, we can say that the maximum error can go even higher than the maximum error of Step 1. To analyze this, we look at Fig. 2.6. The background represents the same as in Fig. 2.4. In these plots, we show the signed error of τ_{s_1}, τ_{s_2}, τ_{s_3} with respect to τ_{ref}. Notice there are

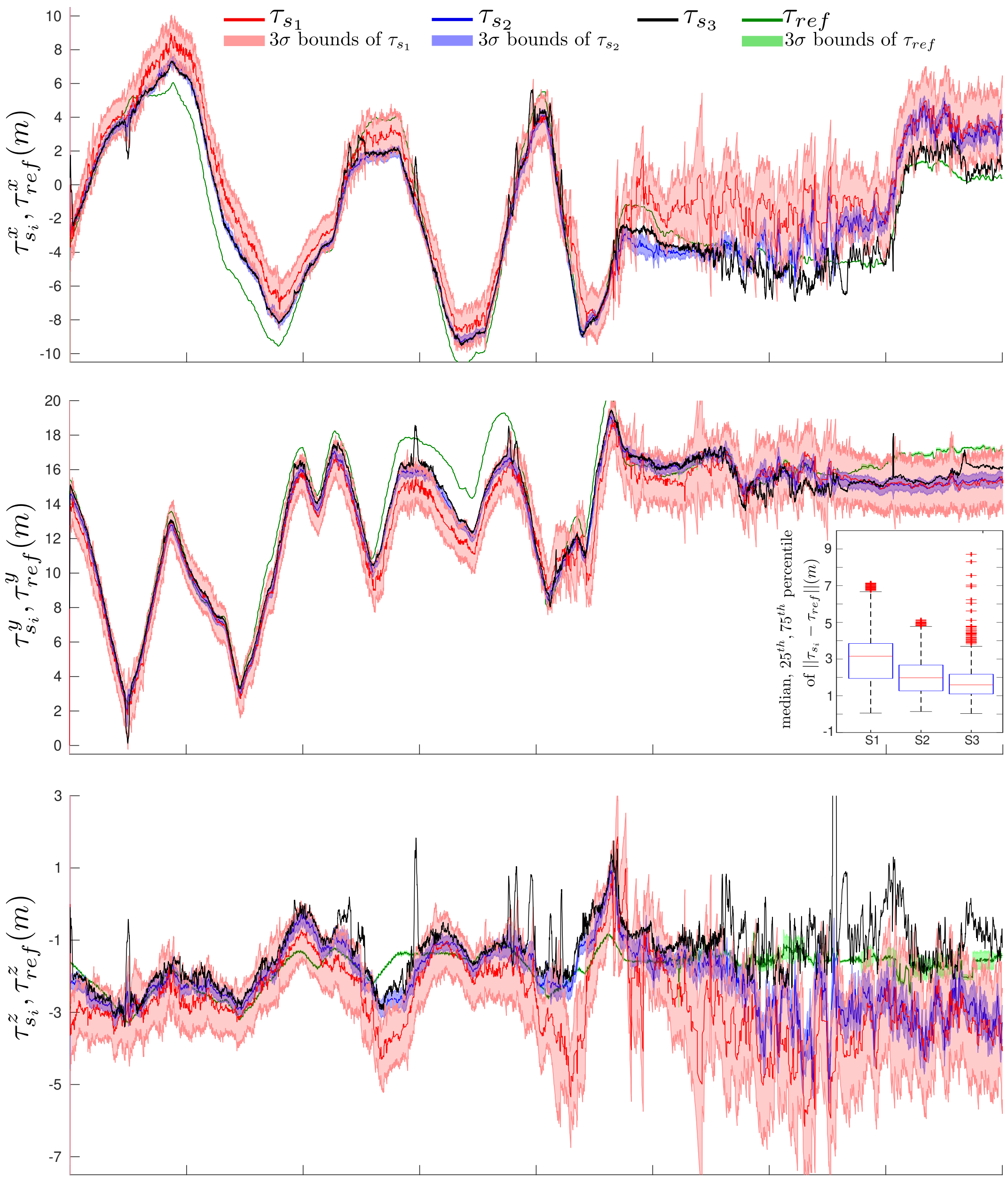


Figure 2.5: These three plots show the root position trajectories from ref, Step 1,2 and 3 in X, Y and Z dimension, respectively. See Sec. 2.2.5 for details.

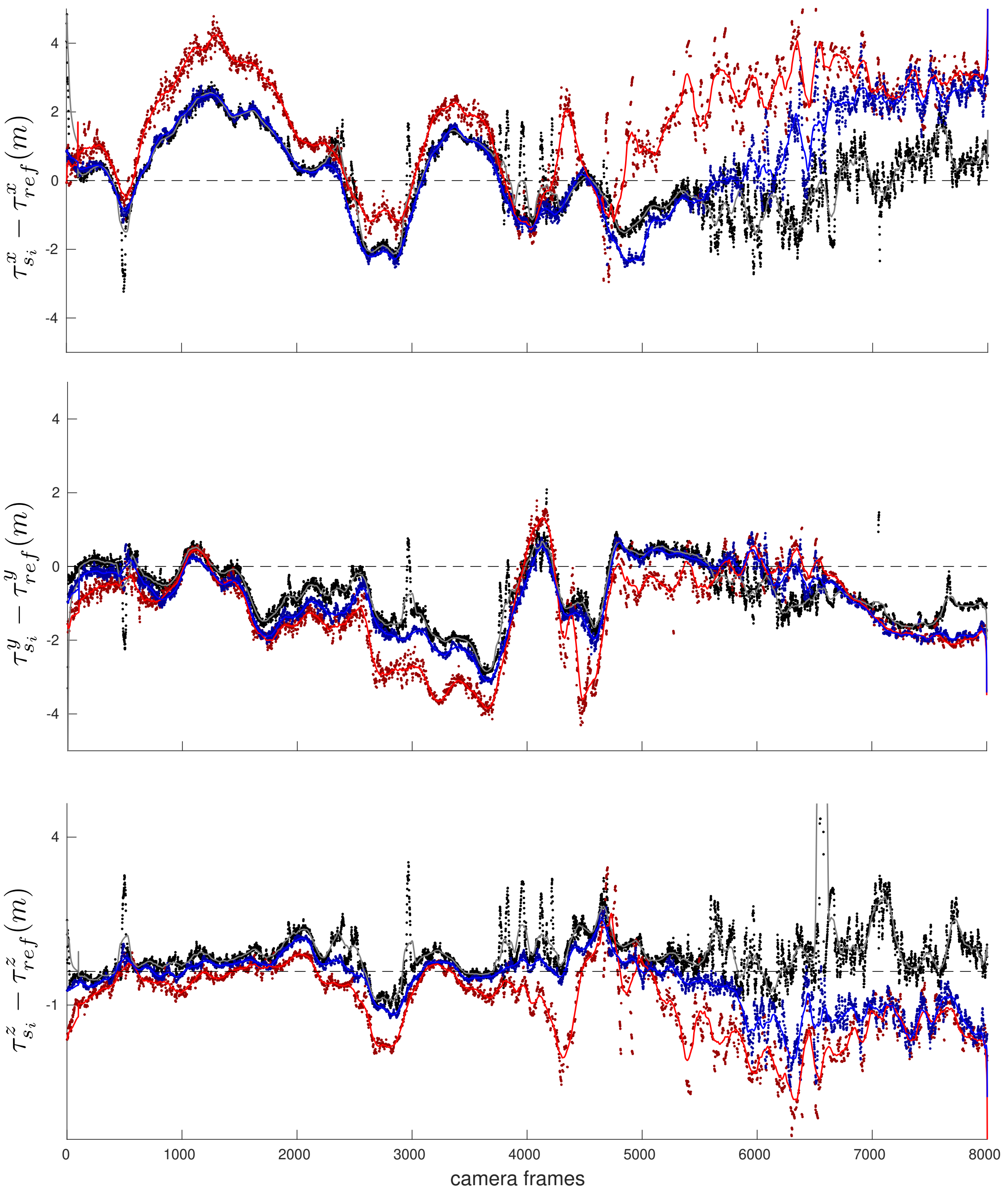


Figure 2.6: These three plots are signed error of the estimated root position from all three steps with respect to the ref. See Sec. 2.2.5 for details.

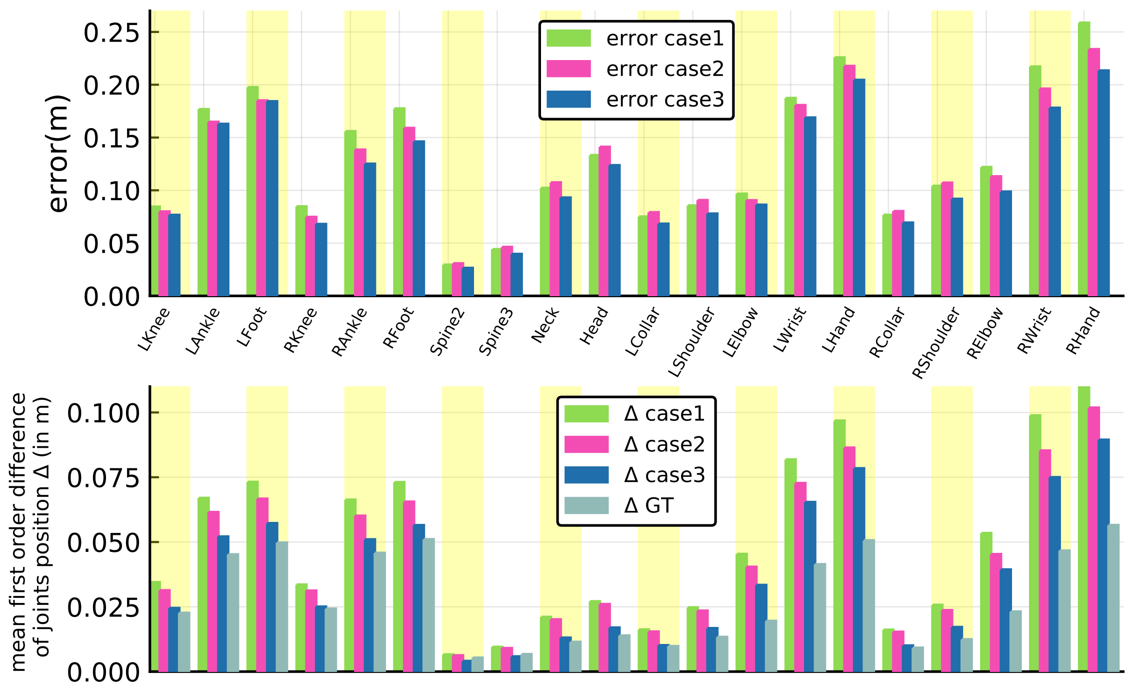


Figure 2.7: Ablation study of our approach.

two error components in these plots. One is a slowly varying component and another is rapidly varying. We show the rapidly varying component by plotting a moving average result over the error. The slow varying error is due to the drift in the UAVs' GPS. Since the person's position estimate is dependent on the UAVs' poses, this drift is reflected in the person's position error. The rapidly moving error is due to the observation error in our 2D estimates. We see that there are sudden jumps in the pose error from Step 3. These correspond to the outliers shown in the inset of the second plot of Fig. 2.5. We see in Fig. 2.6 that these jumps happen when there are fewer detections, which is expected. Since there are fewer detections or no detections in some camera frames, the whole optimization becomes unconstrained. In such a scenario, it becomes highly susceptible to observation errors and the camera pose can be adjusted to fit an erroneous observation.

Ablation study

In Fig. 2.7, we show the advantage of optimizing the camera parameters during human pose estimation. We compare the mean error for each joint in three cases. Case1: we use the camera extrinsics from the online run (i.e. step 1) and do not optimize them during the pose estimation step. Case2: We use the camera extrinsics from the offline run (i.e. step 2) and do not optimize them during the pose estimation step. Case3: we jointly optimize camera extrinsics and pose (our proposed method). For most of the joints, the error in

Case2 is lower than Case1 and lowest for all the joints in Case3. Further, we compare the mean of the first-order difference of joint positions in all the cases with reference to that of the ref. A lower value closer to the ref implies a smoother and more accurate motion estimate. We can see that the value decreases when going from Case1 to 3, getting closer to ref for all the joints. Since we deal with outdoor and unstructured scenarios, getting highly accurate reference (ref) extrinsics of the mobile cameras with minimal on-board computation is extremely difficult. Hence, for our problem, it is important to estimate (and optimize for) both the person's pose and the camera extrinsics, simultaneously.

2.3 Limitations

Our data acquisition system can autonomously plan the motion of the UAVs to follow the person, but its major limitation is the absence of gimbals for the UAV cameras. Since the cameras are rigidly attached to the UAVs, the motion of the UAVs affects their field of view. This results in the person frequently going out of the field of view of the cameras. For example, take a scenario where the person is in the center of the image and the corresponding UAV is hovering in place. Next, the person starts moving away from the UAV and thus moves towards the top part of the image frame. To follow the person, the UAV has to move forward, which results in the pitch down of the UAV and the camera. Since the person is already at the top of the image, this camera pitch results in the person going out of the image frame. A similar scenario happens if the person moves towards the UAV. The UAV has to pitch up to move away, which results in the person going out of the image from the bottom. Since there is an additional constraint of keeping the person centered in the image, these scenarios can result in oscillations if the person moves away or towards the UAV very quickly. In such a case, the camera poses changes very quickly, and it becomes stable again when the UAV's distance from the person is in the desired range. In further chapters, we use UAVs with gimbals and handheld cameras. This allows us to focus on the human pose and shape reconstruction method from the videos, irrespective of the acquisition of those videos.

2.4 Conclusion

In this chapter, we presented AirCap-Pose-Estimator, the first successful demonstration of full-body markerless motion capture from autonomous flying vehicles. AirCap-Pose-Estimator addresses the challenges of i) online image data acquisition of a tracked human subject by multiple fully autonomous UAVs, and ii) human body pose and shape estimation using the acquired image dataset. We show how we leverage state-of-the-art 2D human joint detection methods as noisy sensors and fuse them to obtain consistent 3D estimates of human pose and shape. We show quantitative results by evaluating our reconstructions using reference data. We also show qualitative results by projecting the estimated pose over the acquired images. One of the most important advantages of our method is that it completely removes the need for a subject preparation step, thereby allowing in-the-wild motion capture of any subject.

Chapter 3

AirPose

In the previous chapter, we introduced AirCap-Pose-Estimator, which uses multiple UAVs to track and follow a person, record RGB images, and then post-process them to obtain human 3D pose and shape. Even though the subject detection and tracking happens onboard using a neural network, the subject's 3D pose is estimated using an optimization-based method which is very slow and requires heavy computational resources. Another major limitation of the Aircap-Pose-Estimator is the significant dependency on the data acquisition system which has the following disadvantages. **(1)** Cameras are rigidly attached to the UAVs which results in the subject frequently getting out of the frame, **(2)** the cameras produce very large size images with unnatural colors and inconsistent frame rate, **(3)** the onboard sensors and computation are needed to get the initial poses of the UAVs, and **(4)** the UAVs are heavy and require a trained pilot to fly.

In this chapter, we propose a new method AirPose focusing on the real-time 3D pose and shape estimation of the subject and the poses of the UAVs. To achieve this, we move from an optimization-based method to a neural network based regression method for the human 3D pose estimation. Neural network based methods are generally faster because they do not need multiple iterations of gradient calculations required by optimization methods. We also move away from custom-made UAVs to the commercially available UAVs. These UAVs have cameras attached to a gimbal and are easy to fly even for a beginner. Even though the inertial and GPS sensors are available on these UAVs their measurements are not easily accessible. Therefore, we do not use them and assume only the UAV cameras are available to us.

AirPose is a distributed, multi-view fusion method, designed for the on-board estimation of the 3D body pose and shape of a single person from **multiple, moving, and extrinsically-uncalibrated** cameras. It achieves this by running identical lightweight networks (AirPoseNet) on each UAV, which takes in an image from the camera, combines information from other views and regresses the 3D shape and pose of the person relative to the camera. We build upon an existing monocular method HMR (4) (discussed in Sec. 1.4.1) which takes in a cropped image around the person and scaled to a square size of 224 pixels. This cropping and scaling operation is needed to keep the input data small for faster execution. However, it also results in the loss of information regarding the global position of the person. We propose a novel input representation for cropped camera images which retain this information while maintaining the fast execution speed. The network architecture of AirPoseNet enables distributed processing of different camera views on each UAV and efficient sharing and fusion of relevant information between the UAVs. Training such a network is a significant challenge because of the unavailability of the training data. A multi-view aerial data of a variety of moving subjects with ground-truth poses is extremely difficult to collect. That is why no such data exists. We develop a synthetic data generation pipeline to solve the problem of training AirPoseNet. Since AirPose is a lightweight real-time method which takes in a single image at a time, its mocap quality is not very high. To address this, we propose an off-board optimization-based method AirPose$^+$, which improves the AirPose results by fitting them to the 2D keypoints on the image.

We quantitatively evaluate AirPose on synthetic data and show that it outperforms a baseline model which uses neither the proposed compact image representation nor the information sharing across the views. We also show the ablation study comparing the effects of these two components on different metrics. We further demonstrate and evaluate our approach through hardware-in-the-loop experiments using a real-world dataset collected from 2 commercial UAVs. We show that AirPose runs in real-time on an embedded GPU.

In summary, our novel contributions are: **(1)** A distributed and decentralized system of neural networks (AirPoseNet) for uncalibrated moving cameras that estimates human 3D pose and shape, while simultaneously calibrating the cameras with respect to the human. **(2)** A compact input image representation that significantly improves the human position estimate, even for the monocular case. **(3)** A realistic-looking synthetic training data generation pipeline for overhead, multi-view images of humans with ground-truth pose and shape. **(4)** An off-board optimization-based method, AirPose$^+$, which further refines the mocap quality and the camera calibration (see Fig. 3.1). **(5)** Code and data of our method are available for research purposes at `https://github.com/robot-perception-group/AirPose`.

3.1 Method

Problem Statement. Our goal is to develop a method that accurately estimates the 3D pose and shape of a person from multiple uncalibrated cameras with the following constraints. It should be able to run onboard UAVs with small computation capabilities and limited wireless communication. A naive approach to this problem is to use a state-of-the-art monocular human pose estimator on each UAV. This facilitates distributed and decentralized computation. However, the estimate from one UAV would not benefit from the other UAV's viewpoint. We call this the baseline method. We introduce AirPose, where information from other viewpoints is incorporated into a UAV's estimate and the computation remains distributed and decentralized.

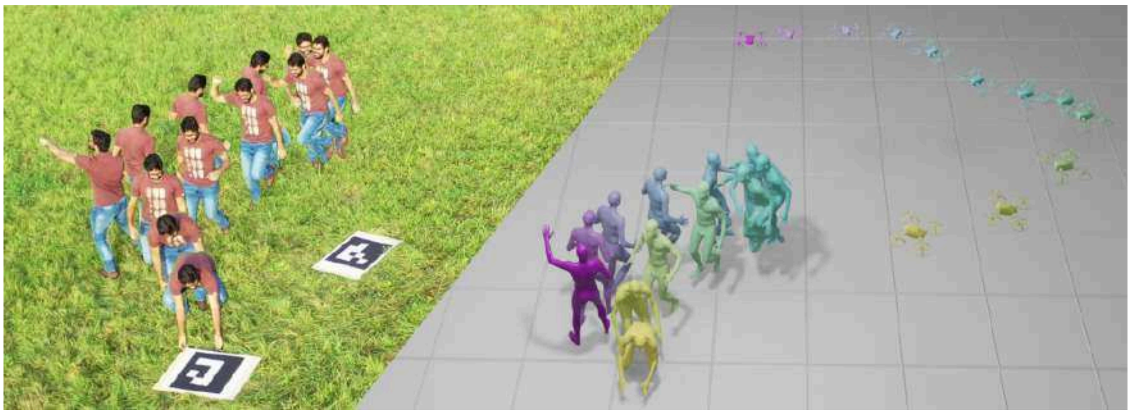


Figure 3.1: AirPose: A novel, distributed, multiview fusion method for 3D pose and shape estimation of humans using uncalibrated moving cameras. Multi-exposure image of a sequence from our real-world data (left). AirPose^{+} estimates of the person (right). UAV poses are also estimated but here we manually place them for clearer illustration. ArUco (6) markers were in the scene but never used for any of our methods.

3.1.1 Baseline Method

We adapt HMR (4) to develop a baseline method for our problem. HMR network consists of the ResNet50 backbone followed by an autoregressor network (see Sec. 1.4.1). We deploy one HMR instance on each UAV outputs SMPL-X (see Sec. 1.2.3) pose and shape estimates of the person in the UAV camera's reference frame. The setup is shown in Fig. 3.2 (left). The input consists of only a cropped and scaled region (where the person is present) of the full-size image. Thus, the output needs to be transformed to the original camera reference frame. The root translation (root refers to the root joint in the person's pose) in the original image frame is given as $\tilde{\tau} = [\tilde{x}, \tilde{y}, \tilde{z}]$, and in the cropped and scaled camera frame is given as $\tilde{\tau}_c = [\tilde{x}_c, \tilde{y}_c, \tilde{z}_c]$, where the vector components correspond to the 3D Euclidean coordinates. The output of the neural network is divided into four components: i) $\tilde{\tau}_c \in \mathbb{R}^3$, ii) $\tilde{\phi} \in \mathbb{R}^6$, the root rotation, iii) $\tilde{\theta} \in \mathbb{R}^{126}$, the articulated pose, and iv) $\tilde{\beta} \in \mathbb{R}^{10}$, the body shape parameters. The dimensions of these parameters are slightly different from the original SMPL-X model because we use the 6D representation (94) for rotations instead of the axis angle representation. SMPL-X pose parameters include hands and face parameters along with the body parameters, but, we use only the body parameters and fill the rest with zero values. The relationship between $\tilde{\tau}$ and $\tilde{\tau}_c$ is given as

$$\tilde{z} = \tilde{z}_c s, \tag{3.1}$$

$$\begin{bmatrix} f_x & 0 & 0 \\ 0 & f_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \tilde{x}/\tilde{z} \\ \tilde{y}/\tilde{z} \\ 1 \end{bmatrix} = \begin{bmatrix} f_x/s & 0 & b_x\, c_x \\ 0 & f_y/s & b_y\, c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \tilde{x}_c/\tilde{z}_c \\ \tilde{y}_c/\tilde{z}_c \\ 1 \end{bmatrix}, \tag{3.2}$$

where f_x and f_y are focal length parameters of the camera and (c_x, c_y) its principal point. b_x and b_y are the normalized coordinates of the cropped region in the original image (see Fig. 3.3). s is the scale applied to resize the cropped region to the size 224 by 224, the input size of the feature extractor. As the baseline is a monocular method, we train it for one camera and run its instance independently for each UAV. The loss for training our baseline method is

$$L_{\text{baseline}} = w_{j2d}L_{j2d} + w_{j3d}L_{j3d} + w_{\phi}L_{\phi} + w_{\theta}L_{\theta} + w_{\beta}L_{\beta} + w_{V}L_{V}, \tag{3.3}$$

$$\begin{gathered} \text{where } L_{j2d} = ||\Pi(\mathcal{J}(\tilde{\tau},\tilde{\phi},\tilde{\theta},\tilde{\beta})) - \Pi(\mathcal{J}(\tau_{gt},\phi_{gt},\theta_{gt},\beta_{gt}))||^2, \\ L_{j3d} = ||\mathcal{J}(\tilde{\theta},\tilde{\beta}) - \mathcal{J}(\theta_{gt},\beta_{gt})||^2, \quad L_{\phi} = ||\tilde{\phi} - \phi_{gt}||^2, \\ L_V = ||\mathcal{S}(\tilde{\theta},\tilde{\beta}) - \mathcal{S}(\theta_{gt},\beta_{gt})||^2, \quad L_{\theta} = ||\tilde{\theta} - \theta_{gt}||^2,\ L_{\beta} = ||\tilde{\beta}||^2, \\ w_{j2d} = 0.01,\ w_{j3d} = 1,\ w_{\phi} = 1,\ w_{\theta} = 100, w_{\beta} = 1 \text{ and } w_V = 100. \end{gathered}$$

$\mathcal{J}$ and $\mathcal{S}$ are the SMPL-X 3D joint and mesh vertex regression functions. For L_{j3d} and L_V, the default values (zero-filled vector) are provided for $\tilde{\tau}$ and $\tilde{\phi}$. Π is the camera projection function. τ_{gt}, ϕ_{gt}, θ_{gt}, β_{gt} are ground truth values of the corresponding SMPL-X parameters w.r.t. the camera. We use an L2 loss function following the previous works (20; 80). The loss weights are chosen as follows. Since there are many loss components, the network training is highly unstable. We first stabilize the training by selecting hyperparameters from a sparse hyperparameter space. Thereafter, we narrow down the search space. We observe that even though the overall training and validation loss keeps going down, the model can overfit to some loss components. In such cases, we stop the training once the model starts overfitting on any loss component.

3.1.2 Proposed Method – AirPose

We first highlight the shortcomings of the baseline method and then present our insights to solve them.

