Flycon

Real-time environment-independent multi-view human pose estimation with aerial vehicles

Nageli, Tobias; Oberholzer, Samuel; Pluss, Silvan; Alonso-Mora, Javier; Hilliges, Otmar

DOI
10.1145/3272127.3275022

Publication date
2018

Document Version
Final published version

Published in
ACM Transactions on Graphics

Flycon: Real-time Environment-independent Multi-view Human Pose Estimation with Aerial Vehicles

TOBIAS NÄGELI, AIT Lab, ETH Zurich
SAMUEL OBERHOLZER, AIT Lab, ETH Zurich
SILVAN PLÜSS, AIT Lab, ETH Zurich
JAVIER ALONSO-MORA, Delft University of Technology
OTMAR HILLIGES, AIT Lab, ETH Zurich

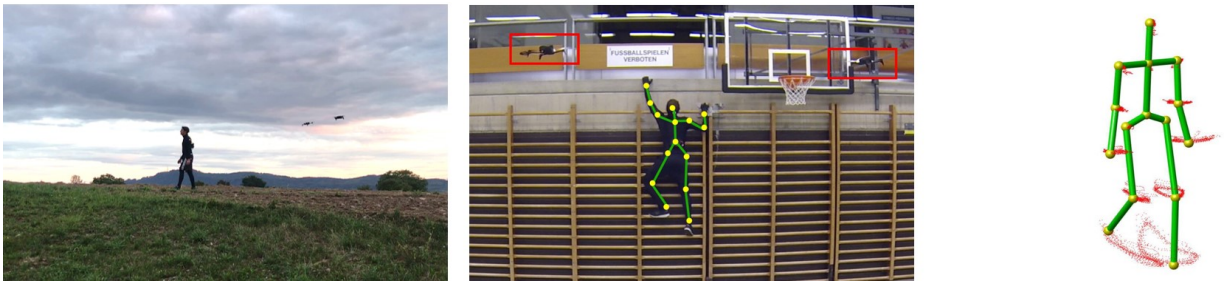

Fig. 1. We propose a novel method for environment-independent estimation of human poses in real-time. We demonstrate the proposed system in a number of compelling outdoor (left) and indoor (middle) experiments. We estimate the positions of a quadrotor swarm as well as the full human pose in real-time. A *model predictive controller* computes optimal quadrotor inputs to follow the human and always keep the markers visible (middle). Our method can accurately estimate articulated motion over long time frames and distances, In the right figure we show the accumulated joint positions, relative to the center of the person, over a 170m long walking sequence.

We propose a real-time method for the infrastructure-free estimation of articulated human motion. The approach leverages a swarm of camera-equipped flying robots and *jointly* optimizes the swarm's and skeletal states, which include the 3D joint positions and a set of bones. Our method allows to track the motion of human subjects, for example an athlete, over long time horizons and long distances, in challenging settings and at large scale, where fixed infrastructure approaches are not applicable. The proposed algorithm uses active infra-red markers, runs in real-time and accurately estimates robot and human pose parameters online without the need for accurately calibrated or stationary mounted cameras. Our method i) estimates a global coordinate frame for the MAV swarm, ii) jointly optimizes the human pose and relative camera positions, and iii) estimates the length of the human bones. The entire swarm is then controlled via a model predictive controller to maximize visibility of the subject from multiple viewpoints even under fast motion such as jumping or jogging. We demonstrate our method in a number of difficult scenarios including capture of long locomotion sequences at the scale of a triplex gym, in non-planar terrain, while climbing and in outdoor scenarios.

Authors' addresses: Tobias Nägeli, AIT Lab, ETH Zurich; Samuel Oberholzer, AIT Lab, ETH Zurich; Silvan Plüss, AIT Lab, ETH Zurich; Javier Alonso-Mora, Delft University of Technology; Otmar Hilliges, AIT Lab, ETH Zurich.

0730-0301/2018/11-ART182 $15.00
https://doi.org/10.1145/3272127.3275022

CCS Concepts: • **Computing methodologies** → **Motion capture**; **Reconstruction**; **Robotic planning**; **Motion path planning**;

Additional Key Words and Phrases: human pose estimation, robotics

ACM Reference Format:
Tobias Nägeli, Samuel Oberholzer, Silvan Plüss, Javier Alonso-Mora, and Otmar Hilliges. 2018. Flycon: Real-time Environment-independent Multi-view Human Pose Estimation with Aerial Vehicles. *ACM Trans. Graph.* 37, 6, Article 182 (November 2018), 14 pages. https://doi.org/10.1145/3272127.3275022

1 INTRODUCTION

Many graphics applications such as character animation for games, sports, biomechanics, VR, and AR rely on accurate human pose information, and virtually every modern movie production leverages Motion Capture (Mocap) systems for special effects. Most commonly, such systems are camera based, either relying on body-worn markers, or more recently even work markerless. Multi-view approaches can now be highly accurate and sometimes provide dense surface reconstructions. The maturing of camera based motion capture technology in turn leads to a desire to use it in increasingly challenging scenarios such as with fast moving actors, large scale scenes and even in outdoors settings. However, practically all existing approaches require a set of environment mounted, accurately calibrated cameras looking into a capture area of fixed size. This requirement for stationary cameras makes application in these settings very tedious, costly and sometimes entirely infeasible.

In this paper we propose an environment-independent approach to multi-view human motion capture that leverages an autonomous swarm of micro aerial vehicles (MAVs), or drones. The drones carry

cameras trained on the subject of interest, who wears a sparse set of active LED markers. The 2D positions of these markers are extracted from the images and the 3D joint positions of the human skeleton are estimated in real-time.

Our approach we address several challenges: First, and in contrast to traditional camera localization approaches that make rigid scene assumptions, the 3D joint locations move in an articulated non-rigid fashion. Second, the cameras move relative to the human and their configuration changes dynamically, which is in contrast to typical human pose estimation approaches where the cameras are assumed to be stationary and calibrated. Third, we do not rely on any external signal, such as GPS for positioning, making our approach applicable both indoors and outdoors. Our method enables motion capture in previously difficult or entirely infeasible scenarios such as continuously reconstructing the full body pose of an athlete throughout an entire workout or capturing actors in remote and difficult to reach locations, for example while climbing.

More concretely we propose a completely self-contained method for the *joint* estimation and control of the states of multiple MAVs and of 3D human skeletal configuration. The proposed algorithm runs in real-time and accurately estimates the positions of the robot swarm and the human pose parameters. Furthermore, we compute in real-time drone trajectories to keep the cameras trained on the subject and therefore the markers in view of the cameras.