Insight 1. Some parts of the person's body could be occluded in one camera view, but available in other cameras. Thus, using the output of each individual network, or even simply averaging the outputs from multiple networks, would not result in an accurate estimate. A systematic approach to information fusion is required to improve the estimate of the person's pose by leveraging complementary information from different views. The fusion problem is exacerbated by limited wireless communication bandwidth between the UAVs, prohibiting the real-time exchange of images among them. To perform fusion of information, while remaining within the communication constraints, we propose a

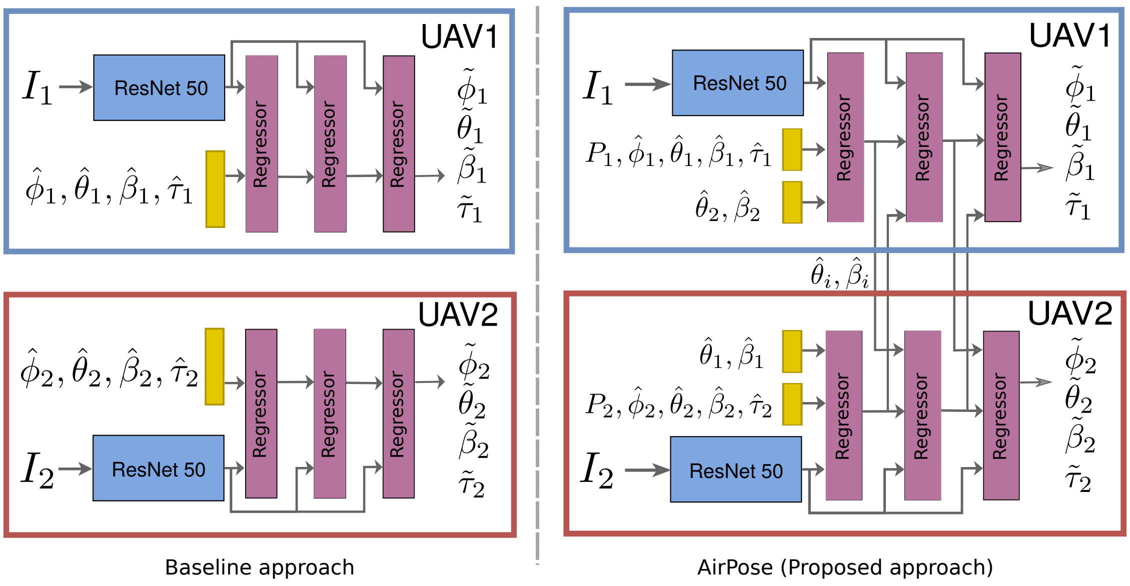

UAV1
I_1
ResNet 50
Regressor
$\hat{\phi}_1, \hat{\theta}_1, \hat{\beta}_1, \hat{\tau}_1$
$\tilde{\phi}_1$ $\tilde{\theta}_1$ $\tilde{\beta}_1$ $\tilde{\tau}_1$
UAV2
$\hat{\phi}_2, \hat{\theta}_2, \hat{\beta}_2, \hat{\tau}_2$
I_2
$\tilde{\phi}_2$ $\tilde{\theta}_2$ $\tilde{\beta}_2$ $\tilde{\tau}_2$
$P_1, \hat{\phi}_1, \hat{\theta}_1, \hat{\beta}_1, \hat{\tau}_1$
$\hat{\theta}_2, \hat{\beta}_2$
$\hat{\theta}_i, \hat{\beta}_i$
$\hat{\theta}_1, \hat{\beta}_1$
$P_2, \hat{\phi}_2, \hat{\theta}_2, \hat{\beta}_2, \hat{\tau}_2$
Baseline approach
AirPose (Proposed approach)

Figure 3.2: The network architecture of the baseline and proposed approach (AirPose). The neural network on each UAV takes in a cropped and scaled image *I* to give the body parameters relative to the camera. Please refer Sec. 4.1 for more information about the symbols.

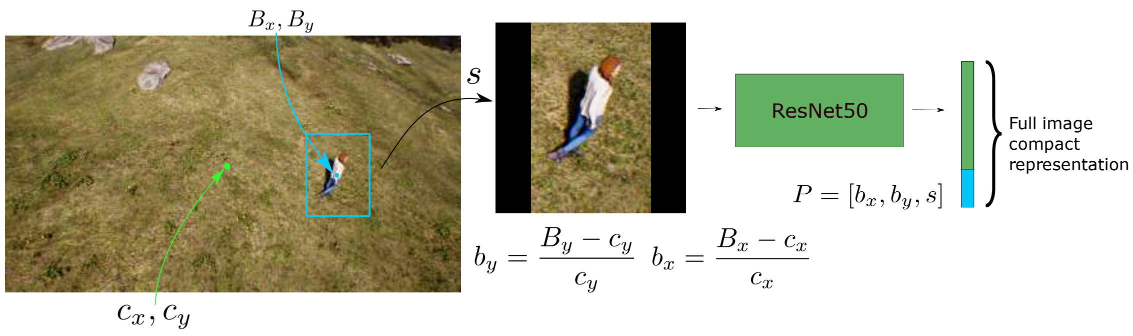

B_x, B_y
s
c_x, c_y
ResNet50
$P = [b_x, b_y, s]$
Full image compact representation
$b_y = \frac{B_y - c_y}{c_y}$
$b_x = \frac{B_x - c_x}{c_x}$

Figure 3.3: The bounding-box region is cropped & scaled to the fixed size image for ResNet50 input. The full-size image is represented by concatenating the ResNet50 features and the cropping & scaling parameter *P*.

novel decentralized and distributed neural network (see Fig. 3.2 (right)). In our proposed network, the estimated articulated pose (θ) and body shape (β) from any autoregression stage of one network, running on one UAV, is fed to the next autoregression stage of another network, running on another UAV. These body parameters are independent of the viewpoints from which the person is being seen. If any body part is occluded in one view, its estimate is improved by using the information shared by the other view. There are three autoregression stages in each instance of the network, hence the total information shared per UAV per image frame is only $2 \cdot (126 + 10) = 272$ float32.

Insight 2. In the baseline method, cropping and scaling of the image results in loss of information, which is crucial for the correct estimation of the root translation. Multiple full-size images where the person is in the same pose but in different locations can result in the same cropped and scaled image. That is why the neural network cannot uniquely determine the 3D location of the person just based on the cropped and scaled image. It needs the original image to resolve the 3D location of the person. However, passing the full image will result in a significant computation overhead and thus a high execution time. Our solution is to provide the network with a compact representation of the full image. This representation is the concatenation of P with the extracted feature vector of the cropped and scaled image, where $P = [b_x, b_y, s]$. Since these parameters are used in the first place to crop and scale the original image, they can also be used to recreate the full-size image with some data loss (such as blurry pixels in the bounding box and blank pixels around it). The network can figure out how to do this inverse operation to estimate the 3D location of the person. This idea is also used by a recent method CLIFF (15). However, as opposed to our method which feeds the cropping and scaling information to the network and lets the network figure out 3D location relative to the original image (camera), they estimate the 3D location of the person relative to the cropped and scaled image (camera) and then analytically compute the 3D location relative to the original image (camera) based on the cropping and scaling parameters.

Based on the above two insights, our new network architecture is conceived as follows. It has a ResNet50 feature extractor followed by an autoregressor stage. The feature extractor extracts latent features from the cropped and scaled image similar to the baseline. These features, concatenated with P, contain the articulated and global pose information of the person. This compact input is used by the autoregressor, along with the SMPL-X parameters, initialized as $\hat{\phi}, \hat{\theta}, \hat{\beta}, \hat{\tau}$ (fixed values). $\hat{\beta}$ is a vector of zeros, and $\hat{\phi}, \hat{\theta}$ are initialized from the same values as in (4). We assume that the subject's position can vary between 0 and 20 meters relative to the camera. That is why, we chose the mean position of the subject to be at $[0, 0, 10]$. We normalize it by dividing with value 20 to adjust the range between 0 and 1, and initialize $\hat{\tau} = [0, 0, 0.5]$. The estimated person position parameter $\tilde{\tau}$ is multiplied by this normalizing factor to get the actual position. The autoregressor architecture is the same as in (4). It consists of three fully connected (FC) layers, with a dropout layer after the first and the second FC layer. The autoregressor eventually outputs the refined SMPL-X parameters w.r.t. the camera. The training loss function of AirPose is

$$L_{\text{AirPose}} = w_{j2d} L_{j2d} + w_{j3d} L_{j3d} + w_{\phi} L_{\phi} + w_{\tau} L_{\tau} + w_{\theta} L_{\theta} + w_{\beta} L_{\beta} + w_V L_V, \tag{3.4}$$

$$\text{where } L_{j2d} = \sum_c ||\Pi(\mathcal{J}(\tilde{\tau}_c, \tilde{\phi}_c, \tilde{\theta}_c, \tilde{\beta}_c)) - \Pi(\mathcal{J}(\tau_{gt,c}, \phi_{gt,c}, \theta_{gt}, \beta_{gt}))||^2$$

$$L_{j3d} = \sum_c ||\mathcal{J}(\tilde{\theta}_c, \tilde{\beta}_c) - \mathcal{J}(\theta_{gt}, \beta_{gt})||^2, \quad L_{\tau} = \sum_c ||\tilde{\tau}_c - \tau_{gt,c}||^2,$$

$$L_\phi = \sum_c ||\tilde{\phi}_c - \phi_{gt,c}||^2, \quad L_\theta = \sum_c ||\tilde{\theta}_c - \theta_{gt}||^2 + ||\tilde{\theta}_1 - \tilde{\theta}_2||^2,$$
$$L_V = \sum_c ||\mathcal{S}(\tilde{\phi}_c, \tilde{\theta}_c, \tilde{\beta}_c) - \mathcal{S}(\phi_{gt,c}, \theta_{gt}, \beta_{gt})||^2$$
$$+||\mathcal{S}(\tilde{\phi}_1, \tilde{\theta}_1, \tilde{\beta}_1) - \mathcal{S}(\tilde{\phi}_2, \tilde{\theta}_2, \tilde{\beta}_2)||^2, \ L_\beta = \sum_c ||\tilde{\beta}_c||^2 + ||\tilde{\beta}_1 - \tilde{\beta}_2||^2,$$
$$w_{j2d} = 0.002, \ w_{j3d} = 1, \ w_\phi = 1, w_\tau = 10, \ w_\theta = 50, \ w_\beta = 1 \text{ and } w_V = 50.$$

Each camera has its own SMPL-X parameter estimates, e.g., θ_c is the articulated pose parameter for camera c, $c \in \{1,2\}$. $\mathcal{S}$ is the SMPL-X vertices regressor function (see Sec. 1.2.3).

Insight 3. Most monocular methods are trained on data that does not contain overhead and oblique views of persons. In aerial mocap, such viewpoints are predominant. On the other hand, there also exist few multiview image datasets on which our baseline method can be trained, let alone those with overhead viewpoints. To address both these dataset-related challenges, we train our networks (both baseline and AirPose) using large sets of synthetic images in realistic virtual environments and fine-tune using a small set of real images from UAVs. During the fine-tuning, the weights of ResNet50 feature extractors are frozen, and only the regressor is trained. We use OpenPose (16; 87) to get the 2D key-points on the images and use them for supervision during the fine-tuning. To make the 2D keypoints more reliable, we also get them from another detector, Alphapose (86). If the OpenPose estimates deviate from the Alphapose estimates by more than a threshold value (of 100 pixels), they are discarded. Using only the 2D keypoints for supervision might result in unnatural body poses. Therefore, we use VPoser to avoid unnatural human poses. We use its encoder network ($\mathcal{V}_E$) instead of the decoder network ($\mathcal{V}_D$), which allows us to directly predict pose parameters θ instead of latent pose parameters v during the evaluation time. In the latter case, an additional computation $\mathcal{V}_D(v)$ is required during test time resulting in a larger overall processing time. We project the estimated poses from our network into the latent space and restrict it to be close to the mean of the VPoser's distribution. VPoser is trained for SMPL (19) model, therefore, we use only the body pose parameters of SMPL-X which also corresponds to the same representation as SMPL, i.e. body joint angles. Note that SMPL and SMPL-X parameters are not interchangeable and they differ a little for the same joint angles. However, such a difference is fine for our case as we are using SMPL-trained VPoser just to avoid unnatural poses and the 2D keypoint is the main supervision to guide the pose estimation. The fine-tuning loss for the baseline approach is given as

$$L_{f_{\text{Baseline}}} = w_{j2d} L_{j2d} + w_\beta L_\beta + w_{vposer} L_{vposer}, \tag{3.5}$$
$$\text{where } L_{j2d} = \sum_n w_n((\Pi(\mathcal{J}_n(\tilde{\tau}, \tilde{\phi}, \tilde{\theta}, \tilde{\beta})) - j_n)^2), \quad L_\beta = ||\tilde{\beta}||^2,$$
$$L_{vposer} = ||\mathcal{V}_E(\tilde{\theta})||^2, \quad w_{j2d} = 0.01, \quad w_\beta = 5 \quad \text{and} \quad w_{vposer} = 1.$$

j_ns are the 2D coordinates estimated by OpenPose for keypoint n. w_ns is the confidence score of the corresponding keypoint estimates from OpenPose. $\mathcal{J}_n$ is the joint regressor function that gives the n^{th} joint position given the SMPL-X pose and shape parameters. $\mathcal{V}_E(\theta)$ is the sample from the latent space distribution obtained after passing θ to the Vposer encoder. The fine-tuning loss for AirPose is

$$L = w_{j2d}L_{j2d} + w_\beta L_\beta + w_{vposer}L_{vposer} + w_\theta L_\theta, \tag{3.6}$$

$$\text{where } L_{j2d} = \sum_{n,c} w_{n,c}((\Pi(\mathcal{J}_n(\tilde{\tau}_c, \tilde{\phi}_c, \tilde{\theta}_c, \tilde{\beta}_c)) - j_n)^2), \quad L_\theta = ||\tilde{\theta}_1 - \tilde{\theta}_2||^2$$

$$L_\beta = \sum_c (||\tilde{\beta}_c||^2) + ||\tilde{\beta}_1 - \tilde{\beta}_2||^2, \quad L_{vposer} = ||\mathcal{V}_E(\tilde{\theta})||^2$$

$$w_{j2d} = 0.01, \quad w_\beta = 5, \quad w_{vposer} = 0.1 \quad \text{and} \quad w_\theta = 100, \tag{3.7}$$

where $w_{n,c}$ is the OpenPose confidence score for joint n and camera view c. Similarly, $\tilde{\tau}_c, \tilde{\phi}_c, \tilde{\theta}_c, \tilde{\beta}_c$ are the SMPL-X parameters for camera view c.

3.1.3 Proposed Approach – AirPose$^+$

Finally, we propose a post-processing optimization method, AirPose$^+$, where we utilize the temporal information to further refine the human pose and shape and camera pose estimates given by AirPose. This is done by minimizing the loss function given in Eq. (3.8). We optimize the parameters θ, β, ϕ_c, τ_c for the whole capture sequence. We also put a constraint over these parameters to be close in adjacent frames. We use the latent representation of VPoser, v, to represent the articulated pose. The SMPL-X pose parameter θ is obtained by passing v through the Vposer decoder $\mathcal{V}_D$ i.e. $\theta = \mathcal{V}_D(v)$.

$$L = \sum_t (w_{j2d}L_{j2d} + w_{vposer}L_{vposer} + w_{temp}L_{temp}) + w_\beta L_\beta, \tag{3.8}$$

$$\text{where } L_{j2d} = \sum_{n,c} w_{c,n}(\rho_\sigma(\Pi(\mathcal{J}_n(\tilde{\tau}_c, \tilde{\phi}_c, \mathcal{V}_D(\tilde{v}), \tilde{\beta})) - j_n)), \quad L_{vposer} = ||\tilde{v}||^2,$$

$$L_{temp} = ||\tilde{\theta}_t - \tilde{\theta}_{t-1}||^2 + \sum_c ||\tilde{\phi}_t - \tilde{\phi}_{t-1}||^2 + ||\tilde{\tau}_t - \tilde{\tau}_{t-1}||^2,$$

$$L_\beta = ||\tilde{\beta}||^2, \quad w_{j2d} = 1, \; w_{vposer} = 0.05, \; w_{temp} = 1 \text{ and } w_\beta = 2000.$$

ρ_σ is the Geman-Mcclure robust penalty function (see Eq. (2.4)). All the individual loss terms except L_β in the Eq. (3.8) are a function of t. However, t is not used in their notation to improve the readability. Unlike AirPose, in AirPose$^+$, articulated pose v is not different for each camera and β is constant for each camera throughout the sequence. We compute $\sum_{n,c} w_{c,n}$ for each frame t and ignore the frame in optimization if this value is below a threshold.

3.2 Training And Evaluation

An ideal dataset for training our network requires synchronized video sequences of many persons with a variety of poses and with ground truth SMPL-X parameters. It is difficult to collect such data because of reasons like the limited battery life of UAVs, weather-related uncertainties, flying permission from authorities, availability of licensed pilots, etc. Thus, we generate realistic-looking synthetic image data in realistic virtual environments. Even though the generated images look realistic, the network trained with these images does not generalize well to real-world images. That is why we fine-tune the network on a small amount of real-world images. The Aircap-Pose-Estimator data we presented in the last chapter is the only existing real-world image data collected using UAVs. However, it is not suitable for fine-tuning our network because of the following reasons. 1) The subject is wearing a mocap suit and a firefighter helmet which are not normal clothing, also, the image colors are not natural looking (see Fig. 2.2). Due to these reasons, the image is very different from the normal images and out of the distribution of the training data. 2) Since the person frequently gets out of the image frame (see Sec. 2.3), we cannot run our bundle adjustment method AirPose$^+$ on it because the method relies on the person being visible in the frame and smooth camera motions. That is why, We collect new real data with two persons and two DJI UAVs (7000 frames per person per UAV). We fine-tune our network using the real data with one person and evaluate it on the other person.

Synthetic Data for Training: We generated the synthetic data ($\sim$ 30000 frames per UAV) by putting realistic human scans (7) in Unreal Engine (UE) and rendering them from multiple viewpoints of 2 UAVs, as shown in fig. 3.4. We use the AirSim plugin (95) for UE to move the scans and the cameras around such that the data generation process is automated. The scans are put in an outdoor UE environment (purchased from UE Marketplace) and moved between -2.75m to 2.75m along the X and the Z-axis (the Y-axis points outward from the ground plane in UE). The cameras are moved independently and randomly around the origin such that they are facing the origin and the distance from the origin is $\sim$10m. The pitch of the cameras varies from $0°$ to $45°$. The SMPL-X fittings to scans are provided by (7). The fittings are done using the gender-specific SMPL-X model.

Real Data for Fine-tuning: We collected 2 real data sequences. Each sequence uses two DJI Mavic UAVs, flying manually around a different person. The raw data are sets of video sequences from the UAVs. During the acquisition, we keep one of the UAVs hovering in place and the other one is manually flown around the person, who performs various motions, covering a wide range of poses at a safe distance from the UAVs. The intrinsic calibration of the UAV cameras was done immediately before the take-off, using the chessboard calibration method. The frames were extracted and manually time-synchronized. We sampled the corresponding frames from the two UAVs and found that they had a constant time difference, which implies that the two devices had the same

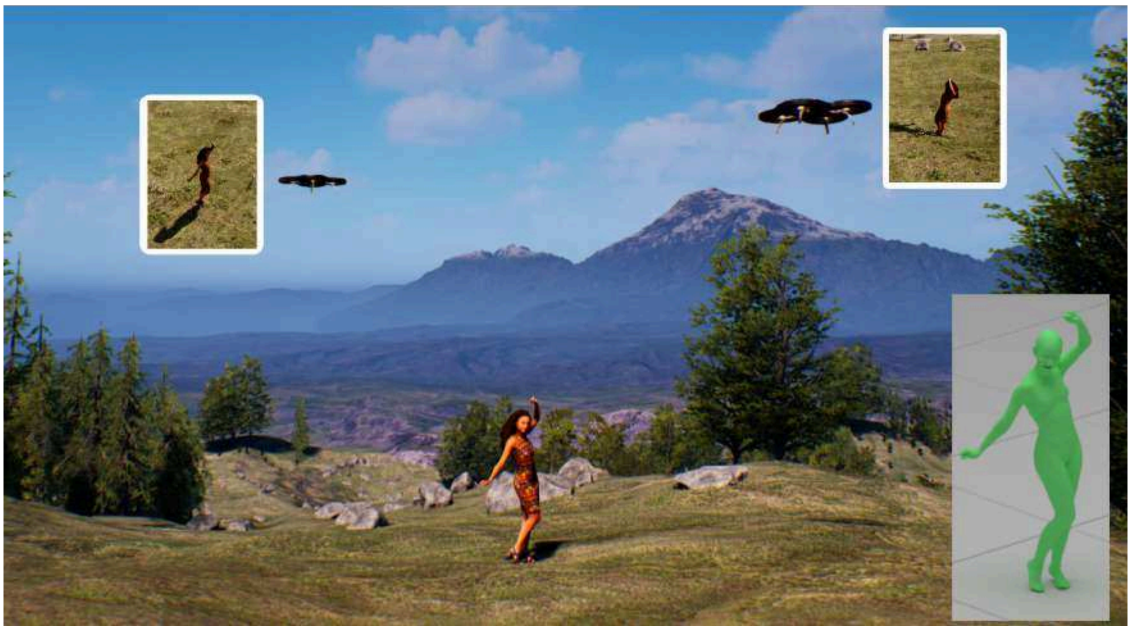


Figure 3.4: A realistic human scan in an Unreal Engine environment with AirSim UAVs (cameras). The corresponding images are shown beside each UAV. The SMPL-X fitting from AGORA dataset (7) is shown in the right bottom corner.

frame rate, and it remained constant throughout the acquisition time. ArUco markers (6) were in the scene, but never used for any of our methods.

Hardware-in-the-loop evaluation and synchronization strategy for online execution: To run our approach online we took a hardware-in-the-loop approach, where communication and synchronisation were performed in real-time on our actual UAV hardware (which has an Intel i7 CPU and Nvidia Jetson TX2 on-board) (28). We implemented a ROS (96) based synchronization framework that selects matching camera frames based on their shutter timestamp. Clocks were synchronized using NTP (97). Communication via 5 GHz wifi with 50 Mbps was handled by ROS. On the Jetson TX2 modules, the AirPoseNet achieved inference speeds of 50 ms per frame (224x224x3 bytes 20 FPS). This was dominated by the first step of the network (i.e. ResNet50 and the first regressor stage, $\sim$ 43 ms) while the other two steps are almost immediate ($\sim$ 2.5 ms). However, in a real-world setting, high-resolution images and communication must be accounted for. We allocated 2 * 25 ms for the communication of the 584-byte encoding, as well as 140 ms for the acquisition and downsampling of the 4K camera images after benchmarking these operations individually on our UAVs. To accommodate sufficient time for image acquisition, inference, and multiple iterations of data communication, we allocated a fixed length time window of 240 ms to process one frame, which results in a fixed overall framerate of 4.17 FPS. If processing could not be completed by the end of the window, the frame was discarded. Taking dropped frames into account, AirPose achieved an average framerate of around 3 FPS. The camera itself was capturing 4K frames at 40 FPS. It must be noted that the bottleneck in this ad-hoc approach is not the AirPoseNet, but acquisition and preprocessing as well as communication overhead. If lower resolution camera images were used in conjunction with optimized buffers to acquire, process and communicate data in parallel, it is easily conceivable that AirPose could reach 20 FPS realtime-throughput at a latency of less than 150 ms. Real image datasets from (27; 28) are not directly usable in our approach due to significant differences with respect to the appearance of the images on which the network is originally trained (i.e. synthetic data). To overcome this problem, we collected a custom video dataset with two DJI drones for evaluation (similar to the data for fine-tuning) and converted them into Rosbags. We manually synchronized the first common frame between the two video sequences. Subsequently, we replayed the Rosbags as if they were taken by the UAV cameras and used the hardware-in-the-loop setting and synchronization architecture as explained above.

3.3 Experiments And Results

3.3.1 Results on the Synthetic Data

We perform an ablation study and compare the following 4 methods using the synthetic data.

1. Baseline: The baseline method is described in Sec. 3.1.1.

2. Baseline + Multi-view: Here, we expand the input size of the baseline regressor to take the θ and β values from another view. However, the regressor does not take the image cropping and scaling information P and estimates the SMPL-X parameters w.r.t. the cropped image.

3. Baseline + Fullcam: Here, in addition to the baseline inputs, we take the cropping and scaling information, P, and estimate the pose of the person w.r.t. the original camera. It is equivalent to AirPose without communication between the two UAVs.

4. AirPose (Baseline + Multi-view + Fullcam): The proposed method, as described in Sec. 3.1.2.

Error Comparison Metrics: We compare the global position and articulated pose estimates of the person from the four methods. Since, the person's pose is estimated by each UAV relative to itself, for computing error we convert the estimates into the global frame

	Baseline	Baseline + Multi-view	Baseline + Fullcam	AirPose
MPE (m)	0.50	0.46	0.22	**0.15**
MPJPE (m)	0.091	0.084	0.077	**0.072**

Table 3.1: **Ablation study** of AirPose on the synthetic dataset.

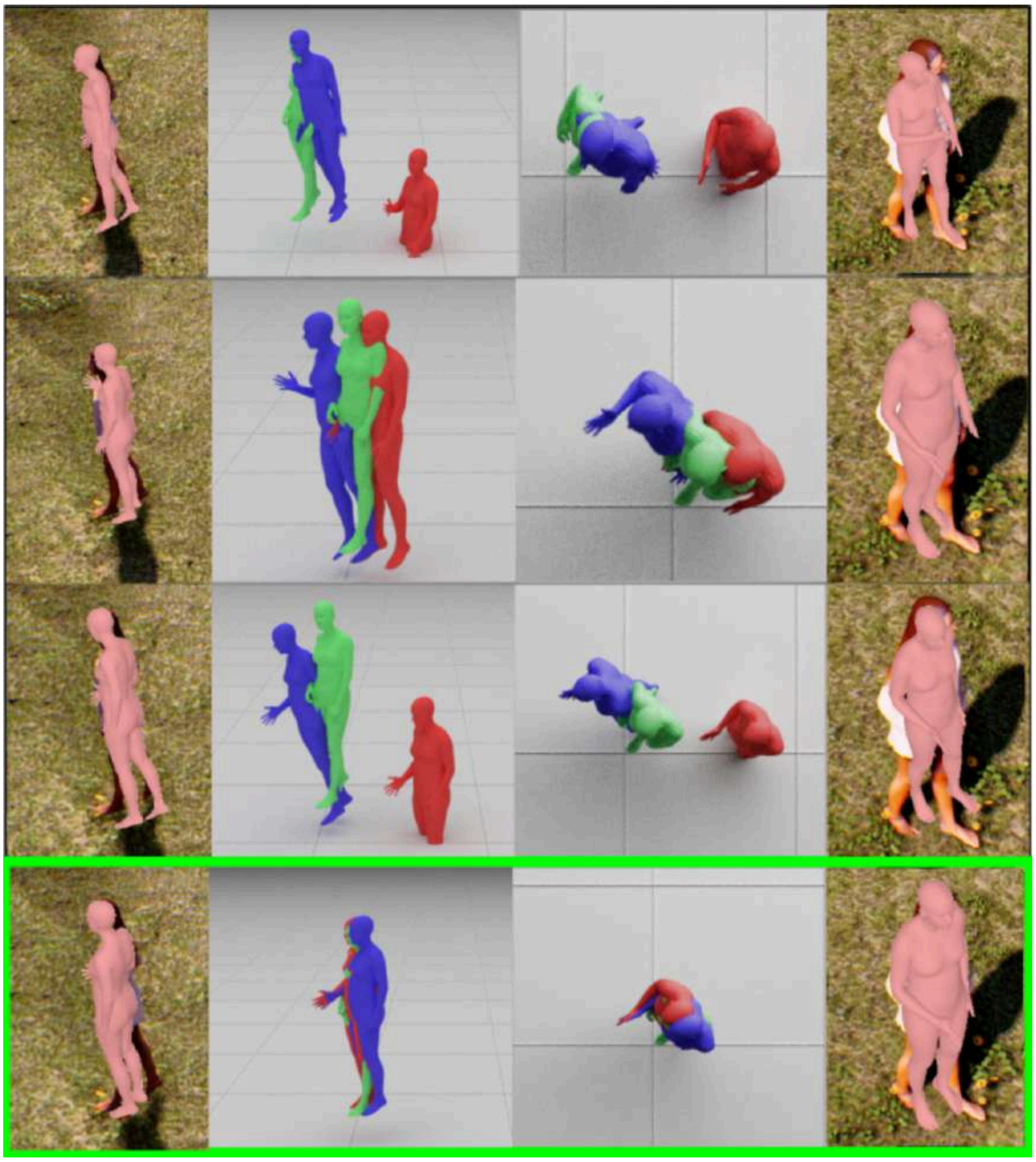


Figure 3.5: Ablation study (Sec. 3.3.1) qualitative results. Row 1: 'Baseline'. Row 2: 'Baseline+Multi-view'. Row 3: 'Baseline+Fullcam'. Row 4: 'AirPose'. 1st and 4th columns are cropped images showing the overlaid estimated mesh w.r.t. the camera. The 2nd and 3rd columns show the front and the top views of the 3D scene, where the two estimates (one from each view) are transformed into the global coordinate frame. Red mesh: estimate from the first camera. Blue mesh: estimate from the second camera. Green mesh: ground truth SMPL-X mesh.

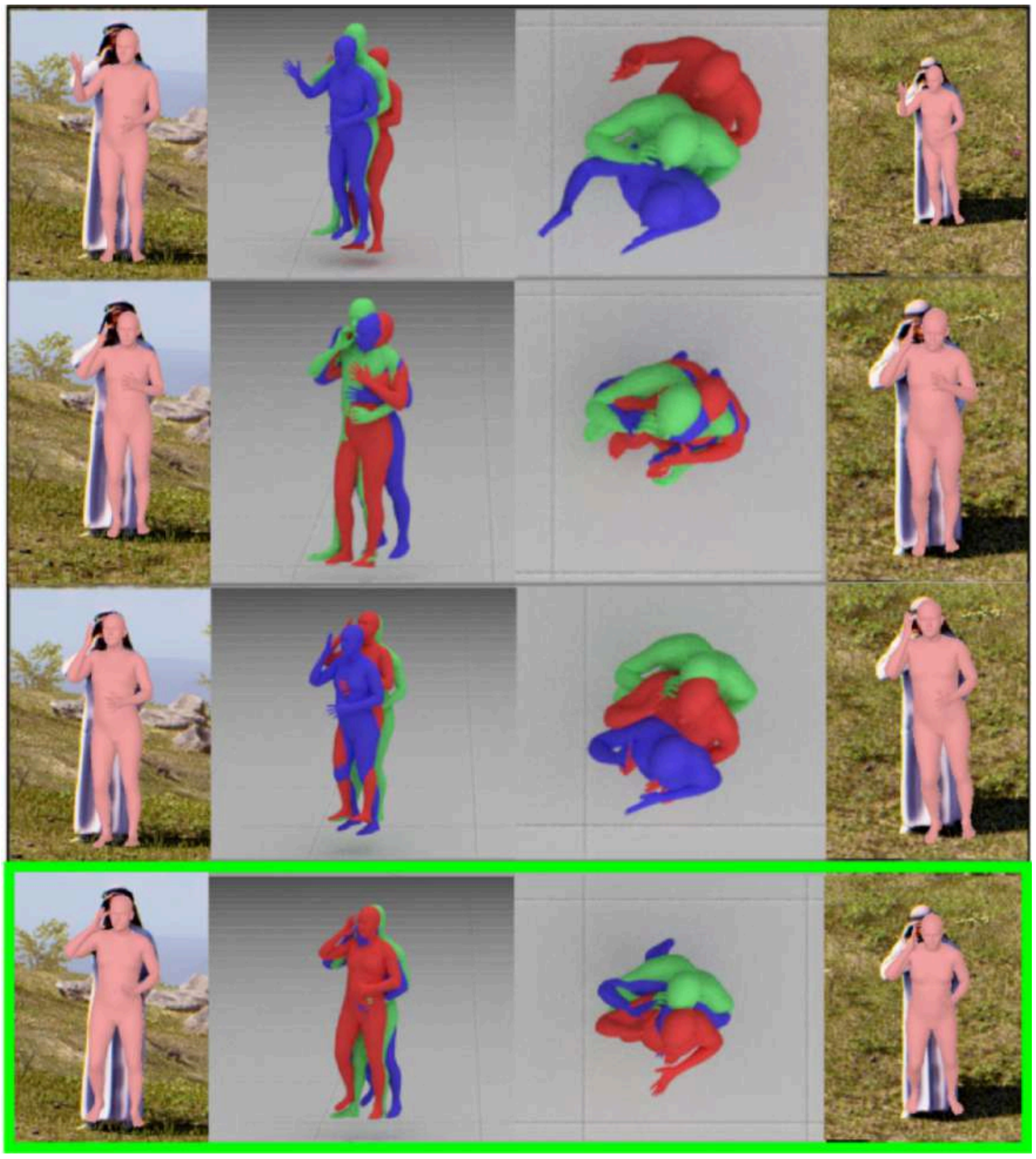


Figure 3.6: Another example showing qualitative results of Baseline, Baseline+Mutli-view, Baseline+Fullcam and AirPose. See the caption of Fig. 3.5 for details.

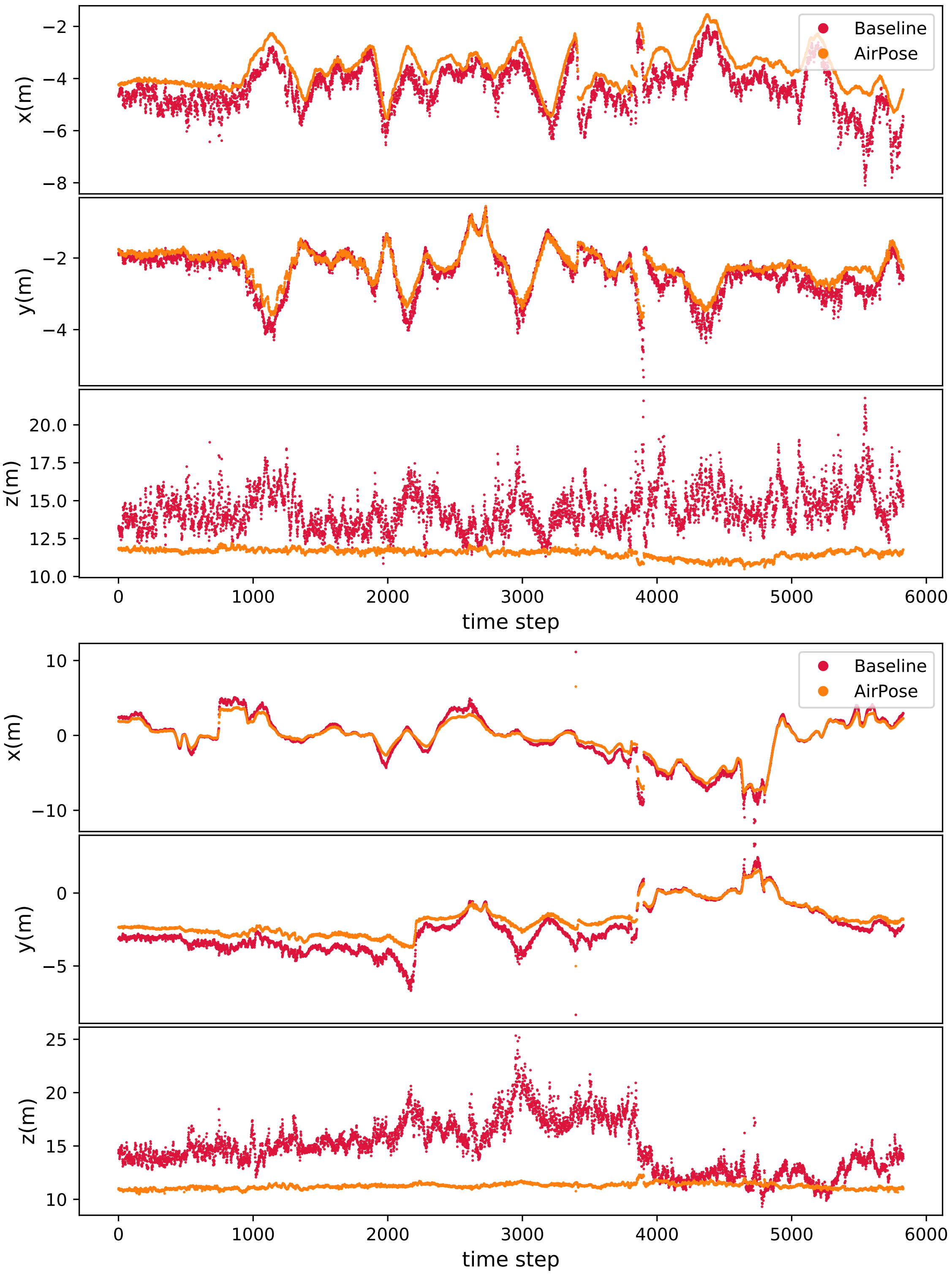


Figure 3.7: Comparison of the person 2 3D position trajectory, estimated by baseline and AirPose methods on UAV 1 (hovering, top) and 2 (circling, bottom).

using the ground truth extrinsics of the UAVs. We calculate the error of each estimate w.r.t. the person's ground truth (GT) in the global frame. To evaluate the global position estimate, we calculate the mean position error (MPE) as MPE $= \frac{1}{2T}\sum_{t=1}^{T}\sum_{c}||\tilde{\tau}_c^o - \tau_{gt}^o||$. This is the mean (over all the images from both UAVs) of the Euclidean distances between the GT and the estimated person's positions in the global frame (denoted by the superscript ^{o}). The total number of image frames are T.

For the evaluation of the articulated pose estimate of the person, we calculate the mean of the joint position errors (MPJPE) as MPJPE $= \frac{1}{22(2T)}\sum_{t=1}^{T}\sum_{n}\sum_{c}||\tilde{\tau}_c^o - \tau_{gt}^o||$, over all the images from both UAV cameras. τ is a function of n but it is not denoted here to improve the readability. The joint error is the Euclidean distance between the estimated joint position and its corresponding GT when the root translation for both the estimate and GT are aligned. Since we are using only the body parameters of SMPL-X, the number of joints in our case is 22.

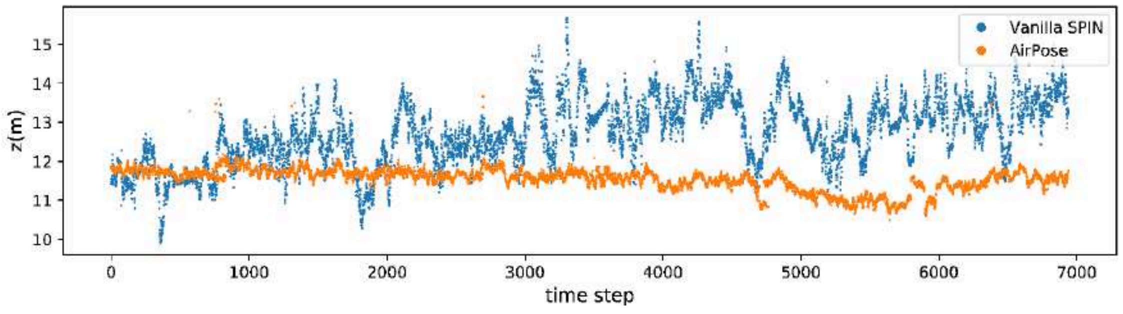


Figure 3.8: Comparison of the estimated Z coordinate of the tracked person's trajectory by AirPose and the vanilla SPIN.

Results and Discussion: Table 3.1 shows the MPE and MPJPE values for the four different methods. AirPose, outperforms all ablated methods on both metrics. The MPE of 'Baseline' and 'Baseline + Multi-view' is similar and much higher than the other two methods. This highlights the problem of using cropped images without the full image information. The input image does not contain information about the position of the person in the full image, and the camera center is incorrectly assumed to be the center of the bounding box. Since the person is viewed from above, there are few self-occlusions present in the dataset. Due to this, all the methods have similar MPJPE. Nevertheless, AirPose, combining information from both views, has the lowest MPJPE. In summary, the ablation shows that our insights and the proposed solutions (in Sec. 3.1.2) are critical. Information about the position of the person in the 2D image improves position estimates in 3D and exploiting information from the other view substantially improves articulated pose estimates. Our proposed method utilizes both of these factors and thus results in an improvement of both the MPE and MPJPE.

Fig. 3.5 provides a qualitative analysis of the ablation study. Note that, from the camera view, the results of the different methods do not appear to differ significantly; human pose estimates projected into the camera can be misleading. In contrast, the 3D views clearly illustrate the errors in pose and shape.

3.3.2 Results on the Real Data

AirPose: After training on the synthetic data, we fine-tune the 'Baseline' and the proposed method, AirPose, on the real image data sequence of person 1 (and test on person 2) using a hardware-in-the-loop setup as described in Sec. 3.2. Since we do not have the camera extrinsic parameters (the global position of the DJI UAV is not accessible), we cannot transform the estimated SMPL-X parameters into a global reference frame to calculate any quantitative metric for comparison, as we did in the synthetic case. However, for both the baseline and AirPose, we show the plots of the X, Y and Z coordinates of the estimated position w.r.t. the UAV cameras in Fig. 3.7 of person 2 and Fig. 3.11 of person 1. The results in these plots demonstrate that the position estimate of the 'Baseline' method is significantly noisier than that of AirPose. In particular, the estimate in the Z direction (i.e. depth estimate from the camera) for UAV 2 has unrealistic variations when using the 'Baseline' method. In contrast, AirPose estimates a much more realistic depth and a smoother, more realistic trajectory of the person. In Fig. 3.8, we compare the depth estimate of our method with vanilla SPIN. The vanilla SPIN depth estimate is significantly noisier, similar to the baseline method, because in both cases the estimation is done on the cropped and scaled image.

AirPose$^+$: We further show the results of AirPose and AirPose$^+$ on the real data (see also `https://youtu.be/xLYe1TNHsfs`). Fig. 3.9 shows the estimated mesh overlaid on the images. We can see AirPose$^+$ improves the AirPose estimate of the articulated pose. Fig. 3.12 shows the estimated position of UAV 2 w.r.t. UAV 1, which was kept hovering in place. We use the estimated SMPL-X pose in each UAV frame to calculate the position of UAV 2 w.r.t. UAV 1. This position estimate is extremely sensitive to even minor estimation errors in any of the two views. A small error in the SMPL-X rotation estimate leads to a significant error in UAV 2's position. For person 1, we can see that the UAV 2 trajectory estimated by AirPose is a bit noisy. AirPose$^+$, however, improves this estimate further, as seen in Fig. 3.12 (top).

The sequence with person 2 contains several complex twisting poses in which the upper body rotates w.r.t. the lower body. For such poses, AirPose fails to estimate the correct root rotation of the person. This results in the estimated UAV 2 position by AirPose being very noisy, as seen in Fig. 3.12 (bottom). However, the position estimate of UAV 2 is significantly improved in this case by AirPose$^+$. Nevertheless, it still has a few sudden jumps. In addition to the twisting poses, errors in keypoint detection are also responsible for errors in person 2's sequence. Such errors are common with 2D detectors, e.g., left/right swap of 2D keypoints (discussed in Sec. 1.2.1).

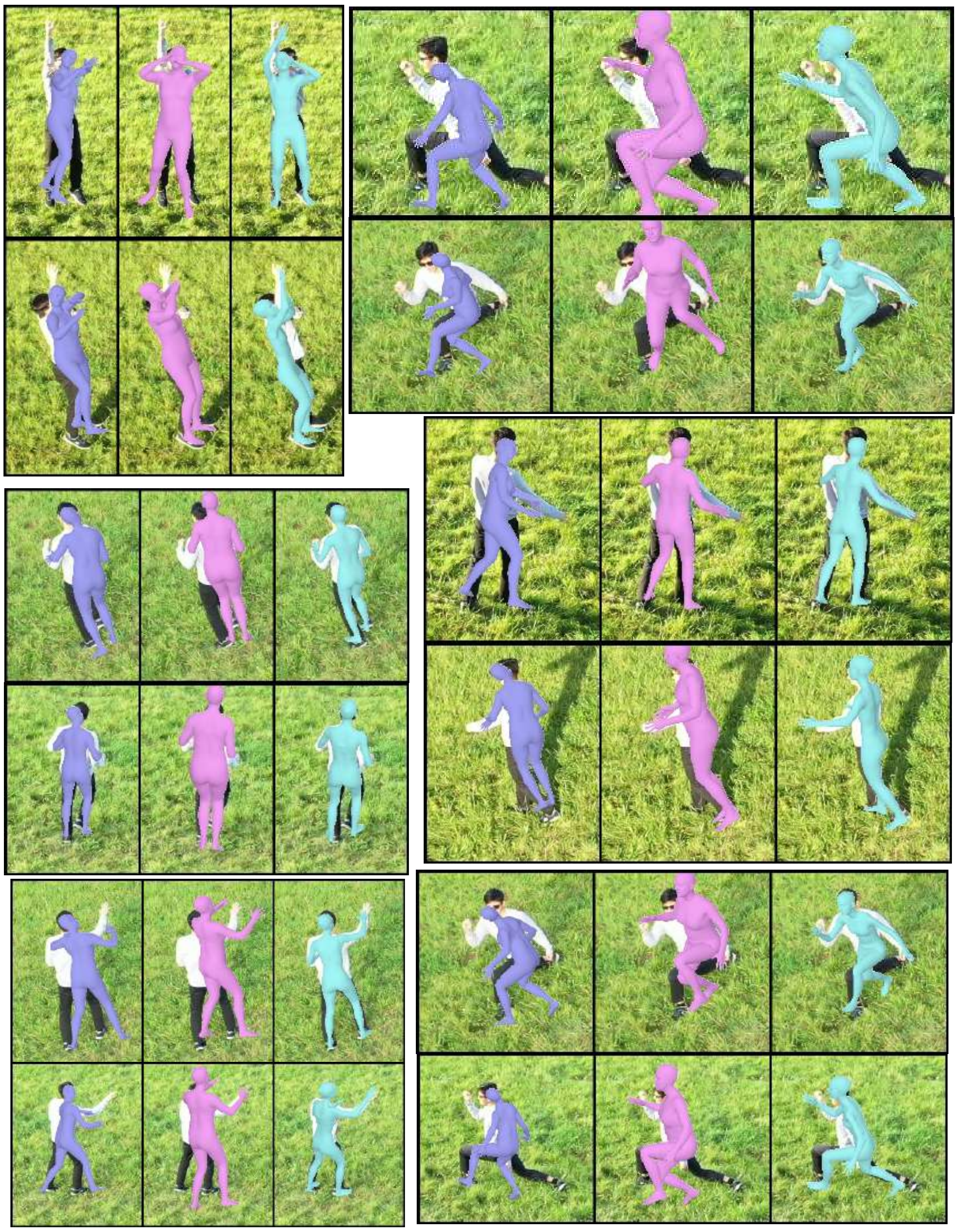


Figure 3.9: **Real data.** Baseline, AirPose and AirPose$^+$ results on real data samples. Each sample contains a 2x3 grid of images with the estimated mesh overlaid. Each row corresponds to one camera view and each column shows projected results from the baseline (blue), AirPose (pink) and AirPose$^+$ (cyan).

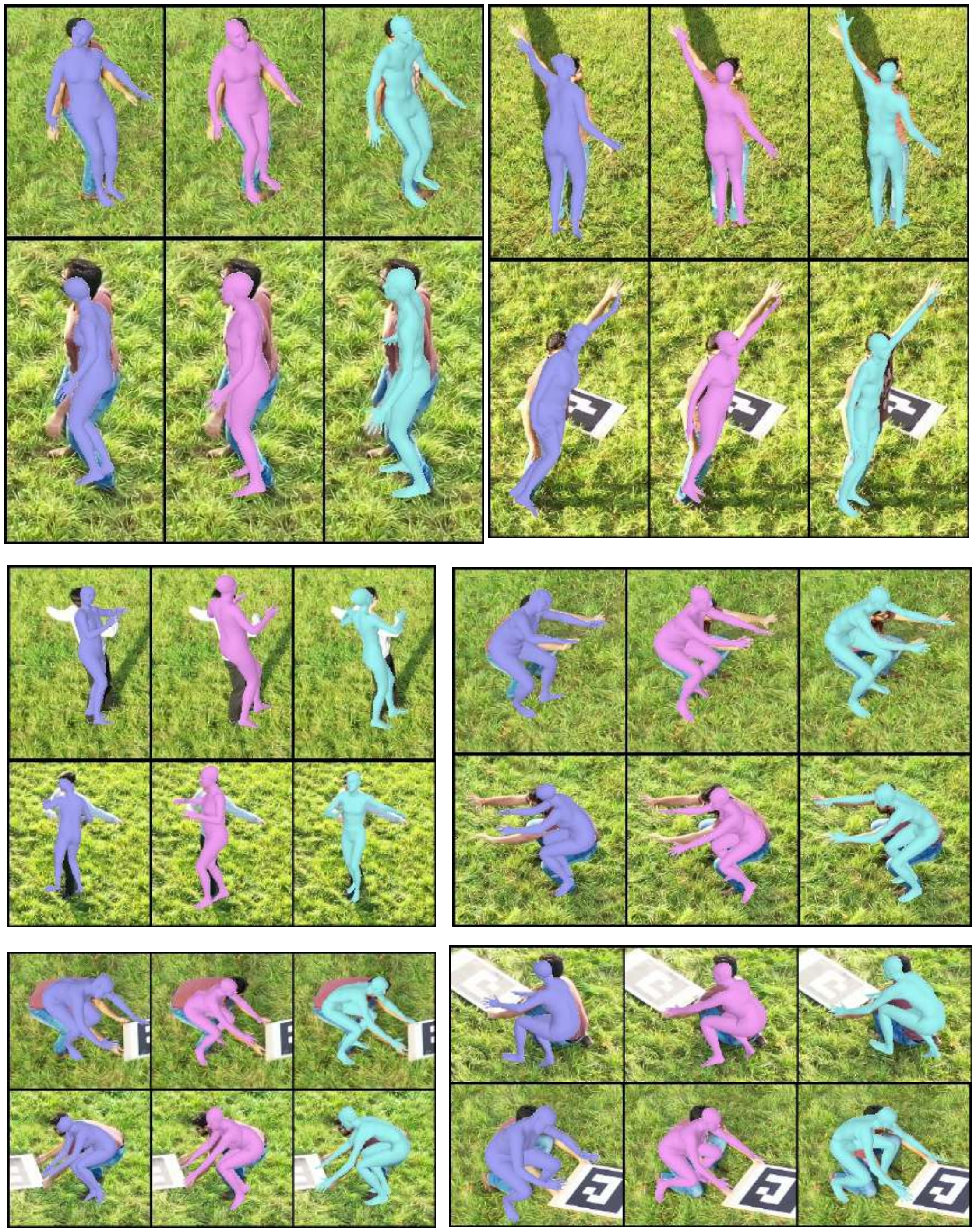
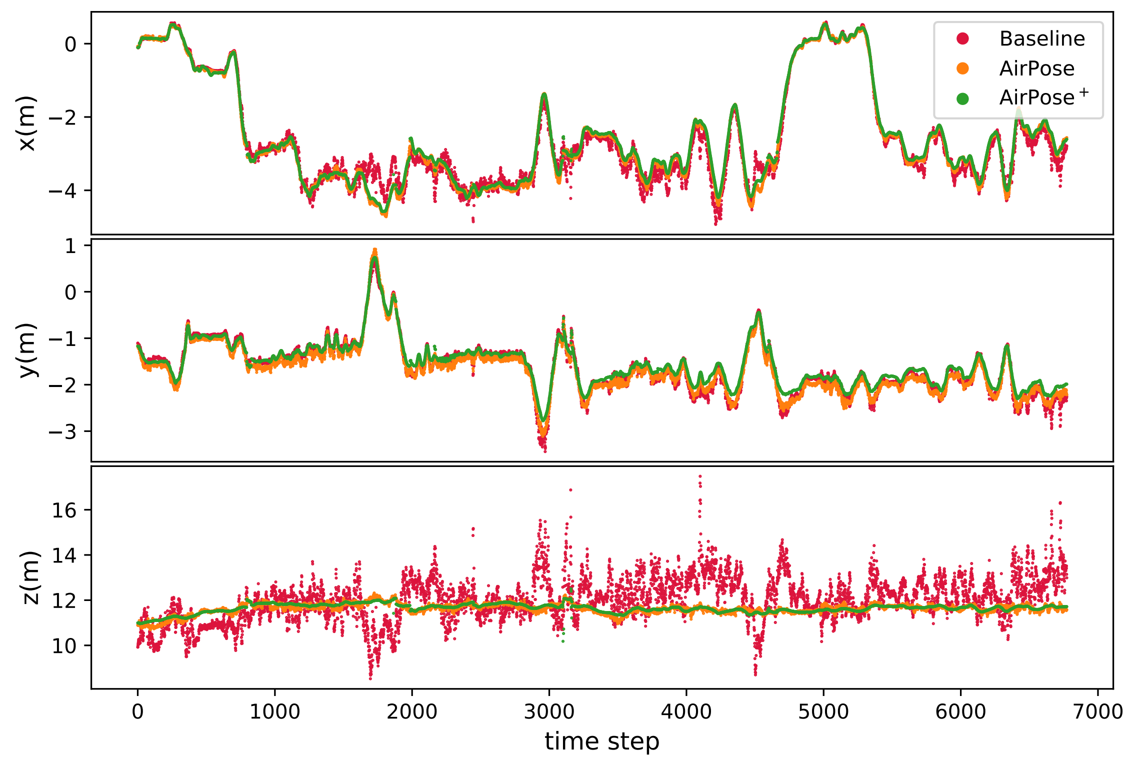


Figure 3.10: More results on real data samples. See the caption of Fig. 3.9 for details.