Our algorithm is inspired by recursive filtering techniques used in robot localization problems. However, in contrast to classical scene reconstruction and camera localization algorithms, the tracked 3D points are not static but move in a complex, articulated fashion. To make this nonlinear state estimation problem of a discrete-time stochastic system tractable in real-time, we pose it as an indirect iterated extended Kalman filter (IEKF) which computes the state estimates as maximum a posteriori (MAP) estimates. In typical camera localization formulations, states are estimated relative to a global world reference frame, which causes the uncertainty with respect to the origin to grow as one moves further away [Castellanos et al. 2004]. To avoid this uncertainty growth, we use a formulation where 3D points and the world origin are expressed with respect to a *moving* reference frame (the lead drone of the swarm). During state propagation and update, linearization is performed around the estimated lead camera frame. In consequence, little linearization error is accumulated over time. This allows us to follow the subject over long distances without drift or loss in pose estimation accuracy.

To our knowledge, we are the first to frame localization and optimal control of a robotic swarm and the estimation of human articulated motion as a *joint* optimization problem and to provide a real-time implementation. Our method, at every frame, i) collects images from all drones, detects and labels 2D joint positions, ii) estimates the state of a *leader* robot from onboard sensors (e.g., IMU, down-facing optical flow sensor), iii) estimates the joint positions of the human skeleton (and the bone lengths) and optimizes the relative positions and orientations of the multi-robot swarm; iv) finally, it computes control inputs for the drones via model-predictive control (MPC) to keep markers observable under subject motion.

We demonstrate our method in a number of compelling usage scenarios that include fast motion, such as running or jumping jacks, and that capture long trajectories (hundreds of meters). Furthermore, we demonstrate the benefits of environment independence by following a subject over different elevations and in difficult terrain such as a climbing wall (see Fig. 1, middle).

2 RELATED WORK

Our work brings together state-of-the-art robotics research on MAV (swarm) state estimation and control and algorithms for motion capture from the computer graphics and vision literature. Here we briefly review the most pertinent work.

Camera-based motion capture: Camera-based capture of articulated human motion is at the core of many graphics and related application domains. Commercial solutions require wearing of marker suits or gloves and depend on multiple calibrated cameras mounted in the environment. To overcome these constraints much research has been devoted to developing marker-less approaches from *multiple cameras* (cf. [Moeslund et al. 2006]). Often such methods trade-in high quality results with offline processing [Ballan et al. 2012; Bregler and Malik 1998; Starck and Hilton 2003] but recently real-time approaches [de Aguiar et al. 2008; Elhayek et al. 2017; Rhodin et al. 2015; Stoll et al. 2011] have been proposed. Such approaches typically fit a skeletal model to image data or represent the human as a collection of Gaussians [Rhodin et al. 2015]. Other approaches to real-time performance include combining discriminative and generative approaches [Elhayek et al. 2017; Oikonomidis et al. 2012]. However, such multi-view approaches always assume stationary, well calibrated cameras and are therefore not suitable in mobile and outdoors scenarios. More recently pose estimation methods have exploited deep convolutional networks (ConvNets) for body-part detection in fully unconstrained *monocular* images [Chen and Yuille 2014; Newell et al. 2016; Tompson et al. 2014; Toshev and Szegedy 2014; Wei et al. 2016]. However, these methods only capture 2D skeletal information. Predicting 3D pose directly from 2D RGB images has been demonstrated using offline [Bogo et al. 2016; Tekin et al. 2016; Zhou et al. 2016] methods and in online settings [Mehta et al. 2017]. Monocular *depth* cameras provide additional information and have been shown to aid robust skeletal tracking [Ganapathi et al. 2012; Shotton et al. 2013; Taylor et al. 2012] and enable dense surface reconstruction even under deformation [Dou et al. 2016; Newcombe et al. 2015; Zollhöfer et al. 2014]. Multiple, specialized structured light scanners have been used to capture high-fidelity dense surface reconstructions of humans [Pons-Moll et al. 2015].

Our approach relies on multiple cameras to estimate skeletal motion and we believe much of the above work is complementary to ours in that marker-less techniques could serve as input to our joint camera and human pose estimation pipeline. In contrast to the above work, our method does not require any infrastructure or calibrated cameras. Because the cameras are airborne all measurements are noisy, unreliable and measure only relative quantities, making the task significantly harder.

Inertial measurement units: Attaching sensors directly onto the body overcomes the need for line-of-sight and enables use without infrastructure. Inertial measurements units (IMU) are the most

prominent type of sensor used for pose estimation. Commercial systems rely on 17 or more IMUs, which fully constrain the pose space, to attain accurate skeletal reconstructions via inverse kinematics [Roetenberg et al. 2007]. It has been shown that good performance can be achieved with fewer sensors by exploiting data-driven methods [Liu et al. 2011; Schwarz et al. 2009; Tautges et al. 2011] or by taking temporal consistency into account, albeit at the cost of high computational cost and therefore offline processing [von Marcard et al. 2017]. While IMUs provide mobility and accuracy, above approaches inherently require user instrumentation. Furthermore, they rely on sophisticated models of the human and hence can not easily be generalized to other subjects. Our implementation currently also requires body-worn markers but in principle can work markerless. More importantly, we optimize 3D point coordinates and only model the human by connecting adjacent joints, thus reducing computational cost of the optimization and making the method applicable to all kinds of articulated motion.

MAVs in graphics and vision: With the consumerization of aerial robots, the graphics community has recently proposed a number of tools and algorithms for the planning of physically feasible quadrotor camera trajectories for aerial videography. Such tools allow for planning of aerial shots in 3D virtual environments [Gebhardt et al. 2016; Joubert et al. 2015; Roberts and Hanrahan 2016] and employ offline optimization methods to ensure that both aesthetic objectives and robot modeling constraints are considered. The methods of [Joubert et al. 2015] and [Gebhardt et al. 2016] generate quadrotor trajectories given user-defined space-time keyframes, whereas the method proposed in [Roberts and Hanrahan 2016] takes physically infeasible trajectories and computes the closest possible feasible trajectory by re-timing the velocities subject to a non-linear quadrotor model. [Gebhardt et al. 2018] studies factors influencing perception of aerial video and propose a.n optimization scheme based on these results. All of the above methods are *offline* and cannot generate control inputs for use in *dynamic* environments. Using a Model Predictive Control (MPC) formulation, [Naegeli et al. 2017] optimizes cinematographic constraints, such as visibility and position on the screen, subject to robot constraints for a single quadrotor. [Nägeli et al. 2017] extends this work to multiple drones and allows actor-driven tracking on a geometric path. The robotics literature has proposed methods to recover the 3D trajectory of a moving person from a MAV mounted camera while mapping the environment [Li et al. 2016; Lim and Sinha 2015]. In contrast, the objective of this paper is to reconstruct the full 3D body pose of a moving subject while planning the MAV trajectories to keep markers in view. For this task, multiple quadrotors are necessary and their position has to be estimated alongside the skeletal joint positions.