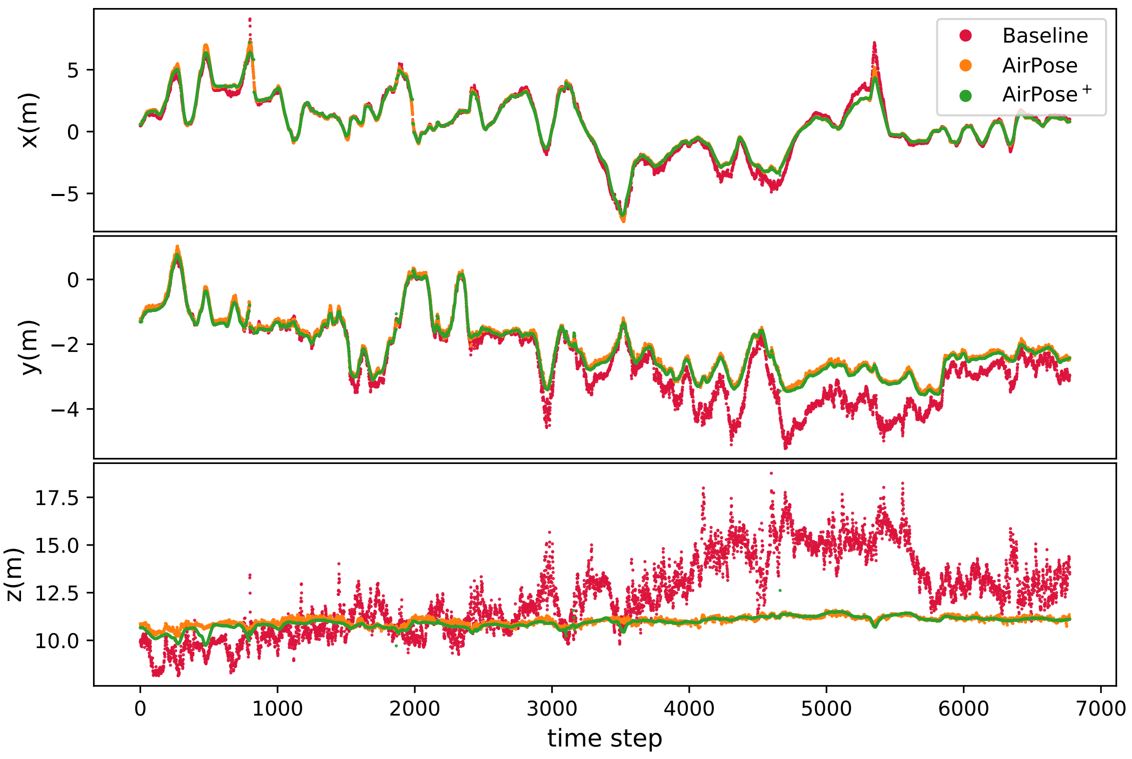
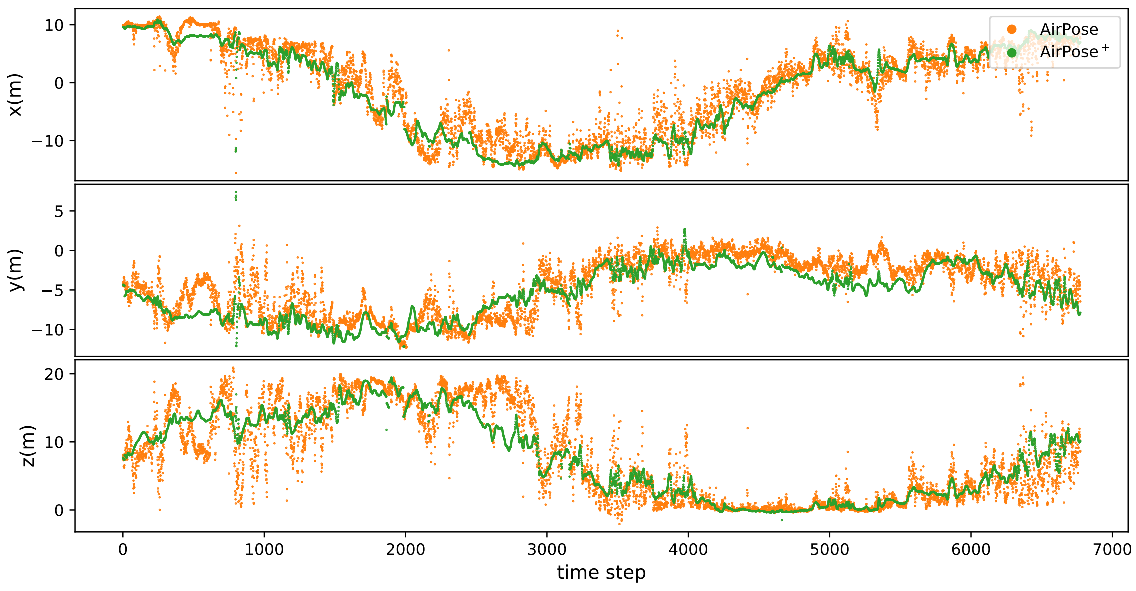


Figure 3.11: Estimated position of person 1 by the baseline method, AirPose and AirPose+ relative to UAV1 (top) and UAV2 (bottom).



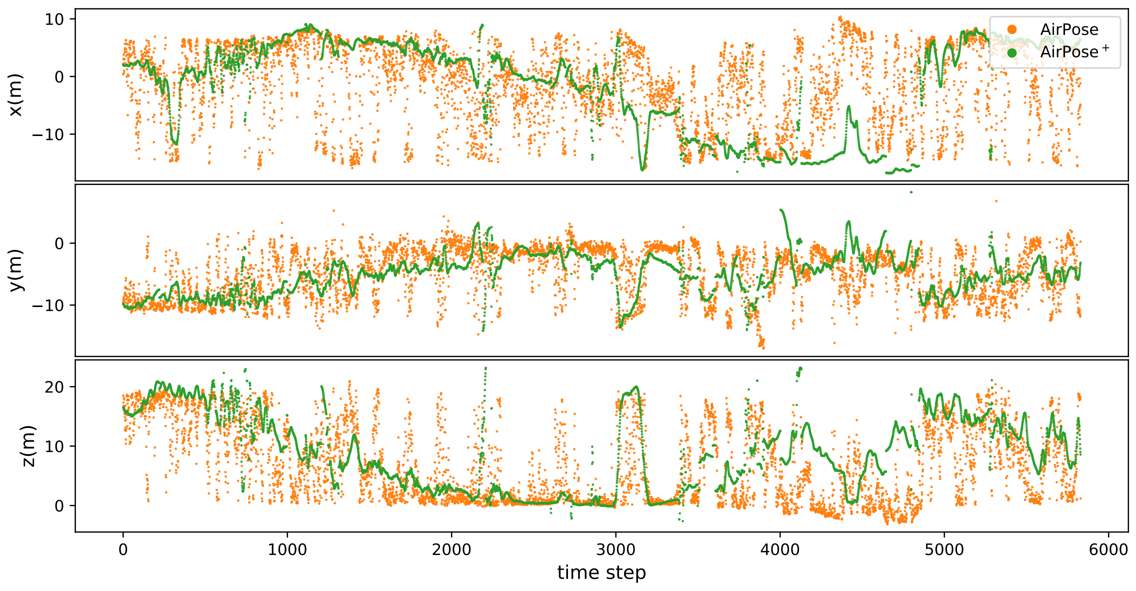


Figure 3.12: Estimated trajectory of UAV 2 w.r.t. UAV1 for the capture sequence with person 1 (top) and 2 (bottom).

3.4 Conclusion

We have presented a new approach for human 3D pose and shape estimation using multiple UAVs. Our novel network architecture is decentralized, distributed, lightweight and requires little inter-UAV communication, making it suitable for on-board deployment on UAVs. We demonstrated that our approach successfully fuses information from multiple viewpoints, improving pose estimates of both the person and the UAVs relative to the person, when compared to baseline methods. We introduced a powerful procedure to train such a network using large computer-generated datasets of synthetic images in virtual environments and to fine-tune on a small set of real images. Through a systematic evaluation on synthetic data, we show that AirPose is more accurate than the state-of-the-art method adapted for this problem. On real image data, captured by two UAVs, we show substantial qualitative improvement over the state-of-the-art method. Thus, AirPose overcomes significant problems currently limiting the deployment of aerial mocap systems in areas such as search and rescue and aerial cinematography.

Chapter 4

SmartMocap

In the previous chapters, we introduced Aircap-Pose-Estimator and AirPose. Aircap-Pose-Estimator can estimate the subject's and the camera poses relative to the world frame but requires onboard sensors and offline processing. AirPose addresses these problems by avoiding the camera extrinsic calibration and using a lightweight and distributed neural network to estimate the subject's and camera poses relative to each other. While such an approach is suitable for applications such as gesture controlling a UAV, this is not suitable for applications where the global motion of the subject is needed for placing him/her into a scene. Additionally, both of the above methods' estimations are per-frame, making them temporally incoherent. AirPose$^+$ provides a solution by using a simple loss which gives smooth motions, but it tends to over-smooth and does not take into account the dynamics of real human motion.

In this chapter, we present a system for outdoor human mocap using a set of extrinsically uncalibrated RGB cameras, where some cameras are static while others are moving. This system is quick to set up, as users can place the cameras and immediately start the capture session. Any camera can be moved during the mocap session to get better visibility of the subject, and each camera, using our method, extrinsically calibrates itself relative to the world using only the sparse 2D keypoints of the human body. Our system does not need a pre-calibration of the extrinsic parameters of the camera (pose of the camera in the 3D space). It, however, needs the camera intrinsics (related to the camera sensor and lens). Since these remain constant for any camera as long as the focal length does not change (e.g. by zooming), the intrinsic calibration needs to be done only once and then can be used in multiple mocap sessions.

Our mocap method takes in the synchronized videos from multiple RGB cameras and estimates the camera poses and the subject's motion in the global reference frame (see Fig. 4.1). The subject's motion is defined as the trajectory of human poses (articulated and global), and shape in all the frames. All the estimates are relative to a global reference frame, which is the ground projection of the human root joint onto the ground plane in the first frame. The ground plane is defined as the XY plane. First, we learn a probability distribution of human motion by training a variational autoencoder using a large human motion dataset (AMASS) (3). The state-of-the-art human motion prior (8) learns the distribution of pose transitions, which is defined as the difference between two

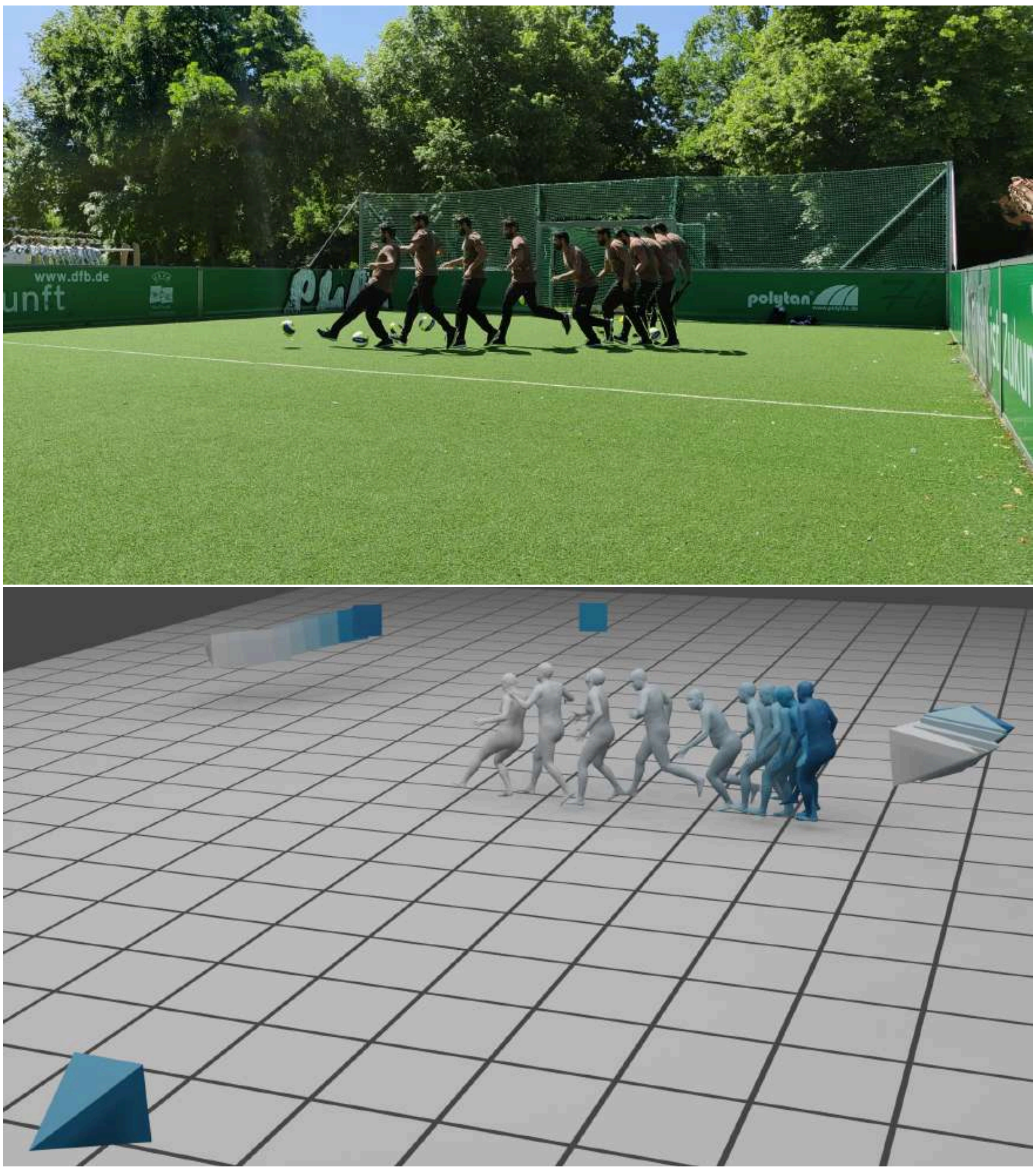


Figure 4.1: Multi-exposure image of a person playing football (top) and the reconstructed motion of the person and the cameras using our method (bottom).

consecutive poses. This is highly sensitive to noise in long-term motion generation/estimation. In contrast, we learn the distribution of the trajectory of body joint positions and joint angles relative to the above-defined world frame. The length of each motion sequence is fixed as 25 frames at the rate of 30 frames per second. We use this probability distribution to fit the SMPL human body model (98) to the sparse 2D keypoints in all the views. These 2D keypoints are obtained using the OpenPose 2D keypoint detector (16). While our motion prior encodes the distribution of human motion relative to the ground plane, the 2D keypoints contain information about the subject's articulated poses and the camera poses relative to the subject. In the optimization formulation, we directly optimize for the camera poses and the human poses in the world frame jointly and condition the human motion using our learned probability distribution of human motions. However, this formulation is highly non-convex and susceptible to converging to a local minimum. Therefore, we first initialize the human position as the mean of human motion in the learned latent space. The articulated human pose and the camera poses are initialized using the estimates of a human pose regressor (30). Since a motion sequence in our learned latent space is of a fixed length of 25 frames and starts from the origin, we take a multi-duration optimization approach. After the initialization stage, we run the optimization stage, where we split the full sequence into chunks of 25 frames and run optimization on these chunks independently. This optimization treats the starting of each chunk as the origin. In the next stage (stitching stage), we stitch these chunks together such that the last frame of a sequence aligns with the first frame of the next sequence. In the final stage, we run the optimization again on this stitched sequence to get the final estimates. For longer sequences, stitching the full sequence can accumulate noise in the orientation estimates, which leads to poor initialization for the next optimization stage. To avoid this, we stitch together a smaller number of chunks, perform optimization, stitch, optimize and iterate. This way, we slowly increase the duration at each stitching stage and perform the alternate stitching and optimization until the final optimization for the full sequence.

In summary, we have the following novel contributions:

- A human motion prior which encodes the global and articulated human motion relative to a global reference frame (ground plane).
- A multi-duration optimization method for estimating camera poses along with the human motion and shape relative to the ground plane using single/multiple extrinsically uncalibrated RGB cameras.

4.1 Method

4.1.1 Goal and preliminaries

Given synchronized image sequences of length T frames from C cameras looking at a moving person, the goal is to estimate the camera motion, the person's shape, and the person's motion, which is defined as the trajectory of the person's poses (articulated and global). We use the SMPL human body model to represent the human poses. As discussed in Sec. 1.2.3, SMPL is parameterized by joint angles ($\theta \in \mathbb{R}^{63}$), body shape parameters ($\beta \in \mathbb{R}^{10}$), root orientation ($\phi \in \mathbb{R}^3$) and root position ($\tau \in \mathbb{R}^3$). We use $N = 22$ body joints from SMPL, which includes 21 body joints and 1 root joint. We exclude the 2 hand joints from the original 24 joints in the SMPL model. Instead of representing the articulated pose as joint angles, we represent the subject's articulated pose at any time t in the latent space of VPoser ($v \in \mathbb{R}^{32}$). The full human motion is then represented as $((\tau_1, \phi_1, v_1), ..., (\tau_T, \phi_T, v_T), \beta)$. The position and orientation of a camera c at any time t is represented as $p_{c,t} \in \mathbb{R}^3$ and, $r_{c,t} \in \mathbb{R}^6$ respectively. Unless explicitly stated, we use the 6D representation (94) to represent the rotations. The camera motion for any camera c is represented as $((r_{c,1}, p_{c,1}), ..., (r_{c,T}, p_{c,T}))$. Our human motion prior uses a different representation of the body pose. The body pose x_t at any time t is the orientation and position of each body joint relative to the world frame, i.e. $x_t \in \mathbb{R}^{22*(6+3)}$. The origin of the world frame is defined as the ground projection of the SMPL root joint in the first frame and the ground plane is defined as the XY plane. The motion prior encodes the fixed length of 25 consecutive poses, thus, the motion sequence for the motion prior is represented as $\mathbf{x} = (x_1, \ldots, x_{25})$. We represent the estimated value of any parameter by putting a tilde over it, e.g. $\tilde{\mathbf{x}}$ is the estimated value of $\mathbf{x}$.

4.1.2 Human motion prior

We use a VAE to learn a distribution in the latent space of human motion sequences of a fixed length. Each motion sequence consists of 25 consecutive human body poses at the rate of 30 frames per second. The origin of the world frame is the ground projection of the root joint in the first frame. The forward-facing direction of the SMPL root joint in the first frame is aligned with the +Y axis of the origin.

Training data

We use the AMASS dataset (3) to train our network. AMASS is a collection of multiple human mocap datasets, unifying them with the SMPL body representation. We follow the preprocessing steps in HuMoR (8) which removes the motion sequences where the person's feet are skating or sliding over the static ground plane. In such motions, the person does not interact with a stationary ground plane but with an object such as a treadmill or skates. The preprocessing step also changes the frame rate to 30 FPS and gives out a total of 11893 motion sequences. We randomly select 25 consecutive frames from any of these sequences and canonicalize them such that the origin is the ground projection of the root joint at the first frame and the person's forward direction is aligned with the origin's +Y axis. Even though the shape of the subjects in AMASS varies, we follow (8) and keep the body shape constant to the mean shape i.e. $\beta = 0$. This reduces the complexity of the model by ignoring the body shape variations at the expense of some possible artefacts such as foot-skating.

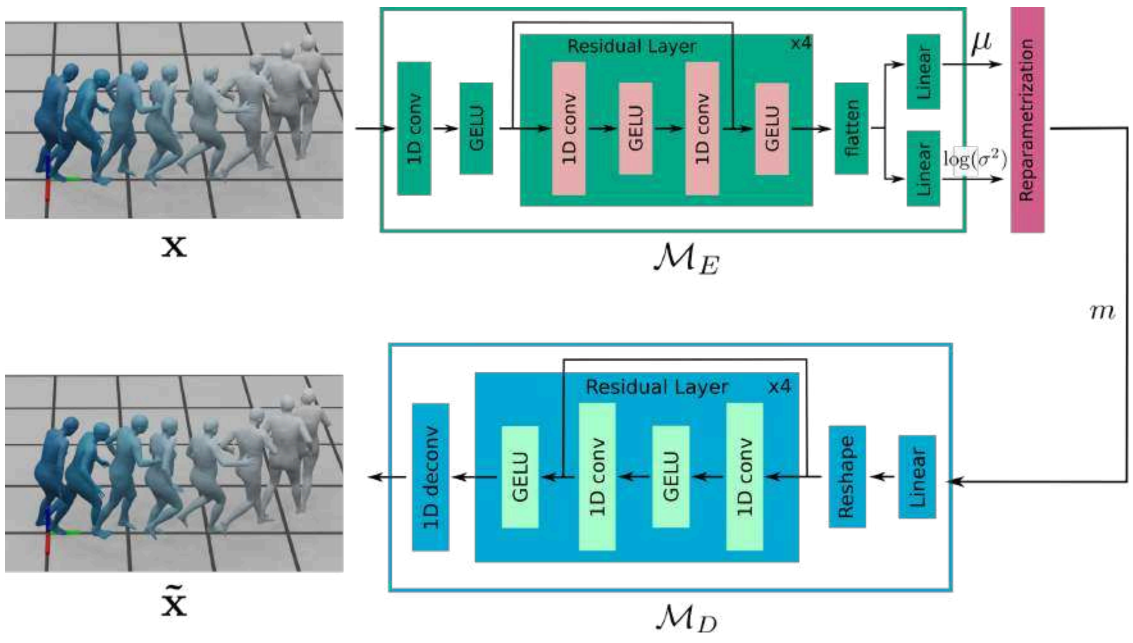


Figure 4.2: Architecture of our human motion prior network.

Model architecture and training

We use a convolutional architecture for both the encoder and the decoder networks, as shown in Fig. 4.2. The encoder ($\mathcal{M}_E$) consists of a 1D convolutional (conv) layer at the input and 4 identical residual blocks. We modify the ResNet (99) residual blocks to create these blocks. We replace the 2D convolutions with 1D convolutions. We further replace the ReLU units with the GELU units within the blocks. We also place a GELU unit after the input 1D conv layer and each of the residual blocks. The output dimension of the first conv layer is 1024. The input and output dimension of each residual block is 1024. Furthermore, two linear layers transform the output of the last residual block to the mean ($\mu \in \mathbb{R}^{1024}$) and log of variance ($\log(\sigma^2) \in \mathbb{R}^{1024}$) of the Gaussian distribution in the latent space, from which the latent value ($m \in \mathbb{R}^{1024}$) is sampled using the reparametrization trick (100). The decoder ($\mathcal{M}_D$) architecture also consists of 4 consecutive residual blocks, similar to the ones in the encoder. The first residual block acts as the input layer, and there is a deconvolutional layer at the output of the decoder.

We train our motion VAE network using a combination of reconstruction ($\mathcal{L}_{rec}$) and KL divergence losses ($\mathcal{L}_{KL}$).

$$\mathcal{L} = \mathcal{L}_{rec} + w_{kl}\mathcal{L}_{KL}, \tag{4.1}$$

where

$$\mathcal{L}_{rec} = ||\mathbf{x} - \tilde{\mathbf{x}}||^2 \quad \text{and} \quad \mathcal{L}_{KL} = -\frac{0.5}{1024}\sum_{i}^{1024}(1 + \log(\sigma_i^2) - \sigma_i^2 - \mu_i^2). \tag{4.2}$$

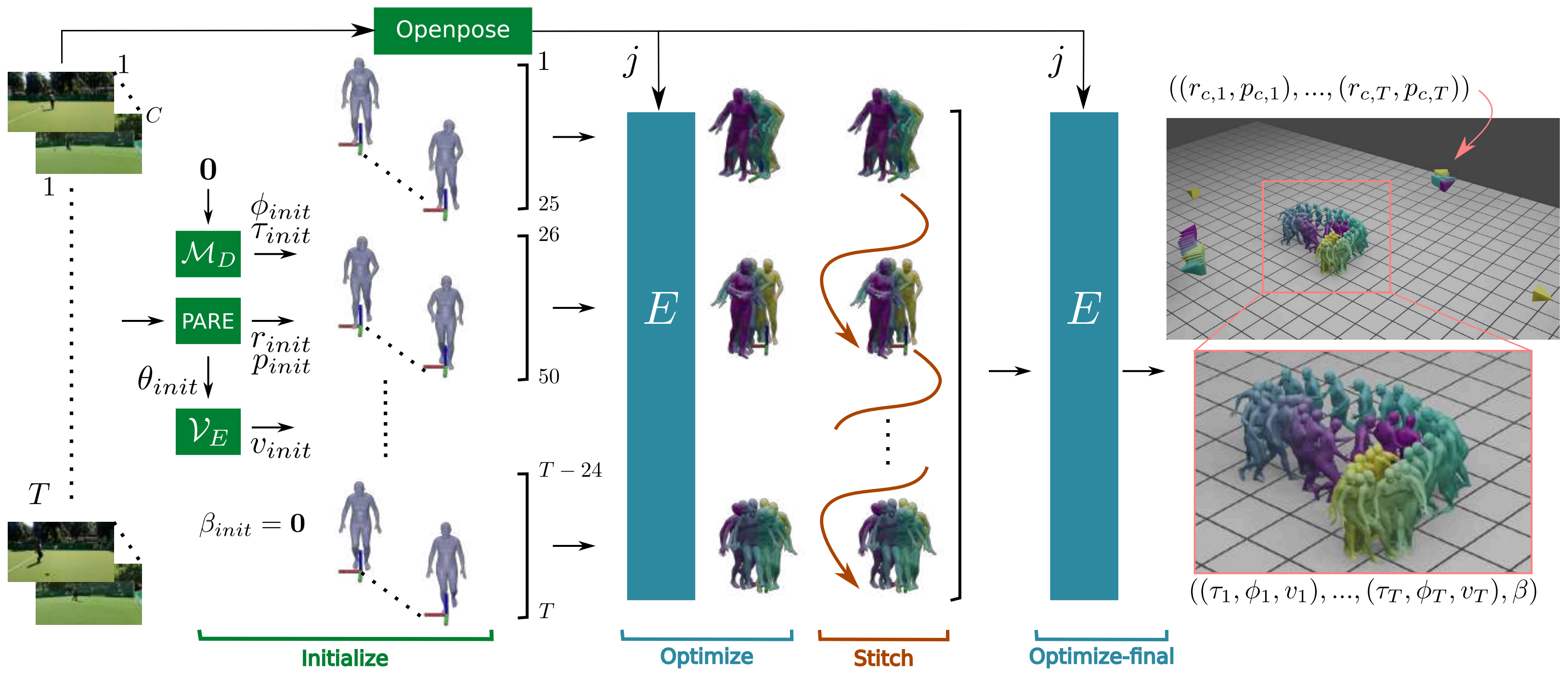


Figure 4.3: Our method takes in synchronized images of a moving person from multiple extrinsically uncalibrated cameras, and processes them in 4 steps 1) Initialize, 2) Optimize, 3) Stitch and 4) Optimize-final to give the motion of the person and the cameras in the world frame.

We employ a 20-epoch cyclic annealing scheme for the parameter w_{kl} (101). Initially, the value of w_{kl} starts at 0 and increases linearly with the training epochs. After 10 epochs, the value reaches 1 and stays constant for another 10 epochs. The value again drops to 0 and the cycle continues.

4.1.3 Camera and human pose estimation

First, we use OpenPose (16) to detect 2D keypoints of the subject in each image. Then we use them in our method, which consists of the following steps, 1) Initialize, 2) Optimize, 3) Stitch and 4) Optimize-final (see Fig. 4.3).

- **Step 1: Initialize.** We initialize the SMPL parameters and camera poses for each frame using the results from PARE (30). PARE gives the camera pose and the person's articulated pose for each image. We take the mean of the articulated poses in all the views (θ_{init}), project it to the VPoser latent space (v_{init}) and use it as the initial articulated pose of the subject. Using VPoser helps in representing the SMPL articulated and global poses separately, which allows us to optimize these parameters in different phases. Additionally, we use its latent space for an additional loss on articulated pose. For the initialization of SMPL position (τ_{init}) and orientation (ϕ_{init}) relative to the ground plane, we use the decoded output of the mean value in the motion prior latent space. PARE gives the camera position and orientation relative to the person. We use the initialized pose of the person to calculate the position (r_{init}) and orientation (p_{init}) of the cameras relative to the ground plane. Since PARE assumes weak-perspective camera, we use the method in (102) to transform the SMPL position estimate in the actual camera. The SMPL shape (β_{init}) is initialized with a vector of zero values.

- **Step 2: Optimize.** We estimate the motion of the person in small intervals (25 frames). We split the full sequence into chunks of length 25 and run the optimization for each chunk independently. This optimization is done in three phases. In the first phase, we optimize the camera poses only. In the second phase, we optimize camera poses along with SMPL position and orientation. In the final phase, we optimize all the parameters, except the SMPL position and orientation in the first frame. The initial SMPL position and orientation in the first frame act as the pivot for all the optimizing parameters. Since optimizing all the parameters together is more likely to get stuck in a local minimum, we follow previous work (81) and do optimization in three phases.

- **Step 3: Stitch.** We stitch together the estimated motions. Since the origin for each chunk is defined as the ground projection of the root joint, we stitch consecutive sequences together such that the root ground projection of the last frame of a chunk is aligned with the first frame of the next chunk. For very long sequences, we stitch together fewer sequences, optimize, stitch and repeat until all the sequences are stitched.

- **Step 4: Optimize-final.** In the final optimization step (optimize-final), we again optimize all the parameters for the fully stitched sequences in three phases, the same as in the previous optimization stage. This step is the final optimization step if all the sequences are stitched. For very long sequences, we stitch together fewer chunks instead of all. Then we optimize and repeat the stitching and optimization cycle until the whole sequence is done.

In all the optimization stages, we minimize the same loss function, which is a weighted combination of multiple loss terms. It is given as

$$\begin{aligned} E = w_{2D}E_{2D} + w_m E_m + w_{3DS}E_{3DS} + w_{COS}E_{COS} + w_{CPS}E_{CPS} \\ + w_\beta E_\beta + w_v E_v + w_{GP}E_{HGP} + w_{CGP}E_{CGP}. \end{aligned} \tag{4.3}$$

The component E_{2D} is the 2D reprojection loss given as

$$E_{2D} = \frac{1}{NT}\sum_{n,c,t} w_{n,t}||\Pi(r_{c,t}, p_{c,t}, \mathcal{J}_n(\mathcal{V}_D(z_t), \tau_t, \phi_t, \beta)) - j_{c,n,t}||^2, \tag{4.4}$$

The loss component E_m is the motion prior loss given as

$$E_m = \sum_{t}^{T-25} ||\mathcal{M}_{E_\mu}(\mathcal{V}_D(v_{t:t+25}), \tau_{t:t+25}, \phi_{t:t+25})||^2, \tag{4.5}$$

where $\mathcal{M}_{E_\mu}$ is the μ part of the motion prior encoder (see Fig. 4.2). The loss component E_{3DS} is a temporal smoothing term for the 3D joint positions. It is given as

$$\begin{aligned} E_{3DS} = \sum_t ||\mathcal{J}(\mathcal{V}_D(v_t), \tau_t, \phi_t, \beta) - \\ \mathcal{J}(\mathcal{V}_D(v_{t-1}), \tau_{t-1}, \phi_{t-1}, \beta)||^2. \end{aligned} \tag{4.6}$$

E_{COS} and E_{CPS} are the camera motion smoothing terms for camera orientation and position, respectively. Following the previous work (81), we use a simple L2 loss on the positions and the 6D representation of the camera orientations. They are given as

$$E_{COS} = \frac{1}{CT}\sum_{c,t} ||r_{c,t} - r_{c,t-1}||^2 \tag{4.7}$$

and

$$E_{CPS} = \frac{1}{CT}\sum_{c,t} ||p_{c,t} - p_{c,t-1}||^2. \tag{4.8}$$

E_β and E_v are the SMPL shape and Vposer regularization terms (81), given as

$$E_\beta = ||\beta||^2 \tag{4.9}$$

and

$$E_v = ||v||^2. \tag{4.10}$$

E_{HGP} and E_{CGP} are the ground penetration terms for both the human and the cameras. These terms avoid the scenarios where the cameras or the human goes below the ground plane. These are given as

$$E_{HGP} = \frac{1}{T}\sum_t max(0, \mathcal{J}^z(\mathcal{V}_D(z_t), \tau_t, \phi_t, \beta_t)) \tag{4.11}$$

and

$$E_{CGP} = \frac{1}{CT}\sum_{c,t} max(0, p^z_{c,t}), \tag{4.12}$$

where, $\mathcal{J}^z$ and p^z are the vertical (z) component of the 3D joint positions and the camera positions, respectively. In all the above equations, T is replaced with 25 in **Step 2: Optimize**.

4.2 Experiments and results

4.2.1 Datasets

We evaluate our method using a sequence taken from each of the three different types of datasets 1) RICH dataset (31), 2) AirPose real-world dataset (81), and 3) Our smartphone dataset, which we introduce here.

1) RICH is collected using 7 static and one moving cameras. It has the ground truth poses of the person and the camera poses for the static cameras. The GT poses of the moving camera are not available.

2) The AirPose real-world data is the same as introduced in Chapter 3.

3) Our smartphone data is collected using 4 smartphone cameras, where two of the cameras are static while the other two are moving. The data is collected when the subject is playing with a football in a small field. The camera setup can be seen in Fig. 4.4. We show the rest of the three cameras and the subject in the frame of camera *Cam1*. *Cam1* and *Cam3* are static, and *Cam2* and *Cam4* are moved along the boundary walls of the arena. We used an open-source app OpenCamera (103) for collecting the video data on the smartphones. Each smartphone records the videos at 30 FPS. Since the frame rates are the same and constant, we manually synchronize the videos by synchronizing just one frame. As with the AirPose real-world dataset, our smartphone dataset also does not have the GT poses of the subject and the cameras.

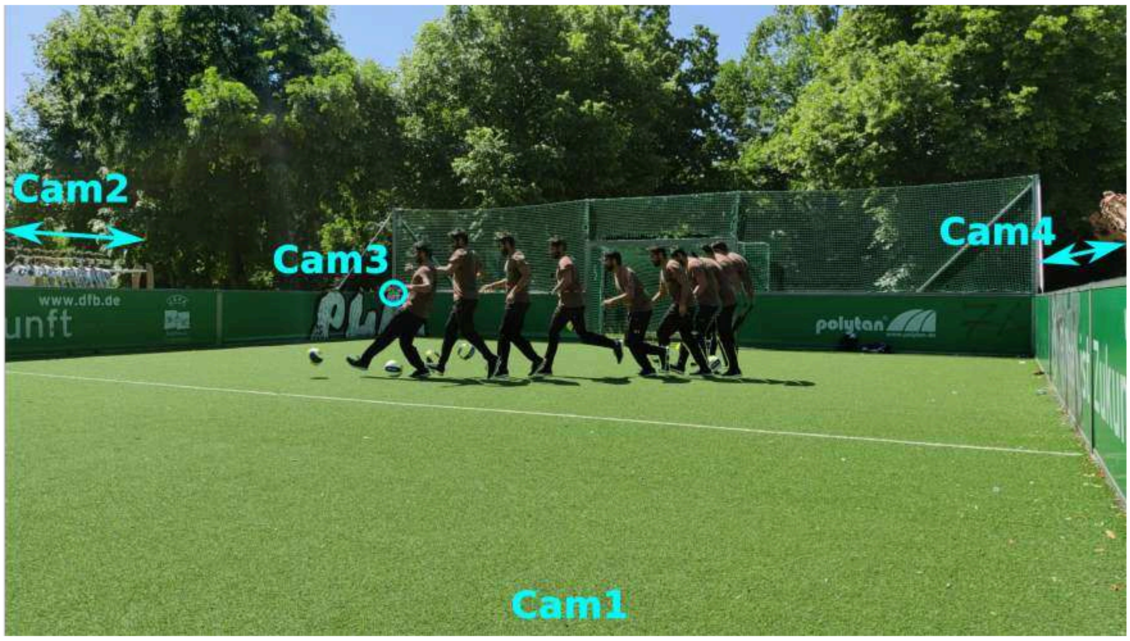


Figure 4.4: Camera setup used to collect our smartphone dataset.

4.2.2 Metrics

We use the following metrics to quantitatively evaluate the reconstruction of our method on the RICH dataset.

1) **Mean camera position error (MCPE):** This is the mean value of the distance between the estimated and the GT position of all the cameras. It is given as

$$MCPE = \frac{1}{C}\sum_{c} ||p_c - \tilde{p_c}||. \tag{4.13}$$

2) **Mean camera orientation error (MCOE):** This metric is the mean geodesic distance between the estimated camera orientation and the GT on the 3D manifold of rotation matrices (104). It is given as

$$MCOE = \frac{1}{C}\sum_{c} \arccos\left(0.5 * (Tr(\tilde{r_c} r_c^\top) - 1)\right), \tag{4.14}$$

where r_c is the matrix representation of the camera orientation.

3) **Mean position error (MPE):** This is the mean value of the distance between the estimated SMPL root position parameter and its GT value provided by the RICH dataset. It is given as

$$MPE = \frac{1}{T}\sum_{t} ||\tau_t - \tilde{\tau_t}||. \tag{4.15}$$

4) **Mean orientation error (MOE):** This metric is the mean geodesic distance between the estimated SMPL root orientation and the GT on the 3D manifold of rotation matrices (104). It is given as

$$MOE = \frac{1}{T}\sum_t \arccos\left(0.5 * (Tr(\tilde{R_{\phi_t}} R_{\phi_t}^\top) - 1)\right), \tag{4.16}$$

where R_{ϕ_t} is the matrix representation of the SMPL root orientation ϕ at any time step t.

5) **Root-aligned mean per-joint position error (RA-MPJPE):** This metric is to quantitatively evaluate the articulated pose estimate (80). It is the distance between the estimated 3D joints and their corresponding values when the SMPL position, orientation of the root joint and shape are aligned with their corresponding GT. Since the goal is to measure the error in joint position due to the difference in articulated pose, we set the shape parameters to be the same (i.e. a zero vector). If we do not do this, the metric will contain the error due to differences in both the shape and articulated pose. The error is given as

$$RA\text{-}MPJPE = \frac{1}{NT}\sum_n ||\mathcal{J}_n(\theta_t) - \mathcal{J}_n(\tilde{\theta}_t)||. \tag{4.17}$$

6) **Mean per-vertex position error (MPVPE):** This metric evaluates the shape estimate relative to the GT. We compute the body vertices using only the estimated and GT shape parameters and then calculate the mean distance between the corresponding vertices as

$$MPVPE = \frac{1}{V}\sum_i ||\mathcal{S}_i(\beta) - \mathcal{S}_v(\tilde{\beta})||, \tag{4.18}$$

where $\mathcal{S}_v$ is the SMPL vertices regressor function giving the position of vertex i and V is the number of vertices in the SMPL model.

Metrics computation for moving camera in RICH

RICH dataset is collected using 7 static cameras and 1 moving camera. The GT poses of the static cameras are provided in the dataset, however, they are not available for the moving camera (*cam8*). Therefore, we estimate a pseudo-GT using a SLAM method (COLMAP (10; 11)). We use the bound boxes provided in the RICH dataset to mask that area and then use these masked images as COLMAP input. COLMAP gives the poses of *cam8*, however, they are not in the same reference frame as estimated by our method. Therefore, we align them using Procrustes alignment (105)s. First, we normalize the positions estimated by our method and COLMAP, then find optimum rotation and scaling parameters which align the normalized COLMAP estimates to the normalized camera position estimates by our method. Then we denormalize both of them and use them to compute the MCPE metric. We use the same rotation parameter to transform the estimated camera rotations by COLMAP too, and then compute the MCOE metric. The alignment procedure is explained below in detail.

Say $\hat{P} \in \mathbb{R}^{N\times 3}$ is the position matrix such that the row t of this matrix is the position of the camera estimated by COLMAP for N frames. Similarly, $\tilde{P} \in \mathbb{R}^{Nx3}$ is the position matrix estimated by our method. We align these positions using the implementation in (106). However, it requires the matrix $\hat{P}$ and $\tilde{P}$ to be normalized. The normalized matrices $\hat{P}_{norm}$ and $\tilde{P}_{norm}$ are computed as

$$\hat{P}_{norm} = (\hat{P} - \mu_{\hat{P}})/||\hat{P}|| \tag{4.19}$$

and

$$\tilde{P}_{norm} = (\tilde{P} - \mu_{\tilde{P}})/||\tilde{P}||. \tag{4.20}$$

$\mu_{\hat{P}} \in \mathbb{R}^3$ and $\mu_{\tilde{P}} \in \mathbb{R}^3$ are the mean of all the 3D positions in $\hat{P}$ and $\tilde{P}$ respectively. $||\hat{P}||$ and $||\tilde{P}||$ are the Frobenius norms of $\hat{P}$ and $\tilde{P}$. The output of Procrustes alignment (106) is a rotation matrix R and a scale parameter s. The normalized and aligned form P_{norm} of COLMAP estimated position matrix $\hat{P}_{norm}$ is computed as

$$P_{norm} = (\hat{P}_{norm}R^T) * s. \tag{4.21}$$

Then we compute P which is the denormalized form of COLMAP estimated position matrix aligned with $\tilde{P}$. Since P_{norm} is transformed and aligned with $\tilde{P}_{norm}$, we use the mean and Frobenious norm corresponding to $\tilde{P}$ and compute P as

$$P = P_{norm} * ||\hat{P}|| + \mu_{\hat{P}}. \tag{4.22}$$

We use the same matrix R returned by Procrustes algorithm (106) to transform the camera rotations estimated by COLMAP. We represent COLMAP estimated camera rotation matrix at frame t as $\hat{r}_t$, and the transformed rotation matrix r_t is computed as

$$r_t = R\hat{r}_t \tag{4.23}$$

We compute the MCPE and MCOE metrics for *cam8* in the RICH dataset as

$$MCPE = \frac{1}{N}\sum_t^N ||p_t - \tilde{p}_t|| \tag{4.24}$$

and

$$MCOE = \frac{1}{N}\sum_t^N \arccos\left(0.5 * (Tr(\tilde{r}_t r_t^\top) - 1)\right), \tag{4.25}$$

where, p_t and $\tilde{p}_t$ are the t^{th} rows of P and $\tilde{P}$.

Cameras	MCPE (cm)	MCOE (rad)	MPE (cm)	MOE (rad)	RA-MPJPE (cm)	MPVPE (cm)
C_1	72.68	0.2	10.89 ± 7.31	0.27 ± 0.1	6.6 ± 4.92	3.07 ± 1.45
$C_{1,8}$	55.85	0.14	7.61 ± 4.14	0.22 ± 0.06	6.24 ± 5.02	3.13 ± 1.5
$C_{1,2,8}$	90.46	0.17	12.72 ± 2.93	0.23 ± 0.07	5.97 ± 4.91	3.0 ± 1.38
$C_{1,2,3,8}$	89.68	0.14	11.67 ± 2.59	0.2 ± 0.07	5.73 ± 4.8	3.02 ± 1.38
$C_{1,..,4,8}$	95.03	0.16	12.39 ± 2.69	0.22 ± 0.07	5.68 ± 4.68	2.91 ± 1.3
$C_{1,..,5,8}$	93.74	0.16	12.49 ± 2.74	0.2 ± 0.07	5.66 ± 4.65	2.91 ± 1.32
$C_{1,..,6,8}$	92.43	0.17	13.42 ± 2.85	0.2 ± 0.06	5.87 ± 4.77	2.31 ± 0.92
$C_{1,..,8}$	88.13	0.18	13.69 ± 3.05	0.19 ± 0.06	6.03 ± 4.94	1.86 ± 0.79

Table 4.1: Evaluation of our method using multiple camera configurations. Camera no. 1-7 are static in the RICH dataset with available GT, and camera no. 8 is moving but GT is not available. Hence, the MCPE and MCOE metrics for rows 2-9 do not include camera 8.

4.2.3 Results and discussion

RICH dataset

We show the evaluation metrics and their corresponding standard deviation values of our method on RICH in Table 4.1 and 4.3. The results of our method using all 8 cameras are shown in the last row of Table 4.1 ($C_{1,..,8}$). We also compare the performance of our method with multiple camera configurations. In the first row, we show the results when only the first camera is used (C_1). Next, we add camera 8, which is moving, and the results are shown in the second row ($C_{1,8}$). We keep adding static cameras one at a time and show the results in further rows. In general, we see improvements in both the RA-MPJPE and the MPVPE metrics as we add cameras. This shows that adding more cameras helps improve the human articulated pose estimates. Adding more views helps in handling occlusions better, and we see in Fig. 4.7 that the RA-MPJPE improves with more camera views but gets saturated after 4 views. This suggests that in this case, cameras 1,2,3 and 8 are sufficient to resolve any uncertainty in the person's articulated pose due to occlusions, and adding more views does not provide any additional information. In Table 4.2, we show the MCPE and MCOE metrics for the moving camera. As discussed in Sec. 4.2.2, we compute the pseudo-GT for the moving camera by estimating its poses using COLMAP and then aligning them with our estimated poses using the Procrustes alignment. We observe that camera configuration $C_{1,2,3,8}$ is again the best camera configuration, considering both the MCPE and MCOE metrics. In Fig. 4.5, we show the camera positions estimated by our method and pseudo-GT for every camera configuration. We can see that as we add more cameras, our estimates are better aligned with the pseudo-GT. However, as we incrementally add static cameras 4, 5, 6 and 7, we observe sudden changes in our estimated position of *cam8* during frames 450-550. This

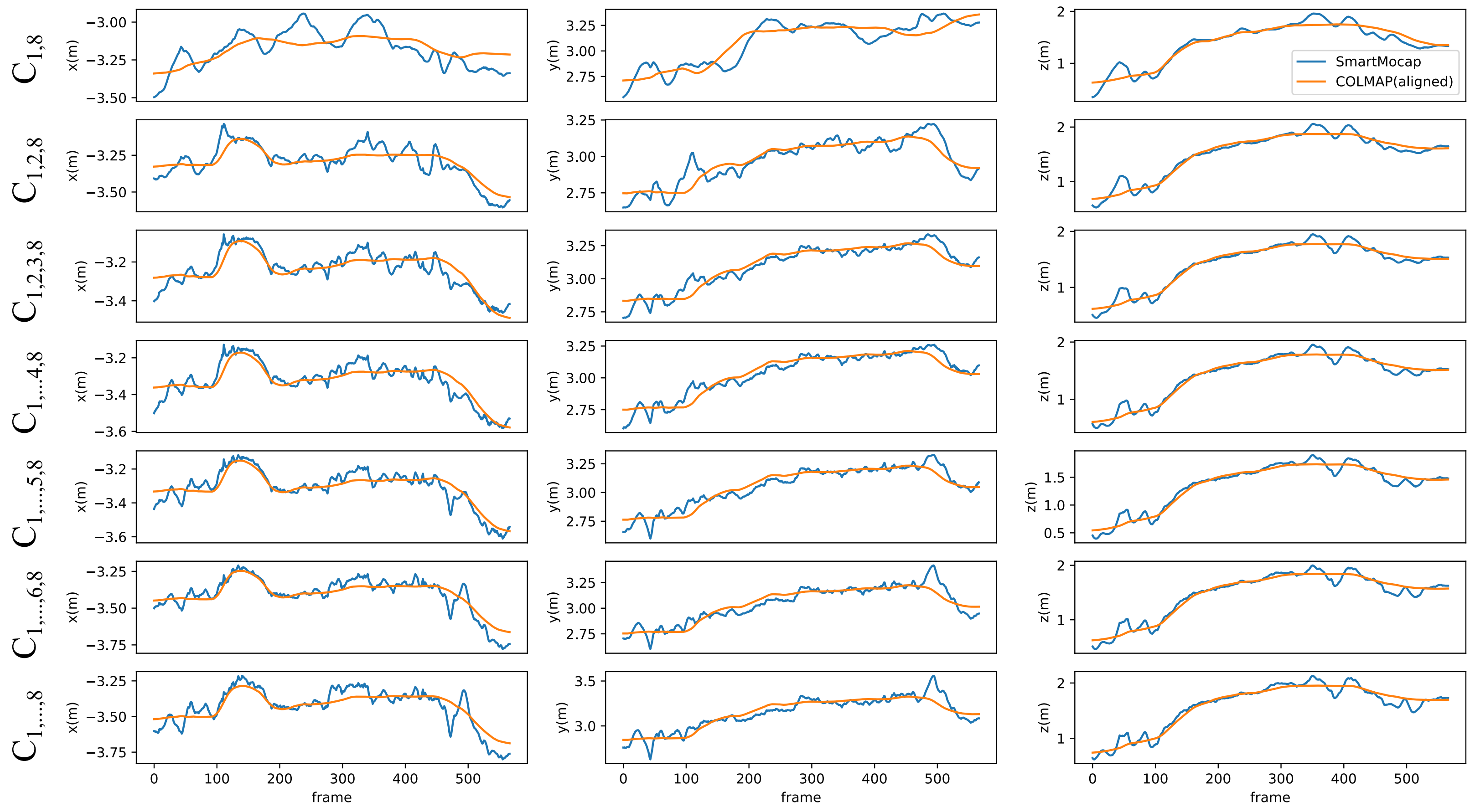


Figure 4.5: Plots showing estimated positions of the moving camera by SmartMocap(blue) and COLMAP(orange) after Procrustes alignment for different camera configurations.

Cameras	MCPE (cm)	MCOE (rad)
$C_{1,8}$	15.14±7.16	1.58±0.002
$C_{1,2,8}$	10.53±6.32	0.16±0.017
$C_{1,2,3,8}$	8.59±5.53	0.08±0.011
$C_{1,..,4,8}$	8.70±5.17	0.10±0.011
$C_{1,..,5,8}$	9.34±5.40	0.07±0.011
$C_{1,..,6,8}$	10.21±5.96	0.08±0.015
$C_{1,..,8}$	11.39±6.32	0.06±0.016

Table 4.2: MCPE and MCOE metric for camera no. 8 (moving camera). GT position and orientation are obtained using COLMAP(10; 11) and aligned with the estimated position using Procrustes alignment.

Camera	MCPE (cm)	MCOE (rad)	MPE (cm)	MOE (rad)	RA-MPJPE (cm)	MPVPE (cm)
GLAMR (9)	250.93	0.27	24.07 ± 4.96	0.48 ± 0.28	9.66 ± 9.51	2.84 ± 1.12
HuMoR (8)	90.09	**0.17**	30.32 ± 8.12	0.48 ± 0.39	10.82 ± 8.14	4.2 ± 2.09
C_1 (ours)	**72.68**	0.2	**10.89** ± **7.31**	0.27 ± 0.1	6.6 ± 4.92	3.07 ± 1.45
$C_{1,..,8}$ (ours)	88.13	0.18	13.69 ± 3.05	**0.19** ± **0.06**	**6.03** ± **4.94**	**1.86** ± **0.79**

Table 4.3: Comparison of our method ($C_{1,..,8}$), monocular version of our method (C_1) and state-of-the-art monocular methods HuMoR (8) and GLAMR (9) on RICH dataset.

is because the person quickly rotates around 180 degrees during these frames, resulting in high variation in *cam8* position. This is the reason we see the MCPE value increasing as cameras 4, 5, 6 and 7 are added incrementally.

Multi-view methods usually perform better than monocular methods when the camera extrinsic parameters are available. It is because extra cameras bring in extra information about the subject's articulated pose (via RGB image) and the camera pose (extrinsic parameters), which results in improving the global and articulated pose estimate of the subject. Adding an extra view only gives information about the articulated pose of the subject and the relative poses of the camera and the subject. That is why the metrics related to the global poses do not improve as more cameras are added. We further explain this by taking a sample frame from our Smartphone dataset. In Fig. 4.6, we show the results of step 3 (of our method) on a single frame which is overlapping in two different

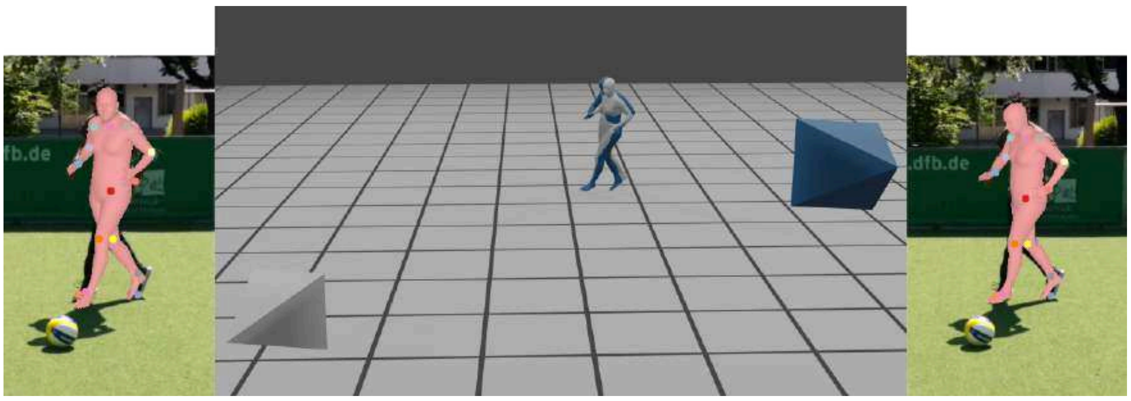


Figure 4.6: The image in the middle shows the step 3 optimization results of our method on a single frame (frame number 25). Blue colored results are from the optimization chunk (1-25) while white colored are from the optimization chunk (25-49). The image overlays are shown on the left and the right for the white and the blue estimates, respectively. The 2D keypoints from the detector are also shown in these images.