In [Huang et al. 2018] the authors use real time monocular pose reconstruction to do a pose reconstruction and use it for a through-the-lens filming system. The user can define the viewpoints for filming in a virtual scene.

In this sense our work is most closely related to [Xu et al. 2017] who leverage depth-cameras mounted on three drones together with a deformable surface energy for dense surface reconstruction of a dynamic user. However, the proposed method relies on depth data, a pre-scanned template mesh (which is deformed and used for data fitting) and target tracking is performed in real-time via [Li et al. 2016], whereas pose reconstruction is reported to run at 3 frames per *minute* on a high-end PC. Our method runs entirely in real-time, while it tracks articulated motion and controls the position of the MAVs. The method estimates the articulated motion of the user from monocular imagery only (we currently rely on markers fixed to the person) and thus can work indoors and outdoors, where depth cameras struggle in direct sunlight.

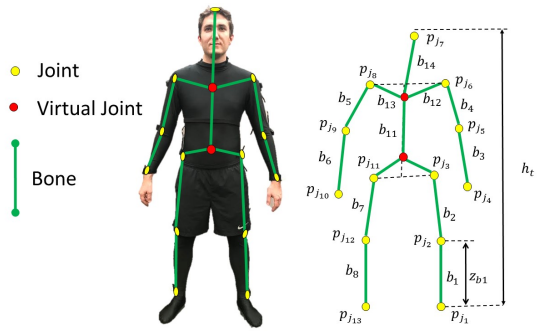

Fig. 2. Schematic of the states used to model the human skeleton $\mathbf{x}_h$. The estimated skeleton constists of 13 real joint markers (yellow), two virtual markers (red) and 14 bones (green). The virtual markers are computed using the physical markers and are introduced for better bone length estimates.

Multi-robot systems: Multi-robot teams are widely studied in robotics, including groups of aerial [Alonso-Mora et al. 2018; Basiri et al. 2013; Lupashin et al. 2011; Michael et al. 2010] robots. To stabilize a formation, each agent requires exact positional knowledge [Pugh and Martinoli 2006]. Existing approaches to formation flight therefore rely either on low precision sensors, which result in large inter-robot distances, or on external infrastructure. Methods for infrastructure-free formation control have been proposed by [Nägeli et al. 2014], albeit requiring the cameras to be trained on the other members of the swarm, rendering it unsuitable for subject tracking. We do not rely on any infrastructure or external tracking and estimate the drone position and human pose in a single, combined optimization framework.

3 OVERVIEW

To solve this challenging problem of online human pose estimation using MAV's in unstructured environments, we make the following key *assumptions*:

(1) **Fast Sampling:** The camera frame rates and our algorithm are fast ($30Hz$) with respect to human motion. Hence, we assume that the pixel displacement from image to image is small for all marker positions.
(2) **Constant bone-length:** Adjacent joints are linked via bones of constant, yet unknown, length. Since the markers are not rigidly attached to the bones, we allow small changes and estimate bone-lengths online, without any prior calibration.
(3) **Observability:** Marker's seen from at least two cameras are called *observable*. The location of individual *unobservable* markers can be predicted via the bone-length constraint.

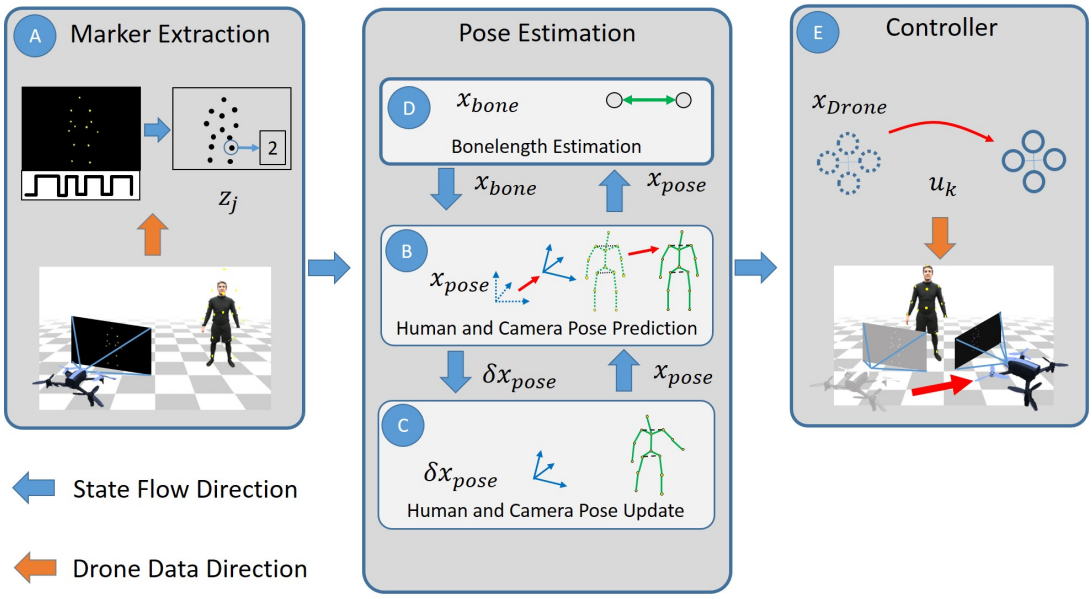

Fig. 3. Method Overview. Left to right: A subject wears a sparse set of active LEDs from which we extract 2D joint detections z_j. A recursive *error-state* filter formulation jointly estimates the position and orientation of multiple flying cameras ($\mathbf{x}_{\text{drone}}$), and the positions of the 3D joints and the length of bones ($\mathbf{x}_{\text{pose}}$). Finally we compute feasible trajectories and corresponding control inputs u_k for the MAVs to keep the human in view.

(4) **Predictive control:** We assume that trajectories can be generated to accurately track the human and to keep it in the camera's frame (see Sec. 6.2). This allows for initialization of the pose *a priori* estimate from the drone trajectory.

We formulate this optimization problem in a recursive filtering framework that allows us to naturally link states and measurements over time and provides a straightforward integration of sensor data as priors for each iteration of the optimization.

Additionally, we accurately estimate the states of all MAVs by fusing the optimized camera poses with odometry measurements attained from onboard sensors, which include downward looking optical flow sensors and IMUs. The drone positions are then controlled to maximize visibility of the subject. Our algorithm, illustrated in Fig. 3 iteratively performs the following steps:

(A) Collect images from all drones, detect and label joints.
(B) Predicts and estimates the state of a *leader* robot from onboard sensors (e.g., IMU, down-facing optical flow sensor)
(C) Perform a joint reconstruction of the articulated human pose and the camera states to obtain the position of the joints and the position of each drone-mounted camera. Update the pose state $\mathbf{x}_{\text{pose}}$. Fuse the camera pose estimate with proprioceptive sensor data (IMU, optical flow) to estimate the full drone state.
(D) Estimate the length of the bones online.
(E) Compute drone inputs, via a receding horizon controller.

4 MODELING AND NOTATION

We now provide the used notation and a brief overview of the model of the human and the multi-robot swarm used in our non-linear estimation and control formulation.

4.1 Terminology

In this paper we define the term *Pose* as the joint-angle configuration of the human, together with the position and orientation of all cameras. If we talk about a specific pose, we specify this by writing *Camera Pose* or *Human Pose*. We call all quadrotors together a swarm. The swarm together with the human is denoted as a formation.

4.2 Notation

Here we introduce the most important notation. For a full treatment we refer to Appendix A. Throughout this paper, states $\mathbf{x}$ are denoted in bold. For a given state $\mathbf{x}$, we denote the estimated state as $\hat{\mathbf{x}}$, a measurement by its measurement function $h(\mathbf{x})$ and an estimated measurement by $h(\hat{\mathbf{x}})$. Residuals are denoted by ρ. We denote points in 3D as $\mathbf{p}$ with a name as subscript, e.g. $\mathbf{p}_q$ for $\mathbf{p}_{\text{quadrotor}}$. A relative vector between two points $\mathbf{p}_a$ and $\mathbf{p}_b$ is denoted as $\mathbf{r}_{ab}$. A superscript $\mathbf{r}^c_{ab}$ indicates the vector $\mathbf{r}_{ab}$ is expressed in frame C. Without superscript the vector is expressed in the (global) inertial frame I.

4.3 Human Pose

The pose of the human is defined by a set of m joints, modelled as 3D points, and their connecting bones. Fig. 2 shows the assumed mapping between bones and joints. The state of all *joints* is denoted by $\mathbf{x}_j$ which contains the position of the m individual *joints* that define the human pose: $\mathbf{x}_j = [\mathbf{p}_{j_1}, \ldots, \mathbf{p}_{j_m}] \in \mathbb{R}^{3m}$. All joints are connected by bones of a certain length. We denote with $\mathbf{x}_b$ the *bone-lengths* state vector: $\mathbf{x}_b = [b_1, \ldots, b_{m-1}] \in \mathbb{R}^{(m-1)}$. The bone-lengths are assumed to be constant, but unknown, and therefore treated as *bias states* for which the exact values are estimated online.

4.4 Drones and Cameras

We follow the drone model proposed in [Naegeli et al. 2017]. We consider n drones, each of them equipped with a camera. The state of each quadrotor is given by its position $\mathbf{p}_q \in \mathbb{R}^3$, its velocity $\dot{\mathbf{p}}_q \in \mathbb{R}^3$and its orientation, i.e. roll Φ_q, pitch Θ_q and yaw ψ_q. For drone i, its camera is attached to the drone with a gimbal of controllable pitch θ_g and yaw ψ_g. For brevity, we assume the camera position and the quadrotor position to be identical. For a detailed description of the full non-linear drone model refer to Appendix B. The full state vector of a drone is defined as:

$$\mathbf{x}_d = [\text{ Quadrotor} \mid \text{Camera }] = [\mathbf{p}_q, \dot{\mathbf{p}}_q, \Phi_q, \Theta_q, \psi_q, \theta_g, \psi_g] \in \mathbb{R}^{11}.$$

The Parrot Bebop's SDK demands angles as input and hence we represent rotations as such for the *control* of the robot and its gimbal. We denote the set of inputs by $\mathbf{u}$, containing the velocity of the drone in the body-z axis, the desired roll and pitch angles of the drone, the angular speed around the body-z axis and the pitch and yaw rates of the camera gimbal. For our *optimization* we always represent rotations as quaternions to avoid gimbal locking. For instance, the 3D camera orientation is denoted by the quaternion $\bar{q}_c = \bar{q}(\theta_g, \psi_g) \in SO(3)$ and the orientation of the drone by $\bar{q}_q = \bar{q}(\Phi_q, \Theta_q, \psi_q) \in SO(3)$.

4.5 State-space Structure and Filtering Strategy

Since all robots and the human move dynamically, solving the problem considered here requires the estimation of the full system state, which consists of the drone states and the human state. This leads to a very large state-space of $11n+3m+(m-1)$. In our implementation this dimensionality is 73. Since the computational cost of a single filter iteration grows cubically with the number of states, a naive implementation would not run in real-time. We leverage two key ideas to reduce the computational cost and render this problem tractable in real-time. (1) We separate the constant, but unknown *bias sates* from the state-space. This technique is known as separate-bias or two-stage estimation [Friedland 1969; Hsieh 2000]. See Sec. 5.4 for details of the online bone length estimation $\mathbf{x}_b$. (2) We separate the drone states that are not necessary for the human pose estimation, but that have fast dynamics, from the overall state-space. Following [Gibbs 2011] we refer to these separable states as *control* states. See Sec. 5 for details on human and drone state estimation.

Based on concepts (1) and (2), we can structure the state space into three groups:

- **Pose**: States used for the human pose estimation.
- **Control**: Additional states used for quadrotor control.
- **Bias**: Bone lengths that are constant but unknown.

We now restructure the state space accordingly:

$$\begin{aligned}\mathbf{x} &= [\underbrace{\text{Cameras, Joints}}_{\text{Pose State}} \mid \underbrace{\text{Quadrotors}}_{\text{Control State}} \mid \underbrace{\text{Bonelength}}_{\text{Bias State}}] \\ &= [\underbrace{\mathbf{x}_{c_1}, \ldots, \mathbf{x}_{c_n}, \mathbf{x}_j}_{\text{Pose State: } \mathbf{x}_{\text{pose}}} \mid \underbrace{\mathbf{x}_{q_1}, \ldots, \mathbf{x}_{q_n}}_{\text{Control State}} \mid \underbrace{\mathbf{x}_b}_{\text{Bias State}}],\end{aligned}$$

To solve this problem, we apply an *error state* Kalman filtering (ESKF) strategy to pose state estimation. This allows us to circumvent dynamic modeling errors [Roumeliotis et al. 1999], singularities in the estimation of the covariance matrices of the camera poses [Lefferts et al. 1982] and filter inconsistencies caused by unobservable states [Castellanos et al. 2004].