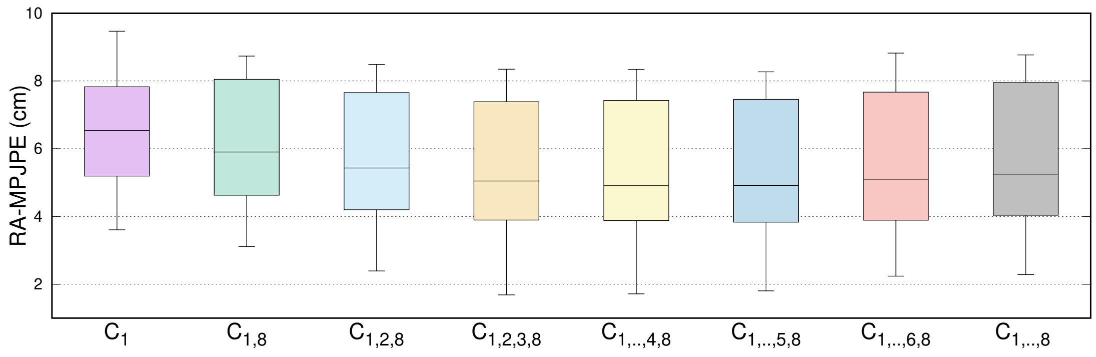


Figure 4.7: Box plot showing RA-MPJPE of our method using different camera configurations.

optimization chunks. For example, if the sample frame number is 25, the optimization chunks are frames 1 to 25 and frames 25 to 49. The blue-colored estimates (subject and camera) are the optimization results performed on the first chunk (from 1-25) and white-colored ones are from the second chunk (from frames 25-49). The origins for both of them are aligned, and the overlays are shown in the left and right side images. We can see that the overlays look nearly correct for both of them, but there are minor differences in the subject's global pose (orientation and position), which results in the camera poses being quite far from each other. This shows that inferring the subject's and the camera poses using just 2D keypoints gives only the information about the articulated pose of the subject and the relative poses of the camera and the subject.

Note that we do not optimize the SMPL position and orientation in the first frame, as it should act as the pivot and bring the person conforming to the first frame using Eq. (4.5) and cameras using Eq. (4.4). Due to this, the estimate in the first frame is noisy and because of Eq. (4.8) and Eq. (4.7), a few initial frames become noisy. That is why we ignore the first 10 frames for evaluation.

In Table 4.3, we compare the monocular and the multi-view version of our method (row 3 and 4) with the reference methods GLAMR (9) (row 1) and HuMoR (8) (row 2), which are the state-of-the-art monocular human pose and shape estimation methods. We see that our method significantly outperforms these methods. Both HuMoR and GLAMR use a motion prior that encodes human motion transitions instead of absolute motions. As we discussed at the beginning of this chapter, reconstructing the motion from the space of motion transitions is very sensitive to noise and can lead to spurious results. In Fig. 4.8, we show the qualitative results of our method and the reference method (HuMoR) and compare them with the GT. The 3D reconstruction of the human and the cameras relative to the ground plane are shown for our method (blue), the GT (green) and the reference method (8) (red). For a clearer illustration, we render each pose by adding a time-dependent offset to the position estimate at that time. Camera pose estimates are unchanged for the rendering. We can see that our estimates are very close to the GT, while the reference method estimates are quite inaccurate. For example, in the left inset box, we see that both feet are on the same side, giving a physically implausible global pose estimate. In the right inset box, the person's body suddenly rotates more than 90°, again resulting in a physically implausible motion. In Fig. 4.9, we do a qualitative comparison of our method with GLAMR (9) by showing the resulting mesh overlaid on the original image. While the results from our method are nearly perfectly aligned with the person in the image, the overlaid GLAMR results do not match the person. This is due to the errors in both the estimated person's poses and the estimated camera pose.

Since SmartMocap is an optimization-based method, it is slow compared to regression-based methods. We observe that it takes around 25 minutes to process a chunk of 25 frames on an i7 CPU. Since the optimization in step 2 (Sec. 4.1.3) can be done in parallel for each chunk of 25 frames, we stick to a simpler CPU implementation which allows us to do so. A GPU implementation of our method which allows parallel processing of 25 frame chunks, is not straightforward with standard libraries such as Pytorch. Compared to Aircap-Pose-Estimator and AirPose^{+}, SmartMocap is the slowest because it iteratively refines every pose in multiple steps (optimize $\rightarrow$ stitch $\rightarrow$ optimize $\rightarrow \cdots$). Also, it cannot run on a GPU like the other two if it has to utilize the parallel execution of frame chunks.

AirPose dataset

In Fig. 4.10, we show qualitative results of our method on the AirPose real-world dataset. We show the cropped version of the original images of the subject, along with the same image with the estimated mesh overlaid on top. Two adjacent columns are the two views at the same time instant. The results show that our method can reconstruct the diverse

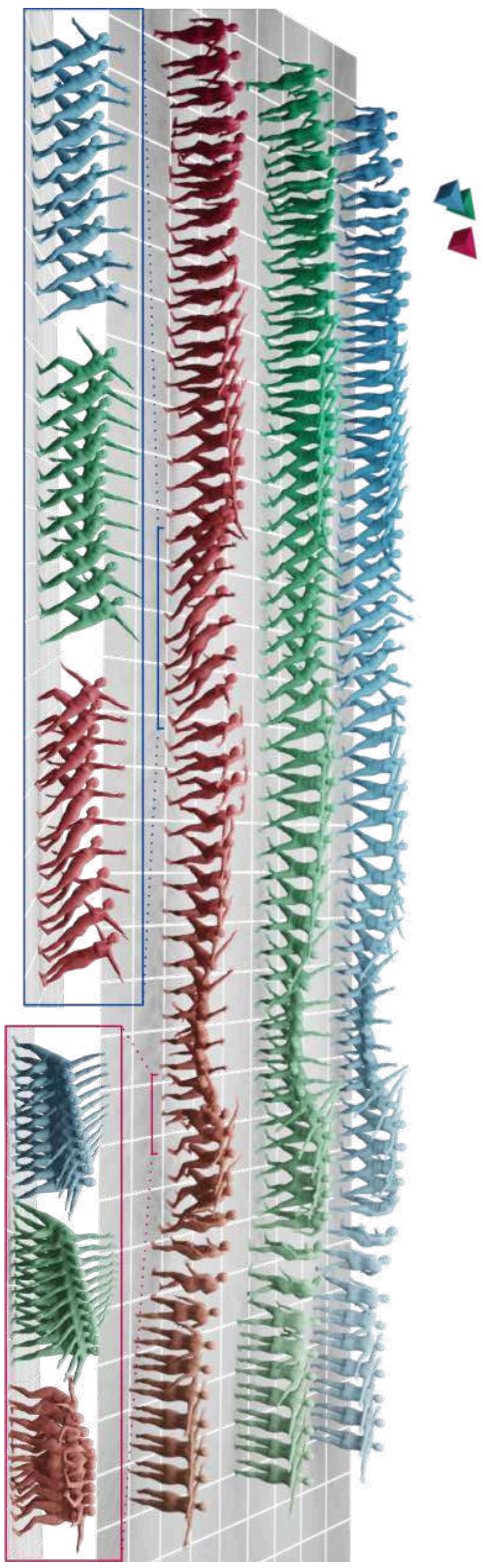


Figure 4.8: Comparison of our method (blue), HuMoR (8) (red) and the GT (green). For clearer illustrations, the human poses are shifted sideways using time-dependent offsets.

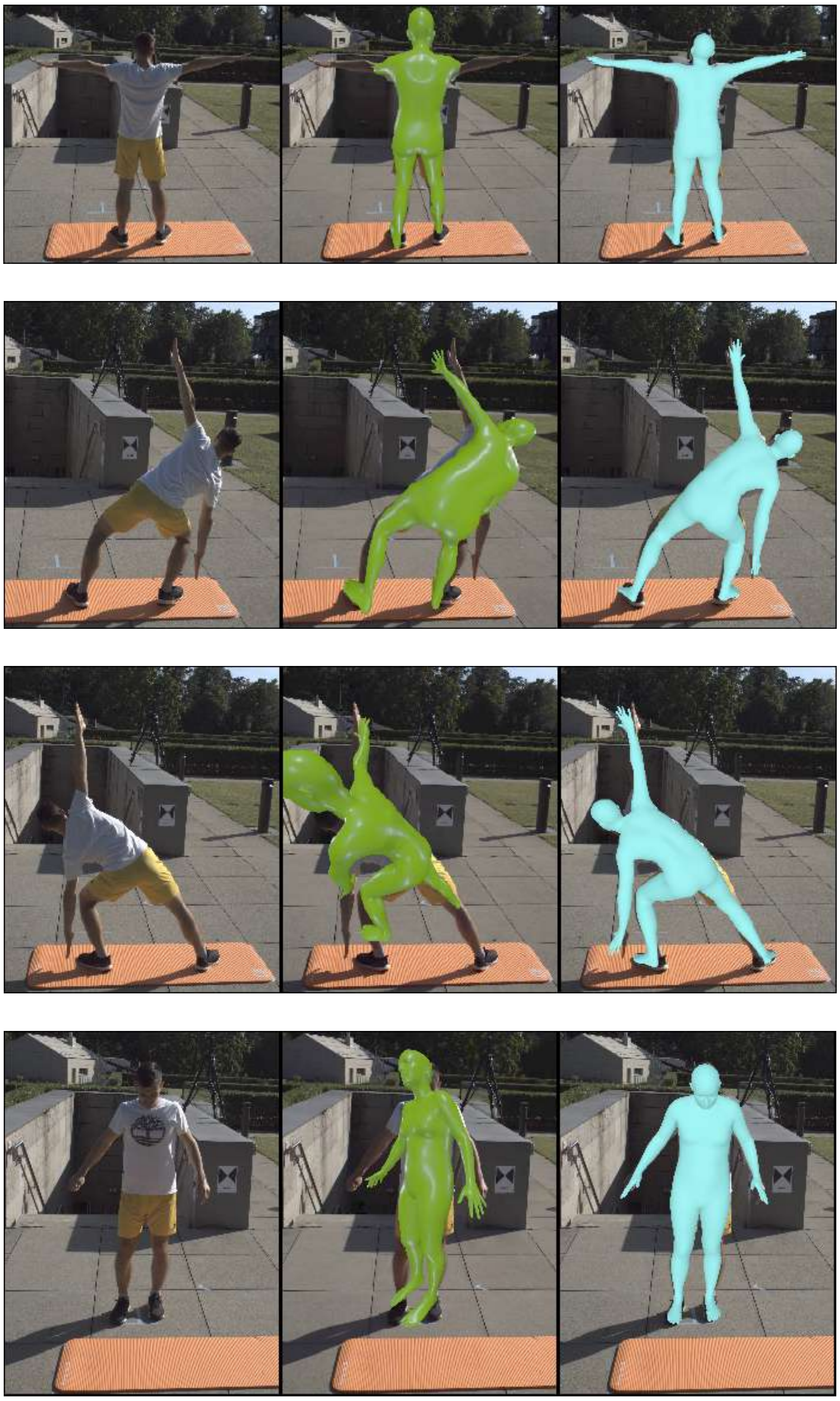


Figure 4.9: Comparison of GLAMR (9) and our method using a single camera on the RICH dataset. We show images at 4 time instants, each containing the original image, the GLAMR results (green) and results from our method (cyan) using a single camera.

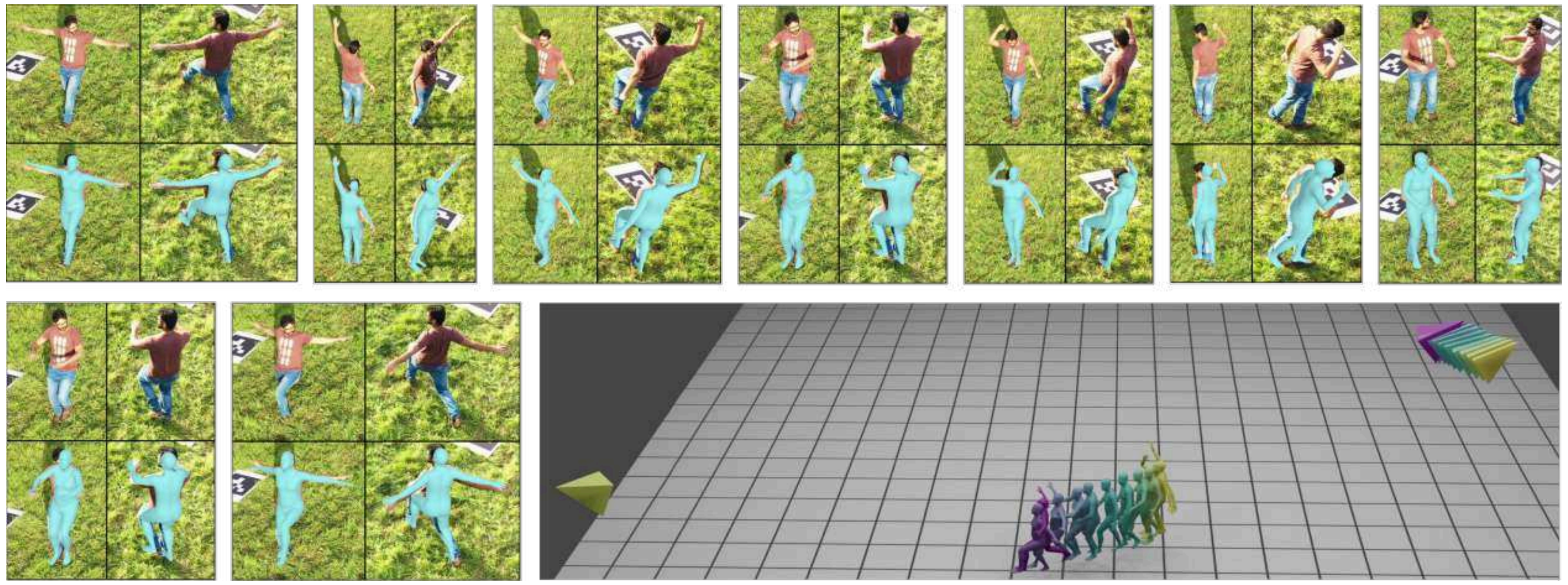


Figure 4.10: Results on the AirPose real-world dataset. The result at each time instant is shown using an image grid of 2×2. The top row shows the cropped region of the actual image, while the bottom row shows the estimated mesh overlaid on top of it. Each column corresponds to each camera view. The bottom-right image shows the full 3D reconstruction of the subject's poses, shape, and the camera poses.

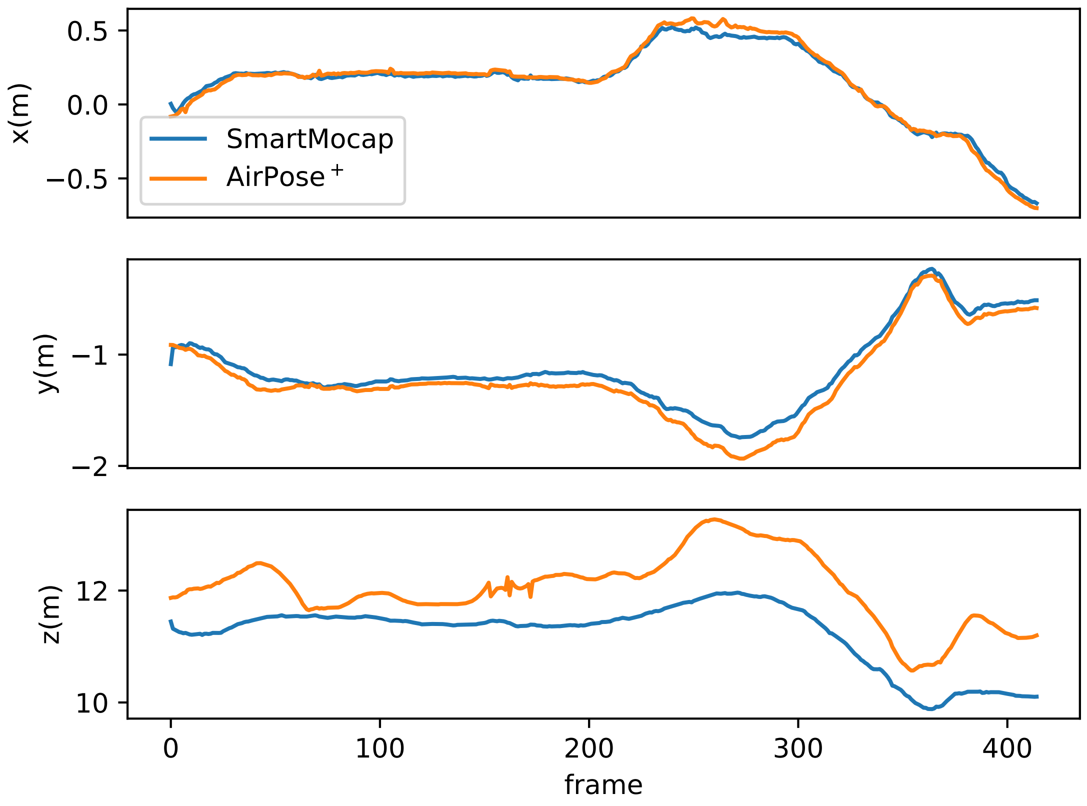


Figure 4.11: Person’s estimated position relative to cam1 on the AirPose dataset.

MPE (cm)	MOE (rad)	RA-MPJPE (cm)	MPVPE (cm)
80.70 ± 32.02	0.29 ± 0.22	5.04 ± 5.66	6.14 ± 3.30

Table 4.4: SmartMocap error metrics computed on the AirPose dataset using AirPose$^+$ estimates as the reference.

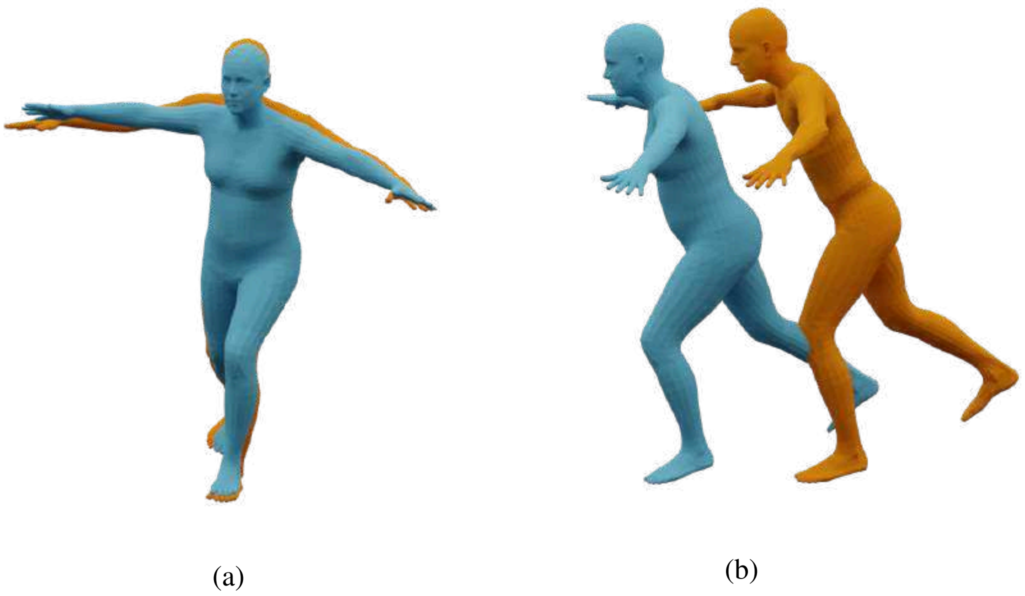


Figure 4.12: 3D bodies relative to the camera estimated by AirPose$^+$ (orange) and SmartMocap (blue). (a) Estimated 3D bodies viewed from the estimated camera. (b) Side view of the estimated 3D bodies.

poses captured from an aerial view. In the bottom-right corner, we show the 3D reconstruction of the subject's poses, shape, and the camera poses for a sub-sequence. The colour gradient from yellow to violet is used to show the time transition. We observe that the subject's reconstructed body is not touching the ground, but lies a bit above the ground plane. This is because the actual terrain is not a plane, but a sloped hilly terrain. Even though the motion prior is trained on the motion sequences performed on a plane surface, our method can still recover the global motion on terrain with a small slope.

We also compare the SmartMocap results on the AirPose data with the results from AirPose$^+$. Since AirPose$^+$ estimates the subject's pose relative to the camera, we take a sequence where cam1 is static and compare the estimated results (including global human pose) relative to cam1. Also, AirPose$^+$ uses SMPL-X model while SmartMocap uses SMPL body model for human body representation. That is why, we first convert the AirPose$^+$ estimates from SMPL-X to SMPL parameters using the model conversion code (107). In table 4.4, we show the metrics MPE, MOE, RA-MPJPE and MPVPE of SmartMocap results computed using the AirPose$^+$ results as reference. To compute them, We use AirPose$^+$ results instead of GT in Eq. (4.15), (4.16), (4.17) and (4.18). We see that the mean difference between the person's estimated position by the two methods is as high as 80 cm. In Fig. 4.11, we can see that this difference is mainly in the z-direction. In Fig. 4.12a, we show the estimated 3D body by SmartMocap (blue) and AirPose$^+$ (orange) rendered from the camera's viewpoint. The same estimates are shown from the side view in Fig. 4.12b, and we can see that while both the bodies overlay over each other when looked from the camera's viewpoint, but the AirPose$^+$ estimate is a bit further than the SmartMocap in camera's looking direction. Since both the methods fit the body to the 2D keypoints, AirPose$^+$ estimates a bigger body further away from the camera and able to fit to the same 2D keypoints. We can see the difference in shape estimation using MPVPE shown in table 4.4. The MOE and RA-MPJPE metrics show that the articulated pose and orientation estimates from both methods are quite close, which is expected since the 2D keypoints are usually sufficient to resolve them.

Smartphone dataset

We also show the qualitative results of our method on our smartphone dataset. Similar to the AirPose results, we show the cropped images, the overlaid estimated mesh and the 3D reconstruction of the subject and the cameras in Fig. 4.13. Our method accurately reconstructed the pose of the subject playing football and the camera motion along the wall of the playing arena. We see that the overlays are near-perfectly aligned with the images, showing the accurate reconstruction of the relative pose of the camera and the person. The complete reconstruction is shown in the bottom-right image, and we see that the subject's motion and the camera motion are temporally and spatially coherent.

4.3 Limitations

Our method assumes a planar ground surface and human motions which do not involve moving ground (e.g. a treadmill) or sliding motions (e.g. skating, skiing, etc.). However, it can be extended to non-planar ground surfaces by encoding the surface, articulated poses and global poses together in a prior. A major limitation in training such a prior network is the unavailability of human mocap data where the ground surface is also captured. Moreover, most existing datasets are collected with a human subject moving on a planar ground surface. Another limitation of SmartMocap is that it needs at least one camera to be static at any given time. If all the cameras are moving at the same time, the 2D keypoints are our motion prior are not sufficient to restrict the resultant motion on a ground plane.

4.4 Conclusion

In this chapter, we presented a method to reconstruct the 3D human poses, shape, and camera poses relative to a global coordinate frame using synchronized RGB videos from single/multiple extrinsically uncalibrated cameras. We use the ground plane as the reference coordinate system and train a human motion prior using a large amount of human mocap data. We use the latent space of this motion prior to fit the SMPL body model to the 2D keypoints in all the views simultaneously. We show our results on two existing dataset and one new dataset that we collect using smartphones. We show that

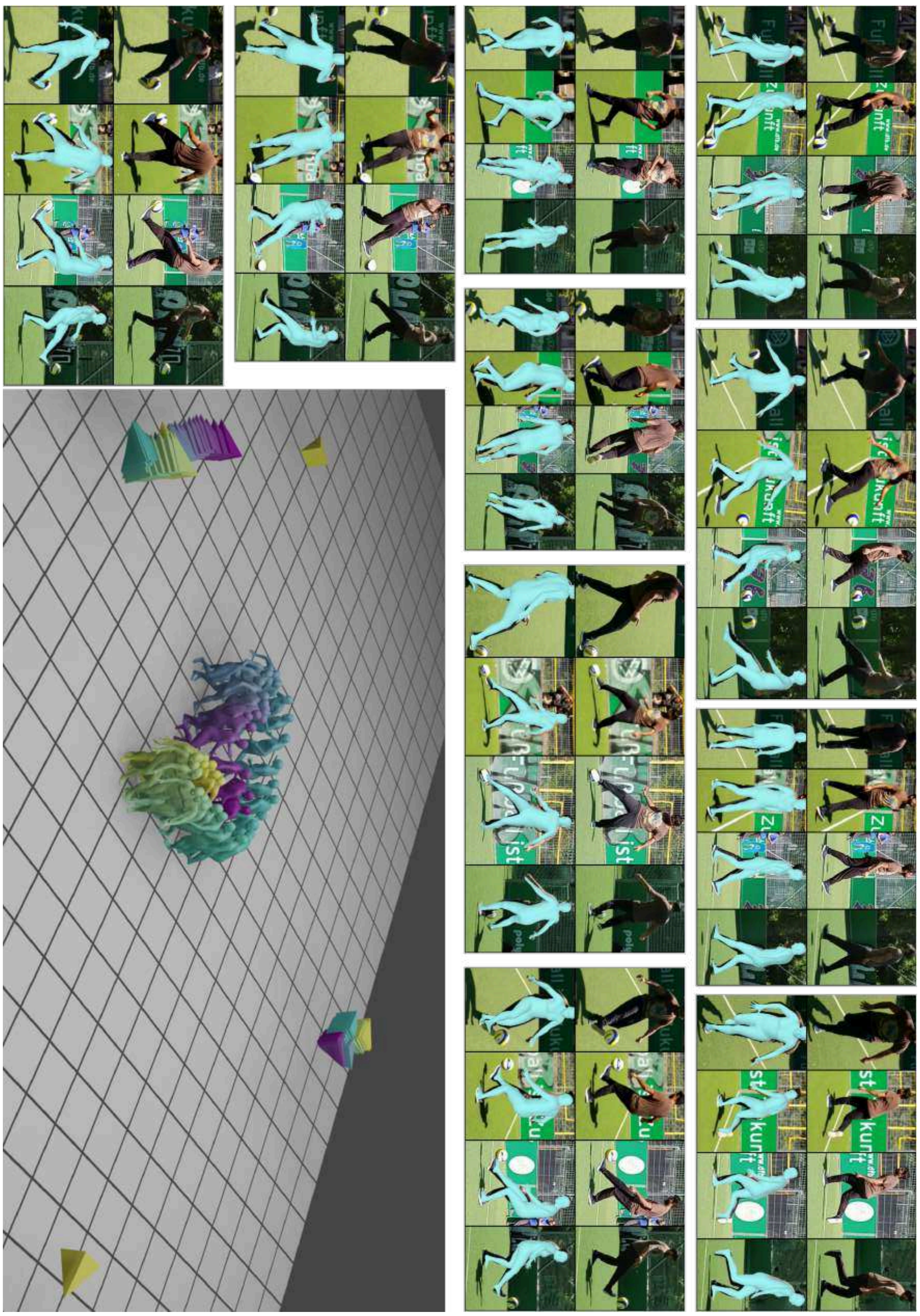


Figure 4.13: Results on our smartphone dataset. The result at each time instant is shown using an image grid of 2×4 (check Fig. 4.10 caption for details).

our method reconstructs the human poses, shapes, and camera poses on all the three datasets. We showed the quantitative results on the RICH dataset, demonstrating that our method achieves more accurate results compared to a state-of-the-art method on the task of monocular human motion reconstruction. We also analyze the effects of multiple views on the performance of our method. We show that our method works for diverse types of camera views by showing qualitative results on all three datasets. The accurate reconstruction by our method on the smartphone data is evidence of the ease of use of our method.