Furthermore, the entire constellation of poses is relative to each other and hence, the solution would be free to drift arbitrarily. For consistency in the estimation, a global position reference for the human-multi-robot team is required. To address this issue we first estimate the global pose of one drone, which sub-sequentially is used as reference frame to express all other poses and feature locations. However, even this reference drone has no access to drift-free positional information and hence a recursive filter would incur in growing uncertainty in the pose estimate. To alleviate this issue, we adopt a robo-centric EKF formulation inspired by [Castellanos et al. 2004; de Palézieux et al. 2016]. In our formulation, the world reference frame and feature locations are expressed with respect to a moving reference frame that is updated to the current estimated *leader* pose after every filter update. The (unobservable) uncertainty of the absolute camera position, traditionally associated with the current estimate, is now associated with the world reference pose. Linearization is now performed around the low uncertainty current estimate of the camera pose, avoiding accumulation of error.

The above robo-centric estimation lends itself to a formalization as error-state filter [Castellanos et al. 2004; Roumeliotis et al. 1999]. Since we assume small motion between frames, we can decouple the absolute, yet unknown, pose state $\mathbf{x}_{\text{pose}}$ into an estimated prior state and an additional small error state $\delta\mathbf{x}_{\text{pose}}$:

- **Prior State**: The prior state $\mathbf{x}_{\text{prior}}$ is the *a priori* estimate of the pose $\mathbf{x}_{\text{pose}}$ using all available onboard sensors (imu, optical flow).
- **Error State**: The error state $\delta\mathbf{x}_{\text{pose}}$ describes the residual between the *a priori* and the *a posteriori* estimate of the pose state $\mathbf{x}_{\text{pose}}$ after fusing prior estimates and camera measurements (i.e., marker locations).

We can now write the *a posteriori* estimate of the pose state:

$$\mathbf{x}_{\text{pose}} := \mathbf{x}_{\text{prior}} \otimes \delta\mathbf{x}_{\text{pose}}. \tag{1}$$

where $\otimes$ denotes the fusion of the *a priori* total state and the *a posteriori* error state. Linear quantities are updated additively, while rotational entries are updated multiplicatively. Note that $\mathbf{x}_{\text{pose}}$ is the desired quantity we seek to optimize. That is, at the end of the procedure detailed in Alg 1, $\mathbf{x}_{\text{pose}}$ will contain the estimate of the camera swarm and the skeletal configuration.

5 JOINT CAMERA AND HUMAN STATE ESTIMATION

Given the above filtering structure, recovering the skeletal configuration of the subject alongside the position of the camera drones now boils down to estimating the *Pose* state $\mathbf{x}_{\text{pose}}$ accurately. This *Pose* state estimate is then used to compute the control inputs for the swarm for the next timestep (Sec. Sec. 6.2) to ensure observability of the human skeleton. We attain this estimate online via recursive estimation, alternating *propagation* and *update* steps.

5.1 Pose State Propagation

To accurately estimate the humans 3D joint positions, we first need to establish where the cameras are relative to the subject - this itself

Algorithm 1 Joint Skeleton and Camera Pose Estimation

loop
  get camera images and label joint positions: ▷ Sec. 7
  for every camera **do**
    $\mathbf{z}_{\text{Blobs}} \leftarrow$MarkerDetection(Image)
    $\mathbf{z}_j \leftarrow$MarkerLabeling(Blobs)
  end for

  estimate human and drone pose: ▷ Sec. 5
  $[\mathbf{x}_{\text{pose}}] \leftarrow$JointPoseEstimation($\mathbf{z}_j, \mathbf{x}_{\text{pose}}$)
  $[\mathbf{x}_b] \leftarrow$BonelengthEstimation($\mathbf{x}_{\text{pose}}$)

  for every drone **do**
    full drone state estimation:
    $[\mathbf{x}_d] \leftarrow$DroneStateEstimation($\mathbf{z}_{odo}, \mathbf{x}_{\text{pose}}$)

    compute drone inputs: ▷ Sec. 6.2
    update cost & constraints, solve MPC Eq. (13)
    apply_inputs($\mathbf{u}^0$) to drone
  end for
end loop

is in the absence of global positioning an unconstrained problem. To initialize our optimization we use the sensors of the drones to get an a-priori estimate $\mathbf{x}_{\text{prior}}$ of the pose state $\mathbf{x}_{\text{pose}}$. We denote by $\mathbf{z}_{\text{odo}\,i}$ the estimated position of drone i, given by an onboard optical flow estimation algorithm [Bristeau et al. 2011]. Note that at this point, none of the drones has any information about the location of the remaining $n-1$ drones.

To establish the relative transformations, the formation requires an absolute position reference. Since the position dynamics of n drones have only $3(n-1)$ independent degrees of freedom [Nägeli et al. 2014], we require only one absolute position estimate. To approximate this global reference, we pick one drone, which we refer to as the *leader* drone and use the associated odometry estimate $\mathbf{z}_{\text{odo}\,1}$ and the resulting position $\mathbf{p}_{q_1}$, as global position estimate of the entire constellations. Note that this estimate drifts over time but experimentally we found it to be sufficiently accurate even over long distances and time horizons (see accompanying video).

Camera pose propagation: Following Sec. 3, we assume (4) that all drones and the human are *approximately* translating with the global frame, defined by the lead camera. For the lead drone we consider that its position estimate is given by its odometry,

$$\mathbf{p}_{q_1}^{k+1} \leftarrow \mathbf{z}_{\text{odo}\,1} \qquad \text{Drone 1 position propagation.}$$

We can then compute the translation $\boldsymbol{\Delta}$ of the lead drone in one time step, $\boldsymbol{\Delta} := \mathbf{z}_{\text{odo}\,1} - \mathbf{p}_{q_1} = \mathbf{p}_{q_1}^{k+1} - \mathbf{p}_{q_1}$.

From assumption (4), the position estimate of the remaining drones can be initialized by adding the position change $\boldsymbol{\Delta}$ of the lead drone to the latest position estimate. For drone $i > 1$,

$$\mathbf{p}_{q_i}^{k+1} \leftarrow \mathbf{p}_{q_i} + \boldsymbol{\Delta} \qquad \text{Drone i>1 position propagation.}$$

Following the covariance update proposed in [Castellanos et al. 2004; de Palézieux et al. 2016], we marginalize out the position error covariance of the position dynamics for the leader drone, $P_{p_1}^{k+1} = \mathbf{0} \in \mathbb{R}^{3\times3}$, and for all remaining drones, $P_{p_i}^{k+1} = P_{p_i}^{k} + Q_{pos} \in \mathbb{R}^{3\times3}$. The parameter Q_{pos} is a diagonal matrix containing the standard deviation of the expected position change from the initial state, and is a tunable parameter.

Human pose propagation: Again applying assumption (4), the estimated center of mass of the human is also translated by $\boldsymbol{\Delta}$,

$$\mathbf{x}_j^{k+1} \leftarrow \mathbf{x}_j + \boldsymbol{\Delta}. \tag{2}$$

The covariance of the skeletal joints state is then given by $P_j^{k+1} = P_j^k + Q_j$. Q_j is again a diagonal matrix containing the standard deviation of the expected position change from the initial state.

5.2 Filter measurements

After having established an initialization of the positions, we estimate $\delta\mathbf{x}_{\text{pose}}$ and hence update the pose state $\mathbf{x}_{\text{pose}}$ using the following measurements:

- **Camera measurements**: Pixel-coordinates of the measured marker positions from each camera, denoted by $\mathbf{z}_j$.
- **Bone-length measurements**: denoted by $\mathbf{x}_b$, and obtained with the estimated bias state, which we discuss in Section 5.4.

Joint residual: At each iteration we receive new camera measurements, in our implementation 2D marker positions extracted from the images. With these measurements we perform an *update* step of the filter. More specifically, we use the pixel measurements of all markers seen by all cameras to minimize the residual between the *estimated* marker positions and the incoming measurements.

Without loss of generality, but slight abuse of notation, we describe the measurement residual of a joint a seen by camera i. To build the residual ρ_j between the joint measurement and the *estimated* joint measurement, we project the *estimated* joint position $\mathbf{p}_j \subset \mathbf{x}_{\text{prior}}$ into the camera frame using the prior camera position $\mathbf{p}_q \subset \mathbf{x}_{\text{prior}}$ and orientation $\bar{q}_c \subset \mathbf{x}_{\text{prior}}$. The projection is performed via a standard pinhole camera model [Hartley and Zisserman 2003]. The estimated 2D joint position is then attained via a projection into undistorted pixel coordinates,

$$\mathbf{h}_j(\mathbf{x}_{\text{prior}}) = \begin{bmatrix} \mathbf{m}_x f_u + C_u \\ \mathbf{m}_y f_v + C_v \end{bmatrix} \quad \text{with} \quad \mathbf{m} = \frac{1}{\mathbf{r}_z^c}\begin{bmatrix} \mathbf{r}_x^c \\ \mathbf{r}_y^c \end{bmatrix}, \tag{3}$$

where $\mathbf{r} = \mathbf{p}_j - \mathbf{p}_q$ is the relative vector between the joint estimate and the camera center, $\mathbf{r}^c = R(\bar{q}_c)\mathbf{r}$ is the vector $\mathbf{r}$ rotated into the camera frame and $\mathbf{r}_z^c$ is the z-component of $\mathbf{r}^c$. Pixel coordinates m are computed via the camera intrinsics $f = [f_u, f_v]$ and $C = [C_u, C_v]$. The residual for joint a seen by camera i is then given by

$$\rho_j = \mathbf{v} - \mathbf{h}_j(\mathbf{x}_{\text{prior}}) \in \mathbb{R}^2, \tag{4}$$

where $\mathbf{v} \subset \mathbf{z}_j$ denotes the measurement for joint a and camera i.

Bone length residual: Conceptually, we treat the bone lengths as constant. However, for convenience we follow [Friedland 1969] and include them as measurements affected by zero mean Gaussian noise into our computations.

An individual bone-length prediction can be computed as the Euclidean distance between two adjacent 3D joint positions $\mathbf{p}_{j_a}$ and

$\mathbf{p}_{j_b}$. It is therefore given by $h_b = \|\mathbf{p}_{j_a} - \mathbf{p}_{j_b}\|$. For a single bone i, the bone-length residual ρ_b is then

$$\rho_b = b_i - h_b(\mathbf{x}_{\text{prior}}) \in \mathbb{R}, \tag{5}$$

where $b_i \subset \mathbf{x}_b$ is the constant, but a-priori unknown, bone-length. We estimate this quantity online, see Sec. 5.4 for details. Intuitively, this residual will ensure that the solution converges to a skeletal configuration in which bones have a constant length. While the physical bone does not change its length at all, this formulation allows for slight variation in relative joint distances. This is due to the difficulties of integrating hard-constraints into recursive filters and due to the fact that the markers and their detections may move relative to the actual joint. Furthermore, this step makes per-user calibration of the system unnecessary.

Finally, the individual residuals are stacked into a singe residual vector $\rho = [\rho_j, \ldots, \rho_j, \rho_b \ldots, \rho_b]^T$. ρ is then used to update the total state.

5.3 Filter Update

To update the total state $\mathbf{x}_{\text{pose}}$, via Eq. (1), we first compute the error state $\delta\mathbf{x}_{\text{pose}}$ by performing a Kalman iteration, thus minimizing the residuals ρ.

We compute the Kalman gain $\mathbf{K}$ with respect to the measurement models $\mathbf{h}_j(\cdot)$ and $\mathbf{h}_b(\cdot)$, evaluated at the current state estimate. We then compute the Jacobian, denoted by $\mathbf{H}$, of Eq. (4) and Eq. (5) with respect to the error state $\delta\mathbf{x}_{\text{pose}}$ and linearized around its expected value $\mathbb{E}\left[\delta\mathbf{x}_{\text{pose}}\right] = \mathbf{0}$. Note that this step in practice is highly involved and involves computation of derivatives for the quaternions in $\mathbf{x}_{\text{pose}} \in SO(3)$, with respect to the error state $\delta\mathbf{x}_{\text{pose}}$. The full Lie-group derivatives are given in the Appendix D.

The *a posteriori* error state is then computed by

$$\delta\mathbf{x}_{\text{pose}} = \mathbf{K}\rho, \tag{6}$$

and the estimated total state is updated such that the expected error state is once again zero $\mathbb{E}[\delta\mathbf{x}] = \mathbf{0}$. This allows us to rewrite Eq. (1):

$$\mathbf{x}^{k+1}_{\text{pose}} = \mathbf{x}^{k}_{\text{pose}} \otimes \delta\mathbf{x}_{\text{pose}}. \tag{7}$$

To make this nonlinear state estimation problem of a discrete-time stochastic system tractable in real-time, we have posed it as an error-state extended Kalman filter (EKF), which computes the state estimates as maximum a posteriori (MAP) estimate. The details of the computation of the fusion step are given in Appendix C. The computed *a posteriori* error state $\delta\mathbf{x}_{\text{pose}}$ is thus only a first order approximation of the true error state. The accuracy of the state estimate can be improved by repeatedly performing an update with a single set of measurements, this is known as an iterated state update (ISEKF) [Gibbs 2011]. Via re-linearization of the measurement equation around the updated state, the IEKF avoids issues with filter convergence, due to accumulated linearization error. The covariance matrix is updated with the standard Kalman Filter equation:

$$\mathbf{P}^{k+1} = (\mathbf{I} - \mathbf{KH})\,\mathbf{P}^{k} \tag{8}$$

We now have attained an estimate of the joint state of the multi-robot human formation including the desired human pose configuration via optimizing $\mathbf{x}_{\text{pose}}$.