Chapter 5

Summary and Future Work

The main challenge we try to solve in this thesis is the reconstruction of 3D human and camera motion from moving RGB cameras. Even though many previous methods try to recover human motion from videos, they focus on recovering the articulated motion of the human using a single camera and leave out the global motion of the human and the camera (4; 19; 37; 38; 39; 108; 109; 110; 111). Recovering this 3D global information from a single video is difficult because the video only encodes the relative motion between the human and the camera. One way to address this is to keep the camera static. However, even with a static camera, it is difficult to recover the global motion of the human subject, especially when the relative depth changes (for example, the person moving towards or away from the camera). Also, a single camera view cannot handle occlusions and results in a less accurate estimate of the 3D pose. A typical approach is to use multiple static cameras (55; 56; 57; 58; 59; 112), calibrate them to get extrinsic parameters relative to an origin, and then estimate the human motion relative to that origin. This works fine for static cameras because the camera extrinsics do not change over time. However, in the case of moving cameras, pre-calibration of the cameras would not work and the cameras need to be calibrated every frame. However, getting extrinsic parameters when the cameras are moving is a challenging problem. We explore multiple approaches to estimate the 3D pose and shape of the person along with the camera poses.

In chapter 2, we mount inertial sensors on the cameras to get their approximate extrinsic parameters and use them to estimate human motion. We introduce Aircap-Pose-Estimator, which is an autonomously flying mocap system consisting of multiple UAVs mounted with RGB cameras, compute, IMU and GPS sensors. All the UAVs process the images from the cameras and track and follow the human subject while maintaining a formation and avoiding obstacles. We use this system to collect the data and process it offline to get the poses of the subject at every time frame. During the online processing, the data from onboard sensors is fused with the person's rough position relative to the cameras to get the online estimate of the person's position and the camera poses. In the offline phase, we use 2D keypoints to fit the SMPL body model to them, estimating the pose of the subject and the cameras together. We show that this step further improves the position estimate of the cameras. Effectively, we use the human subject to improve the extrinsic calibration of the cameras. This is a key contribution of our work, which we use in further works to calibrate the cameras relative to the person (AirPose) or the ground plane (SmartMocap). We quantitatively evaluate Aircap-Pose-Estimator using an existing IMU-based method (29). We also show the qualitative results of our method by overlaying the estimated 3D body on the captured images.

A major limitation of the Aircap-Pose-Estimator is that it is very slow and cannot do the estimation in real-time. In general, optimization based methods are slow because they need to perform several optimization iterations per frame. Neural network based regression methods are usually much faster. However, most of them are monocular (4; 36; 37), and not designed to run on embedded hardware, which is usually the only compute available with moving cameras e.g. handheld cameras and UAVs. The same is the case with the multi-view regression based methods (50; 51; 52; 64). They do not focus on real-time execution, but on fusing information from different views. To address the challenge of real-time multi-view markerless mocap with extrinsically uncalibrated moving cameras, we must address two challenges, 1) fast execution on embedded hardware, and 2) efficient fusion of information from multiple views.

In chapter 3, we propose a lightweight neural network AirPose that can run on embedded hardware in real-time, efficiently fusing the features from multiple views. We start with an existing monocular neural network HMR (4) which can regress the SMPL (19) body pose parameters from a single image. HMR uses the ResNet50 features of the image to iteratively improve the SMPL pose parameters in multiple stages. A major limitation of HMR (4) and other methods inspired from it (22; 26; 42; 54) is that they crop out the region around the person from the original image, scale it to a fixed size and do the inference on it. This cropping and scaling operation reduces the computation significantly, making the network lightweight enough to run on embedded hardware. However, it results in the loss of information which is crucial for global pose estimation of the subject. AirPose addresses this information loss by introducing a compact representation of the original image. This representation allows the network to do the inference on the original image without compromising on the execution speed. We propose a novel network architecture (AirPoseNet) to address the challenge of efficient fusion of information from different views. We utilize the iterative regression stages of HMR and modify them to fuse only the view-independent features (shape and articulated pose of the subject) from different views. Each stage of the AirPoseNet instance takes in the output from the previous stage of the same and another network instance. An instance of AirPoseNet runs onboard in real-time, enabling the UAVs to actively plan their motion depending on the pose of the person. Identical instances of AirPoseNet run on every UAV, taking in the image from the onboard camera and estimating the articulated and global pose of the subject relative to the camera. The features are shared across the UAVs using the onboard wifi. We overcome the challenge of training this network by developing a realistic-looking synthetic data pipeline using Unreal Engine. After getting trained on this data, the network is fine-tuned on a small amount of real-world data, resulting in generalization on the real-world data. We implement AirPose on the same hardware as AirCap-Pose-Estimator and evaluate its execution speed using hardware-in-the-loop experiments. Finally, we introduce AirPose$^+$, which is a post-processing fitting method to get temporally coherent and smooth estimates of human motion.

Every instance of AirPose estimates the pose of the subject relative to the camera. This gives the poses of the cameras and the subject relative to each other. However, there is no information about the transform of the cameras or the subject relative to the 3D world. That is why, the estimated human motion cannot be automatically placed in a 3D world. Multiple previous works on human motion generation use the motion representation relative to a ground plane (113; 114; 115). These methods learn the prior distribution of human motion relative to the ground plane, which allows them to generate human motion with a global component. However, this global motion representation has not been used in the human motion estimation methods until very recently by HuMoR (8) and GLAMR (9). HuMoR conditions the body pose at any time instant on the pose at the previous time instant. This allows them to sample arbitrarily long motion sequences, but it makes the future samples sensitive to noise at any time instant. They are also focused on estimating the subject's global pose and the ground plane from a single static camera. GLAMR first estimates the articulated poses of the person from a single video, then predicts global motion conditioned on the articulated poses. This does not allow the motion-prior to correct the inaccuracies in articulated poses from the video.

In chapter 4, we propose SmartMocap which addresses the general problem of estimating the motion of a human subject using multiple/single static/moving extrinsically uncalibrated cameras. SmartMocap learns a human motion-prior using a motion representation, where the human motion is represented relative to the ground plane. As opposed HuMoR which encodes a pose conditioned on past frames, this motion prior encodes a sequence of human poses (articulated and global), making it more robust to per-frame noise. SmartMocap uses this prior along with the 2D keypoints in batch optimization to estimate the pose of the subject and the cameras relative to the ground plane. We show quantitative and qualitative results on RICH dataset (31) and compare them with HuMoR (8) and GLAMR (9). We compare with AirPose$^+$ on AirPose dataset and show qualitative results on another dataset which we collect using 4 smartphones (2 static and 2 moving). We show the reconstructed motion of the subject and the cameras placed in the 3D world along with the ground plane. SmartMocap proves to be an extremely easy mocap method where a user can just place or hold the cameras and start the mocap session without any preparation.

5.1 Future work

This thesis is a step toward an ultimate mocap system that can reconstruct high-quality 3D human motions from RGB videos taken from single/multiple moving cameras. Some of the features needed for such a mocap system are achieved in this thesis, such as camera extrinsic estimation, human motion-prior with ground interaction, real-time onboard processing etc. There are still many challenges that need to be addressed, and we discuss them along with some possible approaches in the following.

5.1.1 Camera intrinsic parameters

In this thesis, we assume the camera extrinsics are unknown and all of our methods estimate them. This allows us to change the camera positions during the mocap. The mocap session becomes a single-step process where the user can just place the cameras and start capturing motions. However, camera intrinsic parameters (mainly focal length) are also needed for markerless mocap. In all of our methods, we assume that these are available to us. Because of this assumption, our methods cannot be used for videos on the internet, where the camera intrinsic parameters are unknown to us. Most of the monocular regression methods (4; 22; 37; 42; 111) either assume that the person is very far from the camera and use the weak-perspective camera model, or they assume a standard 55° field-of-view (15; 102). These assumptions lead to noisy global pose estimates of the subject, especially in the case of videos shot from smartphone cameras, which are the most common type of moving cameras.

The intrinsic calibration of an RGB camera is commonly done using a checkerboard pattern with known edge length (116). Other classical methods rely upon known objects (117) or geometries (118) in the scene. One way of using these methods along with the markerless mocap is to keep these calibrating objects in the scene. However, these methods are not directly useful in our case of moving cameras because we cannot enforce such calibration patterns to be always present in the frame. Also, this is not generalizable to the existing internet videos. The only way is to get the intrinsic parameters from in-the-wild videos. Recently, there have been works on estimating camera intrinsics from in-the-wild images (119; 120; 121; 122) or videos (123; 124). While they are shown to give accurate results on images and static scenes with moving cameras, their performance in scenes with articulated and moving objects (such as humans) is yet to be tested.

Some of the works try to solve the challenge of unknown camera intrinsics to get better human pose and shape estimates (38; 70; 125). SPEC (24) is a deep neural network which estimates the pose (pitch, roll) and focal length of the camera along with the 3D pose and shape of the person from a single image. However, the focal length is learned by using synthetic data, which is usually not generalizable to the images out of the training data. The camera poses estimation of SPEC is based on estimating the horizon line in the images, which is more suitable for tasks involving human perception (126) rather than 3D estimation. Recent work BEDLAM (23) has shown promising results on in-the-wild images while being trained only on synthetic data. It can be further extended to estimate camera intrinsics along with the human and camera motion from the videos. This would require rendering the synthetic images using a wide variety of camera focal lengths (especially smartphones) and realistic camera motions.

Recent methods (14; 45) use state-of-the-art SLAM methods (127) to get the camera motion. SLAM methods can estimate the camera motion and the environment, but the scale of both is ambiguous. These methods utilize the human subject in the scene to disambiguate this scale. They use the SMPL model which provides a shape space of the human body and restricts the size of humans to stay in a valid range. A similar approach can be used to estimate the camera focal length. Estimating the focal length of a camera is possible if a few keypoints are known with 3D-2D correspondence (128) and the distances between the 3D keypoints are known (116; 129). Even though only static and rigid 3D objects are used until now, this can be extended to our case, where we can use human 3D joints and their corresponding 2D keypoints to get the camera focal length. The difference is that the human body is articulated, and we don't know the human skeleton size beforehand. However using the SMPL shape space can help restrict the set of possible focal length values, and taking into account the several video frames would result in a more confident estimate.

5.1.2 Interaction with environment

In Chapter 4, we train a motion prior which implicitly learns the human interaction with the ground plane. This helps in decoupling the camera motion from the subject's motion. However, this is not sufficient if all the cameras are moving. We need explicit physical constraints on human motion to avoid motion artifacts such as foot sliding. Which will also improve the perceptual quality of the motion (130; 131). A possible approach would be to explicitly use the points of contact between the human body and the ground surface, similar to HuMoR (8). However, getting good-quality contact point labels is not trivial. These labels are usually collected during the mocap using pressure sensors (132; 133). We trained our motion prior using the AMASS dataset which is a collection of multiple pre-existing motion capture datasets transformed in a unified format (SMPL parameters) using MoSH (134). Getting the contact points for an existing dataset is not straightforward. HuMoR first annotates AMASS using heuristic-based annotations such as zero velocity at the contact point, and the contact point should be close to the ground, and then learn to regress them. A similar approach has been taken for predicting contact points in 2D (135), but in both cases, contact points are heuristic-based and not completely reliable.

Recent works on generative motion modelling such as (44; 135; 136; 137; 138; 139; 140) use physics to impose physically correct human ground interaction. A major challenge for such methods is to develop a model of the person which is supported by physics engines. This model is usually developed using primitive objects such as spheres, cylinders, ellipsoids etc., resulting in an approximated human body which cannot change shape realistically as SMPL. Hence, the existing large-scale data such as AMASS cannot be directly used to train the model, instead, the model training relies upon the physics. However, the physics is not accurate because of the approximated modelling of the body parts. For example, the feet are modelled as a cuboid, resulting in inaccurate physical interactions with the ground. The resulting motion looks fine in the physics engine, but the artifacts are seen when visualized with SMPL body (136). Even after such artifacts, the physical modelling proves to help mitigate the foot skating artifacts, which can help decouple the camera and human motion. A possible direction for us is to train or use a pretrained human motion model including the physics, and include it in our human motion estimation pipeline.

In our future work, we also want to estimate human motion on a non-planar ground surface. All the existing works assume a planar ground surface. This is because most of the mocap data is captured on a planar ground surface, thus, the motion models learned on such data are implicitly biased and not generalizable to non-planar surfaces. To learn a motion model for a non-planar surface using data needs a huge amount of mocap data collected on a variety of terrain, which is a very costly process. Some of the previous methods (113; 141; 142) try a different approach to augment the collected mocap data with a variety of terrain. They collect the mocap data and then fit a variety of terrain to the same motion sequence. Another alternative is to use a physics simulator and train a motion model using reinforcement learning (143; 144; 145; 146). However, as mentioned previously, the learned motion is not completely natural because of the approximated modelling of the human body and the physics. Some methods overcome this by adding supervision using the real-world mocap data (144; 145). All these discussed methods are generative models designed for character animation. Using them or modifying them for markerless mocap is an added challenge.

5.1.3 Multiple people

In this thesis, we focus on capturing the motions of a single human subject. However, interesting real-world scenes consist of multiple people interacting with each other. Such motions contain rich information about human-human social interactions. In the future, we will extend our methods to reconstruct such motions in 3D. A straightforward way is a top-down approach where we first detect and track every person in the scene and reconstruct their 3D motion separately (9; 57; 69; 147; 148; 149; 150). A common challenge in such an approach is to reliably track each person because of the severe occlusions in multi-person scenarios. State-of-the-art methods (9; 57; 147) are proven to be good enough for tracking and re-identification in simpler scenarios where the person is not very far and not occluded for a very long time. However, these top-down approaches sometimes result in incoherent estimates which might look fine individually but outputs an inconsistent relative placement of the people in the 3D world (151). Contrarily, the bottom-up approaches try to reconstruct the motion of all the subjects in the scene at once, instead of doing it separately for each subject. This makes the reconstructed motion more coherent in the 3D world (151; 152; 153; 154). These approaches usually involve processing the full image, thus they are slower than the top-down approaches. However, the execution time of bottom-up approaches is independent of the number of people in the scene, while the top-down approach becomes computationally costlier as the number of people in the scene increases.

In all of our works, we estimate the pose of the cameras using the human subject in the scene. The central idea behind all of our approaches is to locate the person relative to every camera, which will give the pose of all the cameras relative to each other. Additionally, in SmartMocap, the person's pose relative to the ground plane is known, thus the poses of the cameras relative to the ground plane are also known. Having multiple people in the image gives more information for the camera pose estimation and has been utilized previously (69). The performance of all our methods will improve from having multiple people in the frame. Even with a simple approach of independent detection and tracking of the subjects, we can enforce a cross-view consistency among the relative poses of the subjects.

5.1.4 Real-time processing

A real-time mocap setup helps animators get quick feedback on their captured motion and saves them time and money. It is difficult to label a method as real-time because its execution time is also dependent on the hardware it is being executed on. The same method which runs at 60 FPS on a powerful desktop GPU might take more than a second on an embedded device. We consider a method as real-time if it can process 3-4 frames per second on a desktop GPU. Although this is not sufficient for applications such as real-time streaming, it is sufficient for applications where a decision has to be made depending on the motion, such as motion planning of UAVs following a subject, or an animator getting feedback on the motion performed by a mocap subject.

Optimization-based methods are too slow to be real-time. They rely on hundreds of forward and backward passes through a computation graph for a single frame, which is usually not possible to do in a fraction of a second even for very powerful computation hardware. Some methods (155; 156; 157; 158) try to learn the gradients instead of calculating them through backpropagation which results in fewer iterations and makes the method robust towards the initialization. Neural network based regression methods (4; 15; 22; 37) are much faster because they directly regress the body parameters, but they need a large amount of annotated data for training. Recent methods (159; 160) learn to regress the 3D pose and shape from intermediate representations such as human 2D keypoints and silhouettes. To learn the regressor, they generate a large amount of image data with paired ground truth pose and shape by rendering the SMPL body from AMASS. However, this process relies upon accurate estimation of the 2D keypoints or silhouettes from the images during test time.

Our method AirPose, uses a large amount of synthetic data AGORA (7) to overcome the challenge of data unavailability. AGORA consists of static human 3D scans, therefore, we can only train our model to estimate the subject's pose on a single frame. The latest dataset BEDLAM (23) consists of full motion sequences and the network trained on this dataset has shown good generalizability on real-world images. BEDLAM consists of videos of clothed SMPL bodies with realistic clothing and hair rendered in Unreal Engine. Their motions are the real motion sequences taken from the AMASS dataset. This dataset will allow us to train a model that can estimate the full motion of the subject using a static camera. This dataset is still limited to static cameras but it can be further enriched by adding realistic camera motions in Unreal Engine. Possible approaches to acquire realistic camera motions could be using sensors from handheld cameras or a SLAM method (14; 127; 161) on the videos from the internet.

5.2 Conclusion

In this thesis, we address the challenges associated with 3D human pose and shape reconstruction from multiple RGB cameras. While state-of-the-art methods for human shape and pose estimation need calibrated cameras, we focused on estimating both the 3D shape and motion of a person and the 3D motion of the cameras themselves.

In the first chapter, we introduce the challenges in markerless mocap and the limitations of current methods. In the second chapter, we introduce AirCap-Pose-Estimator, which employs a system consisting of multiple UAVs with RGB cameras, onboard compute and IMU and GPS sensors to collect data. We process this data offline to estimate the camera poses and the 3D pose and shape of the person. First, we use the onboard sensors to approximate camera poses in the 3D world. Then we fit the SMPL body model to the 2D keypoints in all the views simultaneously, while improving the camera pose estimates at the same time.

In the third chapter, we introduce AirPose, a real-time onboard method to estimate the shape and pose of the person relative to the cameras. This method introduces a compact representation of full images and efficient feature fusion across views, which allows AirPose to give more accurate estimates without compromising on execution speed. AirPose is trained on a large amount of synthetic data, followed by fine-tuning on small real-world data. We further introduce AirPose$^+$ which is an offline bundle-adjustment method to enhance the accuracy of human pose and shape estimation giving smooth and temporally coherent human motion.

However, a limitation of AirPose and AirPose$^+$ is their inability to estimate the 3D human motion relative to the world. We address this in the fourth chapter by introducing SmartMocap. SmartMocap estimates the 3D shape and pose of both the person and the cameras relative to the ground plane. We achieve this by training a motion prior network which encodes the human motion relative to the ground plane. We use the 2D keypoints in all the views to fit the 3D human shape and motion in the latent space of our human motion prior. our motion representation keeps the estimated human motion relative to the ground plane, and, the 2D keypoints resolve the camera motions relative to the human motion. Thereby allowing all the estimates to be placed in a 3D scene relative to a common reference frame which is the ground plane.

In summary, this thesis provides a comprehensive exploration of 3D human shape and motion estimation methods using multiple uncalibrated moving cameras. The contributions made in each chapter collectively advance the understanding and capabilities of long-range markerless human mocap systems.

Bibliography

[1] F. Bogo, A. Kanazawa, C. Lassner, P. Gehler, J. Romero, and M. J. Black. 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, pages 561–578. Springer International Publishing, Oct. 2016.

[2] P. Guan, A. Weiss, A. O. Bãlan, and M. J. Black. Estimating human shape and pose from a single image. In *2009 IEEE 12th International Conference on Computer Vision*, pages 1381–1388, 2009.

[3] N. Mahmood, N. Ghorbani, N. F. Troje, G. Pons-Moll, and M. Black. Amass: Archive of motion capture as surface shapes. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5441–5450, 2019.

[4] A. Kanazawa, M. J. Black, D. W. Jacobs, and J. Malik. End-to-end recovery of human shape and pose. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7122–7131, 2018.

[5] N. Kolotouros, G. Pavlakos, M. J. Black, and K. Daniilidis. Learning to reconstruct 3D human pose and shape via model-fitting in the loop. In *Proceedings International Conference on Computer Vision (ICCV)*, pages 2252–2261, Oct. 2019.

[6] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292, 2014.

[7] P. Patel, C.-H. P. Huang, J. Tesch, D. T. Hoffmann, S. Tripathi, and M. J. Black. Agora: Avatars in geography optimized for regression analysis. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13463–13473, 2021.

[8] D. Rempe, T. Birdal, A. Hertzmann, J. Yang, S. Sridhar, and L. J. Guibas. Humor: 3d human motion model for robust pose estimation. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 11468–11479, 2021.

[9] Y. Yuan, U. Iqbal, P. Molchanov, K. Kitani, and J. Kautz. Glamr: Global occlusion-aware human mesh recovery with dynamic cameras. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11028–11039, 2022.

[10] J. L. Schönberger and J.-M. Frahm. Structure-from-motion revisited. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4104–4113, 2016.

[11] J. L. Schönberger, E. Zheng, J.-M. Frahm, and M. Pollefeys. Pixelwise view selection for unstructured multi-view stereo. In B. Leibe, J. Matas, N. Sebe, and

M. Welling, editors, *Computer Vision – ECCV 2016*, pages 501–518, Cham, 2016. Springer International Publishing.

[12] Y. Huang, F. Bogo, C. Lassner, A. Kanazawa, P. V. Gehler, J. Romero, I. Akhter, and M. J. Black. Towards accurate marker-less human shape and pose estimation over time. In *International Conference on 3D Vision (3DV)*, pages 421–430, 2017.

[13] K. Iskakov, E. Burkov, V. Lempitsky, and Y. Malkov. Learnable triangulation of human pose. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7717–7726, 2019.

[14] V. Ye, G. Pavlakos, J. Malik, and A. Kanazawa. Decoupling human and camera motion from videos in the wild. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21222–21232, 2023.

[15] Z. Li, J. Liu, Z. Zhang, S. Xu, and Y. Yan. Cliff: Carrying location information in full frames into human pose and shape estimation. In S. Avidan, G. Brostow, M. Cissé, G. M. Farinella, and T. Hassner, editors, *Computer Vision – ECCV 2022*, pages 590–606, Cham, 2022. Springer Nature Switzerland.

[16] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(1):172–186, 2021.

[17] H.-S. Fang, J. Li, H. Tang, C. Xu, H. Zhu, Y. Xiu, Y.-L. Li, and C. Lu. Alphapose: Whole-body regional multi-person pose estimation and tracking in real-time. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(6):7157–7173, 2023.

[18] Opencv camera calibration issues. https://github.com/opencv/opencv/issues/8813.

[19] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black. SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, 34(6):248:1–248:16, Oct. 2015.

[20] G. Pavlakos, V. Choutas, N. Ghorbani, T. Bolkart, A. A. Osman, D. Tzionas, and M. J. Black. Expressive body capture: 3d hands, face, and body from a single image. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10967–10977, 2019.

[21] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[22] N. Kolotouros, G. Pavlakos, M. Black, and K. Daniilidis. Learning to reconstruct 3d human pose and shape via model-fitting in the loop. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2252–2261, 2019.

[23] M. J. Black, P. Patel, J. Tesch, and J. Yang. Bedlam: A synthetic dataset of bodies exhibiting detailed lifelike animated motion. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8726–8737, 2023.

[24] M. Kocabas, C.-H. P. Huang, J. Tesch, L. Müller, O. Hilliges, and M. J. Black.

Spec: Seeing people in the wild with an estimated camera. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 11015–11025, 2021.

[25] J. Liang and M. Lin. Shape-aware human pose and shape reconstruction using multi-view images. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4351–4361, 2019.

[26] M. Kocabas, N. Athanasiou, and M. J. Black. Vibe: Video inference for human body pose and shape estimation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5252–5262, 2020.

[27] E. Price, G. Lawless, R. Ludwig, I. Martinovic, H. H. Bülthoff, M. J. Black, and A. Ahmad. Deep neural network-based cooperative visual tracking through multiple micro aerial vehicles. *IEEE Robotics and Automation Letters*, 3(4):3193–3200, 2018.

[28] R. Tallamraju, E. Price, R. Ludwig, K. Karlapalem, H. H. Bülthoff, M. J. Black, and A. Ahmad. Active perception based formation control for multiple aerial vehicles. *IEEE Robotics and Automation Letters*, 4(4):4491–4498, 2019.

[29] T. von Marcard, B. Rosenhahn, M. J. Black, and G. Pons-Moll. Sparse inertial poser: Automatic 3d human pose estimation from sparse imus. *Comput. Graph. Forum*, 36(2):349–360, may 2017.

[30] M. Kocabas, C.-H. P. Huang, O. Hilliges, and M. J. Black. Pare: Part attention regressor for 3d human body estimation. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 11107–11117, 2021.

[31] C.-H. P. Huang, H. Yi, M. Höschle, M. Safroshkin, T. Alexiadis, S. Polikovsky, D. Scharstein, and M. J. Black. Capturing and inferring dense full-body human-scene contact. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13264–13275, 2022.

[32] H.-J. Lee and Z. Chen. Determination of 3d human body postures from a single view. *Computer Vision, Graphics, and Image Processing*, 30(2):148–168, 1985.

[33] C. J. Taylor. Reconstruction of articulated objects from point correspondences in a single uncalibrated image. *Computer Vision and Image Understanding*, 80(3):349–363, 2000.

[34] V. Parameswaran and R. Chellappa. View independent human body pose estimation from a single perspective image. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pages II–II, 2004.

[35] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis. Scape: Shape completion and animation of people. *ACM Trans. Graph.*, 24(3):408–416, jul 2005.

[36] G. Moon and K. M. Lee. I2l-meshnet: Image-to-lixel prediction network for accurate 3d human pose and mesh estimation from a single rgb image. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VII*, page 752–768, Berlin, Heidelberg, 2020. Springer-Verlag.

[37] N. Kolotouros, G. Pavlakos, and K. Daniilidis. Convolutional mesh regression for single-image human shape reconstruction. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4496–4505, 2019.

[38] V. Leroy, P. Weinzaepfel, R. Brégier, H. Combaluzier, and G. Rogez. Smply benchmarking 3d human pose estimation in the wild. In *2020 International Conference on 3D Vision (3DV)*, pages 301–310, 2020.

[39] A. Gupta and L. Carlone. Online monitoring for neural network based monocular pedestrian pose estimation. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–8, 2020.

[40] B. Zhang, K. Ma, S. Wu, and Z. Yuan. Two-stage co-segmentation network based on discriminative representation for recovering human mesh from videos. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5662–5670, 2023.

[41] K.-C. Wang, Z. Weng, M. Xenochristou, J. P. Araújo, J. Gu, C. K. Liu, and S. Yeung. Nemo: 3d neural motion fields from multiple video instances of the same action. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 22129–22138, 2023.

[42] H. Choi, G. Moon, J. Y. Chang, and K. M. Lee. Beyond static features for temporally consistent 3d human pose and shape from a video. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1964–1973, 2021.

[43] Y. Sun, Q. Bao, W. Liu, T. Mei, and M. J. Black. Trace: 5d temporal regression of avatars with dynamic cameras in 3d environments. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8856–8866, 2023.

[44] R. Yu, H. Park, and J. Lee. Human dynamics from monocular video with dynamic camera movements. *ACM Trans. Graph.*, 40(6), dec 2021.

[45] D. F. Henning, T. Laidlow, and S. Leutenegger. Bodyslam: Joint camera localisation, mapping, and human motion tracking. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIX*, page 656–673, Berlin, Heidelberg, 2022. Springer-Verlag.

[46] H. Joo, H. Liu, L. Tan, L. Gui, B. Nabbe, I. Matthews, T. Kanade, S. Nobuhara, and Y. Sheikh. Panoptic studio: A massively multiview system for social motion capture. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 3334–3342, 2015.

[47] G. Pavlakos, X. Zhou, K. G. Derpanis, and K. Daniilidis. Harvesting multiple views for marker-less 3d human pose annotations. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1253–1262, 2017.

[48] H. Rhodin, F. Meyer, J. Spörri, E. Müller, V. Constantin, P. Fua, I. Katircioglu, and M. Salzmann. Learning monocular 3d human pose estimation from multiview images. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8437–8446, 2018.

[49] H. Qiu, C. Wang, J. Wang, N. Wang, and W. Zeng. Cross view fusion for 3d human pose estimation. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4341–4350, 2019.

[50] Y. He, R. Yan, K. Fragkiadaki, and S.-I. Yu. Epipolar transformer for multi-view human pose estimation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 4466–4471, 2020.

[51] L. Zhang, K. Zhou, L. Liu, Z. Li, X. Zhao, X.-D. Zhou, and Y. Shi. Progressive multi-view fusion for 3d human pose estimation. In *2023 IEEE International Conference on Image Processing (ICIP)*, pages 1600–1604, 2023.

[52] H. Tu, C. Wang, and W. Zeng. Voxelpose: Towards multi-camera 3d human pose estimation in wild environment. In A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, editors, *Computer Vision – ECCV 2020*, pages 197–212, Cham, 2020. Springer International Publishing.

[53] Z. Chen, X. Zhao, and X. Wan. Structural triangulation: A closed-form solution to constrained 3d human pose estimation. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part V*, page 695–711, Berlin, Heidelberg, 2022. Springer-Verlag.

[54] Z. Li, M. Oskarsson, and A. Heyden. 3d human pose and shape estimation through collaborative learning and multi-view model-fitting. In *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1887–1896, 2021.

[55] J. Gall, C. Stoll, E. de Aguiar, C. Theobalt, B. Rosenhahn, and H.-P. Seidel. Motion capture using joint skeleton tracking and surface estimation. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1746–1753, 2009.

[56] E. de Aguiar, C. Stoll, C. Theobalt, N. Ahmed, H.-P. Seidel, and S. Thrun. Performance capture from sparse multi-view video. *ACM Trans. Graph.*, 27(3):1–10, aug 2008.

[57] Y. Zhang, Z. Li, L. An, M. Li, T. Yu, and Y. Liu. Lightweight multi-person total motion capture using sparse multi-view cameras. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5540–5549, 2021.

[58] S. Corazza, L. Mündermann, A. M. Chaudhari, T. Demattio, C. Cobelli, and T. P. Andriacchi. A markerless motion capture system to study musculoskeletal biomechanics: Visual hull and simulated annealing approach. *Annals of Biomedical Engineering*, 34(6):1019–1029, May 2006.

[59] Y. Liu, C. Stoll, J. Gall, H.-P. Seidel, and C. Theobalt. Markerless motion capture of interacting characters using multi-view image segmentation. In *CVPR 2011*, pages 1249–1256, 2011.

[60] V. Belagiannis, S. Amin, M. Andriluka, B. Schiele, N. Navab, and S. Ilic. 3d pictorial structures revisited: Multiple human pose estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(10):1929–1942, 2016.

[61] S. Amin, M. Andriluka, M. Rohrbach, and B. Schiele. Multi-view pictorial struc-

tures for 3d human pose estimation. In *Procedings of the British Machine Vision Conference 2013*, BMVC 2013. British Machine Vision Association, 2013.

[62] D. Tome, M. Toso, L. Agapito, and C. Russell. Rethinking pose in 3d: Multi-stage refinement and recovery for markerless motion capture. In *2018 International Conference on 3D Vision (3DV)*, pages 474–483, 2018.

[63] G. Hua, H. Liu, W. Li, Q. Zhang, R. Ding, and X. Xu. Weakly-supervised 3d human pose estimation with cross-view u-shaped graph convolutional network. *IEEE Transactions on Multimedia*, 25:1832–1843, 2023.

[64] J. Y. Chang, G. Moon, and K. M. Lee. V2v-posenet: Voxel-to-voxel prediction network for accurate 3d hand and human pose estimation from a single depth map. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5079–5088, 2018.

[65] K. Bartol, D. Bojanić, and T. Petković. Generalizable human pose triangulation. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11018–11027, 2022.

[66] F. Bogo, A. Kanazawa, C. Lassner, P. Gehler, J. Romero, and M. J. Black. Keep it smpl: Automatic estimation of 3d human pose and shape from a single image. In B. Leibe, J. Matas, N. Sebe, and M. Welling, editors, *Computer Vision – ECCV 2016*, pages 561–578, Cham, 2016. Springer International Publishing.

[67] K. Takahashi, D. Mikami, M. Isogawa, and H. Kimata. Human pose as calibration pattern: 3d human pose estimation with multiple unsynchronized and uncalibrated cameras. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1856–18567, 2018.

[68] B. Jiang, L. Hu, and S. Xia. Probabilistic triangulation for uncalibrated multi-view 3d human pose estimation. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 14804–14814, 2023.

[69] B. Huang, Y. Shu, T. Zhang, and Y. Wang. Dynamic multi-person mesh recovery from uncalibrated multi-view cameras. In *2021 International Conference on 3D Vision (3DV)*, pages 710–720, 2021.

[70] J. Dong, Q. Shuai, J. Sun, Y. Zhang, H. Bao, and X. Zhou. imocap: Motion capture from internet videos. *Int. J. Comput. Vision*, 130(5):1165–1180, may 2022.

[71] A. Elhayek, C. Stoll, K. I. Kim, and C. Theobalt. Outdoor human motion capture by simultaneous optimization of pose and camera parameters. *Comput. Graph. Forum*, 34(6):86–98, sep 2015.

[72] N. Hasler, B. Rosenhahn, T. Thormahlen, M. Wand, J. Gall, and H.-P. Seidel. Markerless motion capture with unsynchronized moving cameras. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 224–231, 2009.

[73] T. Nägeli, S. Oberholzer, S. Plüss, J. Alonso-Mora, and O. Hilliges. Flycon: real-time environment-independent multi-view human pose estimation with aerial vehicles. *ACM Trans. Graph.*, 37(6), dec 2018.

[74] X. Zhou, S. Liu, G. Pavlakos, V. Kumar, and K. Daniilidis. Human motion capture

using a drone. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2027–2033, 2018.

[75] L. Xu, Y. Liu, W. Cheng, K. Guo, G. Zhou, Q. Dai, and L. Fang. Flycap: Markerless motion capture using multiple autonomous flying cameras. *IEEE Transactions on Visualization and Computer Graphics*, 24(8):2284–2297, 2018.

[76] M. Jacobsson, J. Willén, and M. Swarén. A drone-mounted depth camera-based motion capture system for sports performance analysis. In H. Degen and S. Ntoa, editors, *Artificial Intelligence in HCI*, pages 489–503, Cham, 2023. Springer Nature Switzerland.

[77] H. Ci, M. Liu, X. Pan, f. zhong, and Y. Wang. Proactive multi-camera collaboration for 3d human pose estimation. In *The Eleventh International Conference on Learning Representations*, 2023.

[78] C. Ho, A. Jong, H. Freeman, R. Rao, R. Bonatti, and S. Scherer. 3d human reconstruction in the wild with collaborative aerial cameras. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5263–5269, 2021.

[79] S. Kiciroglu, H. Rhodin, S. N. Sinha, M. Salzmann, and P. Fua. Activemocap: Optimized viewpoint selection for active human motion capture. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 100–109, 2020.

[80] N. Saini, E. Price, R. Tallamraju, R. Enficiaud, R. Ludwig, I. Martinovic, A. Ahmad, and M. Black. Markerless outdoor human motion capture using multiple autonomous micro aerial vehicles. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 823–832, 2019.

[81] N. Saini, E. Bonetto, E. Price, A. Ahmad, and M. J. Black. Airpose: Multi-view fusion network for aerial 3d human pose and shape estimation. *IEEE Robotics and Automation Letters*, 7(2):4805–4812, 2022.

[82] N. Saini, C.-H. P. Huang, M. J. Black, and A. Ahmad. Smartmocap: Joint estimation of human and camera motion using uncalibrated rgb cameras. *IEEE Robotics and Automation Letters*, 8(6):3206–3213, 2023.

[83] R. Tallamraju, N. Saini, E. Bonetto, M. Pabst, Y. T. Liu, M. J. Black, and A. Ahmad. Aircaprl: Autonomous aerial human motion capture using deep reinforcement learning. *IEEE Robotics and Automation Letters*, 5(4):6678–6685, 2020.

[84] A. Elhayek, C. Stoll, K. Kim, and C. Theobalt. Outdoor human motion capture by simultaneous optimization of pose and camera parameters. *Comput. Graph. Forum*, 34(6):86–98, Sept. 2015.

[85] E. Price, G. Lawless, R. Ludwig, I. Martinovic, H. H. Bülthoff, M. J. Black, and A. Ahmad. Deep neural network-based cooperative visual tracking through multiple micro aerial vehicles. *IEEE Robotics and Automation Letters*, 3(4):3193–3200, 2018.

[86] H.-S. Fang, S. Xie, Y.-W. Tai, and C. Lu. Rmpe: Regional multi-person pose

estimation. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2353–2362, 2017.

[87] T. Simon, H. Joo, I. Matthews, and Y. Sheikh. Hand keypoint detection in single images using multiview bootstrapping. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4645–4653, 2017.

[88] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In B. Leibe, J. Matas, N. Sebe, and M. Welling, editors, *Computer Vision – ECCV 2016*, pages 21–37, Cham, 2016. Springer International Publishing.

[89] G. Pavlakos, V. Choutas, N. Ghorbani, T. Bolkart, A. A. Osman, D. Tzionas, and M. J. Black. Expressive body capture: 3d hands, face, and body from a single image. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10967–10977, 2019.

[90] Xsens motion capture system. https://www.xsens.com.

[91] Y. Xiu, J. Li, H. Wang, Y. Fang, and C. Lu. Pose Flow: Efficient online pose tracking. In *BMVC*, 2018.

[92] Pytorch. https://pytorch.org/.

[93] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, 2015.

[94] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li. On the continuity of rotation representations in neural networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5738–5746, 2019.

[95] S. Shah, D. Dey, C. Lovett, and A. Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In M. Hutter and R. Siegwart, editors, *Field and Service Robotics*, pages 621–635, Cham, 2018. Springer International Publishing.

[96] Stanford Artificial Intelligence Laboratory et al. Robotic operating system, 2009.

[97] D. Mills. Internet time synchronization: the network time protocol. *IEEE Transactions on Communications*, 39(10):1482–1493, 1991.

[98] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black. Smpl: a skinned multi-person linear model. *ACM Trans. Graph.*, 34(6), oct 2015.

[99] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[100] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013.

[101] H. Fu, C. Li, X. Liu, J. Gao, A. Celikyilmaz, and L. Carin. Cyclical annealing schedule: A simple approach to mitigating KL vanishing. In J. Burstein, C. Doran, and T. Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language*

Technologies, Volume 1 (Long and Short Papers), pages 240–250, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[102] I. Kissos, L. Fritz, M. Goldman, O. Meir, E. Oks, and M. Kliger. Beyond weak perspective for monocular 3d human pose estimation. In A. Bartoli and A. Fusiello, editors, *Computer Vision – ECCV 2020 Workshops*, pages 541–554, Cham, 2020. Springer International Publishing.

[103] Opencamera-sensors android application. https://f-droid.org/en/packages/com.opencamera_sensors.app/.

[104] Distance between rotations. http://boris-belousov.net/2016/12/01/quat-dist/.

[105] P. H. Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10, Mar. 1966.

[106] Orthogonal procrustes alignment. https://docs.scipy.org/doc/scipy/reference/generated/scipy.linalg.orthogonal_procrustes.html#scipy.linalg.orthogonal_procrustes.

[107] Smplx to smpl body model transfer. https://github.com/vchoutas/smplx/tree/main/transfer_model.

[108] C. Zheng, S. Zhu, M. Mendieta, T. Yang, C. Chen, and Z. Ding. 3d human pose estimation with spatial and temporal transformers. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 11636–11645, 2021.

[109] S. Li, L. Ke, K. Pratama, Y.-W. Tai, C.-K. Tang, and K.-T. Cheng. Cascaded deep monocular 3d human pose estimation with evolutionary training data. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6172–6182, 2020.

[110] J. N. Kundu, S. Seth, V. Jampani, M. Rakesh, R. Venkatesh Babu, and A. Chakraborty. Self-supervised 3d human pose estimation via part guided novel image synthesis. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6151–6161, 2020.

[111] T. Zhang, B. Huang, and Y. Wang. Object-occluded human shape and pose estimation from a single color image. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7374–7383, 2020.

[112] Y. Wang, F. Xu, C. Pun, W. Xiao, J. Nie, J. Xiong, H. Gao, and F. Xu. New multi-view human motion capture framework. *IET Image Processing*, 14(12):2668–2674, Oct. 2020.

[113] D. Holden, T. Komura, and J. Saito. Phase-functioned neural networks for character control. *ACM Trans. Graph.*, 36(4), jul 2017.

[114] J. Li, R. Villegas, D. Ceylan, J. Yang, Z. Kuang, H. Li, and Y. Zhao. Task-generic hierarchical human motion prior using vaes. In *2021 International Conference on 3D Vision (3DV)*, pages 771–781, 2021.

[115] C. He, J. Saito, J. Zachary, H. Rushmeier, and Y. Zhou. Nemf: Neural motion fields for kinematic animation. In S. Koyejo, S. Mohamed, A. Agarwal, D. Bel-

grave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 4244–4256. Curran Associates, Inc., 2022.

[116] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.

[117] Z. Zhang. Camera calibration with one-dimensional objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(7):892–899, 2004.

[118] H. Zhang, K.-y. K. Wong, and G. Zhang. Camera calibration from images of spheres. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(3):499–502, 2007.

[119] O. Bogdan, V. Eckstein, F. Rameau, and J.-C. Bazin. Deepcalib: a deep learning approach for automatic intrinsic calibration of wide field-of-view cameras. In *Proceedings of the 15th ACM SIGGRAPH European Conference on Visual Media Production*, CVMP '18, New York, NY, USA, 2018. Association for Computing Machinery.

[120] S. Workman, C. Greenwell, M. Zhai, R. Baltenberger, and N. Jacobs. Deepfocal: A method for direct focal length estimation. In *2015 IEEE International Conference on Image Processing (ICIP)*, pages 1369–1373, 2015.

[121] L. Jin, J. Zhang, Y. Hold-Geoffroy, O. Wang, K. Blackburn-Matzen, M. Sticha, and D. F. Fouhey. Perspective fields for single image camera calibration. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17307–17316, 2023.

[122] S. Zhu, A. Kumar, M. Hu, and X. Liu. Tame a wild camera: In-the-wild monocular camera calibration. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[123] A. Gordon, H. Li, R. Jonschkowski, and A. Angelova. Depth from videos in the wild: Unsupervised monocular depth learning from unknown cameras. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8976–8985, 2019.

[124] J. Fang, I. Vasiljevic, V. Guizilini, R. Ambrus, G. Shakhnarovich, A. Gaidon, and M. R. Walter. Self-supervised camera self-calibration from video. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 8468–8475, 2022.

[125] A. Arnab, C. Doersch, and A. Zisserman. Exploiting temporal context for 3d human pose estimation in the wild. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3390–3399, 2019.

[126] Y. Hold-Geoffroy, D. Piché-Meunier, K. Sunkavalli, J.-C. Bazin, F. Rameau, and J.-F. Lalonde. A perceptual measure for deep single image camera and lens calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(9):10603–10614, 2023.

[127] Z. Teed and J. Deng. Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34 -*

35th Conference on Neural Information Processing Systems, NeurIPS 2021, Advances in Neural Information Processing Systems, pages 16558–16569. Neural information processing systems foundation, 2021. Publisher Copyright: © 2021 Neural information processing systems foundation. All rights reserved.; 35th Conference on Neural Information Processing Systems, NeurIPS 2021 ; Conference date: 06-12-2021 Through 14-12-2021.

[128] S. Jain and U. Neumann. Real-time camera pose and focal length estimation. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 1, pages 551–555, 2006.

[129] L. Citraro, P. Márquez-Neila, S. Savarè, V. Jayaram, C. Dubout, F. Renaut, A. Hasfura, H. Ben Shitrit, and P. Fua. Real-time camera pose estimation for sports fields. *Machine Vision and Applications*, 31(3), Mar. 2020.

[130] S. A. Etemad, A. Arya, A. Parush, and S. DiPaola. Perceptual validity in animation of human motion. *Comput. Animat. Virtual Worlds*, 27(1):58–71, jan 2016.

[131] P. S. A. Reitsma and N. S. Pollard. Perceptual metrics for character animation: sensitivity to errors in ballistic motion. *ACM Trans. Graph.*, 22(3):537–542, jul 2003.

[132] L. Mourot, L. Hoyet, F. L. Clerc, and P. Hellier. Underpressure: Deep learning for foot contact detection, ground reaction force estimation and footskate cleanup. *Computer Graphics Forum*, 41(8):195–206, Dec. 2022.

[133] J. Scott, B. Ravichandran, C. Funk, R. T. Collins, and Y. Liu. From image to stability: Learning dynamics from human pose. In A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, editors, *Computer Vision – ECCV 2020*, pages 536–554, Cham, 2020. Springer International Publishing.

[134] M. Loper, N. Mahmood, and M. J. Black. Mosh: motion and shape capture from sparse markers. *ACM Trans. Graph.*, 33(6), nov 2014.

[135] D. Rempe, L. J. Guibas, A. Hertzmann, B. Russell, R. Villegas, and J. Yang. Contact and human dynamics from monocular video. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V*, page 71–87, Berlin, Heidelberg, 2020. Springer-Verlag.

[136] Y. Yuan, J. Song, U. Iqbal, A. Vahdat, and J. Kautz. Physdiff: Physics-guided human motion diffusion model. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15964–15975, 2023.

[137] S. Shimada, V. Golyanik, W. Xu, and C. Theobalt. Physcap: physically plausible monocular 3d motion capture in real time. *ACM Trans. Graph.*, 39(6), nov 2020.

[138] S. Shimada, V. Golyanik, W. Xu, P. Pérez, and C. Theobalt. Neural monocular 3d human motion capture with physical awareness. *ACM Trans. Graph.*, 40(4), jul 2021.

[139] K. Xie, T. Wang, U. Iqbal, Y. Guo, S. Fidler, and F. Shkurti. Physics-based human motion estimation and synthesis from videos. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 11512–11521, 2021.

[140] B. Huang, L. Pan, Y. Yang, J. Ju, and Y. Wang. Neural mocon: Neural motion control for physically plausible human motion capture. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6407–6416, 2022.

[141] D. Holden, O. Kanoun, M. Perepichka, and T. Popa. Learned motion matching. *ACM Trans. Graph.*, 39(4), aug 2020.

[142] M. Kapadia, X. Xianghao, M. Nitti, M. Kallmann, S. Coros, R. W. Sumner, and M. Gross. Precision: precomputing environment semantics for contact-rich character animation. In *Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '16, page 29–37, New York, NY, USA, 2016. Association for Computing Machinery.

[143] C. Tessler, Y. Kasten, Y. Guo, S. Mannor, G. Chechik, and X. B. Peng. Calm: Conditional adversarial latent modelsnbsp; for directable virtual characters. In *ACM SIGGRAPH 2023 Conference Proceedings*, SIGGRAPH '23, New York, NY, USA, 2023. Association for Computing Machinery.

[144] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne. Deepmimic: example-guided deep reinforcement learning of physics-based character skills. *ACM Trans. Graph.*, 37(4), jul 2018.

[145] M. Hassan, Y. Guo, T. Wang, M. Black, S. Fidler, and X. B. Peng. Synthesizing physical character-scene interactions. In *ACM SIGGRAPH 2023 Conference Proceedings*, SIGGRAPH '23, New York, NY, USA, 2023. Association for Computing Machinery.

[146] X. B. Peng, G. Berseth, and M. van de Panne. Dynamic terrain traversal skills using reinforcement learning. *ACM Trans. Graph.*, 34(4), jul 2015.

[147] J. Rajasegaran, G. Pavlakos, A. Kanazawa, and J. Malik. Tracking people by predicting 3d appearance, location and pose. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2730–2739, 2022.

[148] Y. Zheng, R. Shao, Y. Zhang, T. Yu, Z. Zheng, Q. Dai, and Y. Liu. Deepmulticap: Performance capture of multiple characters using sparse multiview cameras. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6219–6229, 2021.

[149] Y. Sun, W. Liu, Q. Bao, Y. Fu, T. Mei, and M. J. Black. Putting people in their place: Monocular regression of 3d people in depth. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13233–13242, 2022.

[150] R. Khirodkar, S. Tripathi, and K. Kitani. Occluded human mesh recovery. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1705–1715, 2022.

[151] Z. Qiu, Q. Yang, J. Wang, H. Feng, J. Han, E. Ding, C. Xu, D. Fu, and J. Wang. Psvt: End-to-end multi-person 3d pose and shape estimation with progressive video transformers. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21254–21263, 2023.

[152] Y. Sun, Q. Bao, W. Liu, Y. Fu, M. J. Black, and T. Mei. Monocular, one-stage, regression of multiple 3d people. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 11159–11168, 2021.

[153] J. Zhang, D. Yu, J. H. Liew, X. Nie, and J. Feng. Body meshes as points. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 546–556, 2021.

[154] A. Zanfir, E. Marinoiu, M. Zanfir, A.-I. Popa, and C. Sminchisescu. Deep network for the integrated 3d sensing of multiple people in natural images. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 8420–8429, Red Hook, NY, USA, 2018. Curran Associates Inc.

[155] E. Corona, G. Pons-Moll, G. Alenyà, and F. Moreno-Noguer. Learned vertex descent: A new direction for 3d human model fitting. In S. Avidan, G. Brostow, M. Cissé, G. M. Farinella, and T. Hassner, editors, *Computer Vision – ECCV 2022*, pages 146–165, Cham, 2022. Springer Nature Switzerland.

[156] J. Song, X. Chen, and O. Hilliges. Human body model fitting by learned gradient descent. In A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, editors, *Computer Vision – ECCV 2020*, pages 744–760, Cham, 2020. Springer International Publishing.

[157] A. Zanfir, E. G. Bazavan, M. Zanfir, W. T. Freeman, R. Sukthankar, and C. Sminchisescu. Neural descent for visual 3d human pose and shape. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14479–14488, 2021.

[158] H. Zhang, Y. Tian, Y. Zhang, M. Li, L. An, Z. Sun, and Y. Liu. Pymaf-x: Towards well-aligned full-body model regression from monocular images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 45(10):12287–12303, oct 2023.

[159] G. Pavlakos, L. Zhu, X. Zhou, and K. Daniilidis. Learning to estimate 3d human pose and shape from a single color image. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 459–468, 2018.

[160] Y. Zhang, H. Zhang, L. Hu, H. Yi, S. Zhang, and Y. Liu. Real-time monocular full-body capture in world space via sequential proxy-to-motion learning. *arXiv preprint arXiv:2307.01200*, 2023.

[161] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics*, 37(6):1874–1890, 2021.