5.4 Bone length estimation

The bias, or bone-length states $\mathbf{x}_b$, of our filter remain constant over time, but are unknown a-priori. We use an additional linear Kalman filter with a zero order state propagation model [Gibbs 2011] to estimate the bonelength, given the estimated joint positions $\mathbf{p}_j \in \mathbf{x}_{\text{prior}}$. We only perform the filter update of a bone if both corresponding joints $\mathbf{p}_{ja}$ and $\mathbf{p}_{jb}$ are seen at least by two cameras.

6 CAMERA CONTROL

For accurate human pose estimation we must ensure that the human is always in the field of view of each drone and that each drone in the swarm records the human from a different viewpoint. To achieve this, we build upon the control method of [Naegeli et al. 2017], defining a N-step finite-horizon constrained non-linear optimization problem at time instant k. Note that here we assume known drone and human states, as well as 2D marker positions, given by the filter.

Robot model: To generate correct control inputs, a mathematical model of the drone in form of a non-linear differentiable function $f : \mathbb{R}^{n_x \times n_u} \rightarrow \mathbb{R}^{n_x}$, discretized using a standard forward Euler approach, is needed. The discrete-time state update equation of the drone is

$$\mathbf{x}^{k+1}_{d} = f(\mathbf{x}^{k}_{d}, \mathbf{u}^{k}),$$

where n_x is the dimension of the state $\mathbf{x}_d \in \mathbb{R}^{n_x}$, n_u is the dimension of the input $\mathbf{u} \in \mathbb{R}^{n_u}$ and superindex k denotes the discrete time instant. In our experiments we use a Parrot Bebop2 and include dynamics of the (software) gimbal. This results in $n_x = 11$ and $n_u = 6$, see Appendix B. With this model we define a number of cost terms to constrain the camera motion relative to the user.

6.1 Marker visibility

To ensure that each drone can observe as many of the markers as possible, we ask each drone to keep the bounding box of the detected and labeled marker positions at a desired 2D position on-screen. To the orientation of the human to maximize marker coverage, we control the relative distance to the human via the size of the projected bounding box and the viewing direction of each drone with respect. Via a constant velocity model we then predict the human states $\mathbf{x}_h$ into the future. These include the position $\mathbf{p}_h$ of the center of the bounding box and its orientation. Image space locations are controlled via a quadratic error measure $c_{\text{i}} : \mathbb{R}^{n_x+6} \rightarrow \mathbb{R}_+$ on the residual ϵ_m of the actual and desired look-at vectors:

$$c_{\text{i}}(\mathbf{x}_{\text{pose}}) = \|\epsilon_m\|_2 \quad \text{with} \quad \epsilon_m = \frac{\mathbf{r}^c_{ch}}{\|\mathbf{r}^c_{ch}\|} - \frac{\mathbf{r}^c_{d}}{\|\mathbf{r}^c_{d}\|}, \tag{9}$$

where $\mathbf{r}^c_{ch}$ is the ray from the camera to the human and $\mathbf{r}^c_d = (\mathbf{m}_d, 1) \in \mathbb{R}^3$ is the vector through the desired screen-space position, where pixel coordinates $\mathbf{m}_d$ are computed via the camera intrinsics.

The screen-size of the bounding box is controlled via the quadratic error function $c_{\text{d}} : \mathbb{R}^7 \rightarrow \mathbb{R}_+$ on the residual between the actual σ and the desired σ_d Euclidean distance between the user's position $\mathbf{p}_h$, extracted from $\mathbf{x}_h$, and the camera's $\mathbf{p}_q$:

$$c_{\text{d}}(\mathbf{x}_{\text{pose}}) = \left\| \|\mathbf{p}_h - \mathbf{p}_q\|_2 - \sigma_d \right\|_2 . \tag{10}$$

Similarly, the relative viewing angle per drone is controlled via the quadratic error function $c_a : \mathbb{R}^{n_x+6} \rightarrow \mathbb{R}_+$ on the residual ϵ_a of the camera relative to the orientation of the human:

$$c_a(\mathbf{x}_{\text{pose}}) = \|\epsilon_a\|_2 \quad \text{with} \quad \epsilon_a = \frac{\mathbf{r}_{ch}}{\|\mathbf{r}_{ch}\|} - \frac{\mathbf{a}_d}{\|\mathbf{a}_d\|}, \tag{11}$$

where $\mathbf{r}_{ch}$ is the vector from the center of the camera to the human in global frame, and $\mathbf{a}_d$ is the desired relative viewing orientation, given by

$$\mathbf{a}_d = \left[\sin\theta_d \cos(\psi_d + \psi_h), \quad \sin\theta_d \sin(\psi_d + \psi_h), \quad \cos\theta_d\right]^T,$$

where ψ_h is the current orientation of the human and θ_d and ψ_d are the desired viewing angles, both specified by the user and different for each drone to observe the human from different view points.

6.2 Trajectory optimization

For a given drone, and in a slight abuse of notation, we denote by $\mathbf{x} = [{\mathbf{x}_d}^0, \ldots, {\mathbf{x}_d}^N]$ and $\mathbf{u} = [\mathbf{u}^0, \ldots, \mathbf{u}^{N-1}]$ the computed trajectory and inputs, where $\mathbf{x}_d^0$ and $\mathbf{u}^0$ are the initial states and drone inputs.

We take a linear combination of the error measures for image location Eq. (9), size Eq. (10) and viewing angle Eq. (11) to define a stage cost for trajectory optimization:

$$J_k = a_l c_l(\mathbf{x}^k_{\text{pose}}) + a_d c_d(\mathbf{x}^k_{\text{pose}}) + a_a c_a(\mathbf{x}^k_{\text{pose}}) \quad , \tag{12}$$

where the scalar weight parameters $a_l, a_d, a_a > 0$ can be set interactively to control the (relative) importance of the different terms. The trajectory and control inputs of the drone at each time step are computed via the solution of the following N-step finite horizon constrained nonlinear optimization problem at time instant t.

$$\min_{\mathbf{x},\mathbf{u}} \quad \sum_{k=0}^{N-1} \left(J_k + {\mathbf{u}^k}^T \mathbf{R} \mathbf{u}^k\right) + a_N J_N \tag{13}$$

$$\begin{aligned}
\text{subject to} \quad & \mathbf{x}^0 = \hat{\mathbf{x}}_d(t) && \text{(Initial state)} \\
& \mathbf{x}_d^{k+1} = f(\mathbf{x}_d^k, \mathbf{u}^k), && \text{(Dynamics)} \\
& \mathbf{x}_d^k \in \mathcal{X}, && \text{(State constraints)} \\
& \mathbf{u}^k \in \mathcal{U}, && \text{(Input constraints)} \\
& && \forall k \in \{0, \ldots, N-1\} \\
& {\mathbf{x}_d}^N \in \mathcal{X}, && \text{(State constraints)}
\end{aligned}$$

where $\mathbf{R} \in \mathbb{S}_+^{n_u}$ is a positive definite penalty matrix to avoid excessive use of the control inputs. The scalar $a_N > 0$ is a weight parameter used to weight a terminal cost J_N on the final stage. This is common in finite-horizon schemes to mimic long horizons, approximating the infinite horizon solution. The vector $\hat{\mathbf{x}}_d(t)$ denotes the estimated value of the current state $\mathbf{x}_d$. Finally, the sets $\mathcal{X}$ and $\mathcal{U}$ denote the sets of feasible states and inputs for the drone, respectively. These can be derived from physical limits of the environment and by the internal constraints of the flying camera hardware, e.g. bounds on vertical and horizontal velocities as well as on roll and pitch angles. We obtained the limits from the documentation of the Parrot SDK [Par 2015]. While each quadrotor model has different values of these bounds, in general such bounds exist and can be assumed to be known for a particular model.

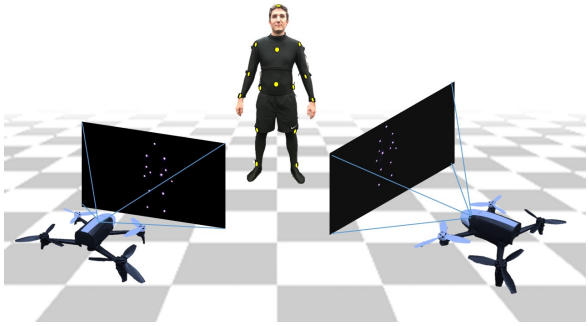

Fig. 4. Our system consists of two drones observing 13 active LED markers worn on the users body. The controller (see Sec. 6.2) computes drone inputs to keep as many markers as possible visible. From the 2D detections observed by the MAVs the human pose is estimated in real-time.

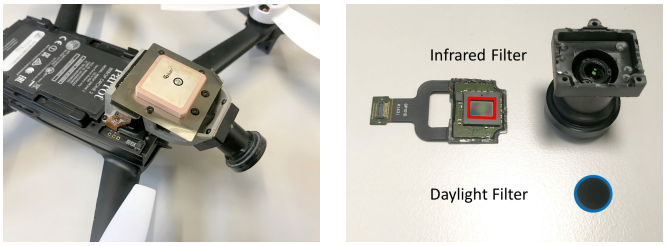

Fig. 5. The active marker detection scheme only requires little modification to the Parrot Bebop's hardware and is hence cheap. First, the lens-mount has to be removed using a heat-gun (left). Then, the infrared filter (red) can be removed and the daylight filter (blue) can be fitted to the lens mound. In the last step, the camera has to be reassembled and re-calibrated.

Additional constraints for avoiding collisions between the drones and between each drone and the tracked human could also be added, analogously to [Naegeli et al. 2017].

The drone is actuated using the optimal inputs from the first step $\mathbf{u}_0$. Importantly, a new trajectory is recomputed at each time-step, taking updated sensor data into consideration.

7 IMPLEMENTATION

Our experiments are conducted on a standard desktop PC (Quadcore Intel i7 CPU@3.5 GHz). The subjects are tracked directly by the drones via a custom active LED marker scheme. *No* external motion capture system was used. We implement the recursive estimation algorithm using Matlab.

Quadrotor hardware: We use Parrot Bebop2 quadrotors in all our experiments with an integrated electronic gimbal the camera has been modified to remove daylight illumination but record IR illumination (cf. Fig. 5). All communication between the drones and the host PC is handled via ROS [Quigley et al. 2009] and we directly send the control inputs from the first time-step $\mathbf{u}_0$ *without* an additional feedback controller for trajectory tracking.

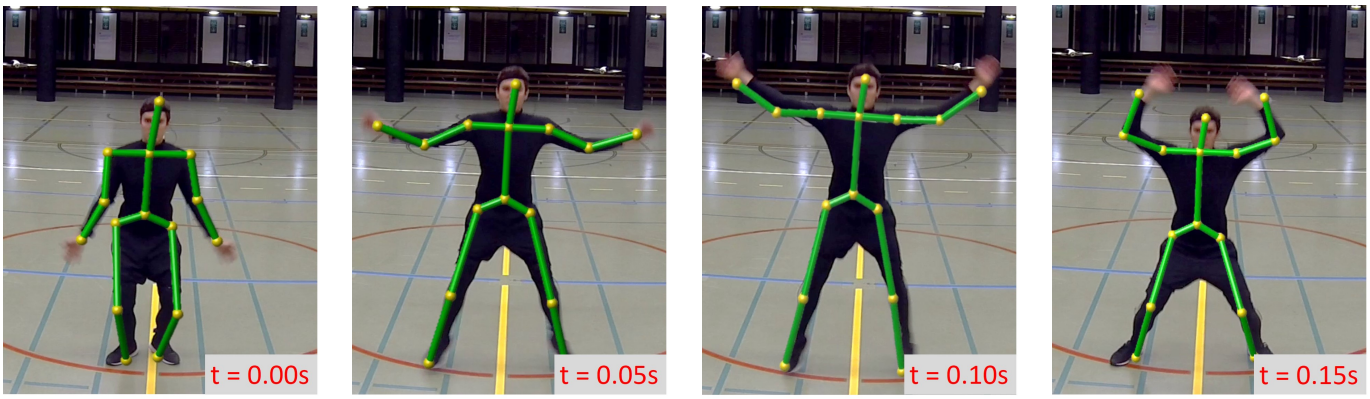

Fig. 6. Experiment 1: A subject performing jumping jacks. The sequence is captured with a camera and the reconstruction of our system is overlaid. The speed limitation of motions we can track is limited by the framerate of the Parrot Bebop 2 live image stream and therefore by the marker tracking. The estimated joint positions are indicated in yellow, the estimated skeleton is marked in green.

7.1 Active markers

Our method takes 2D joint detections as input. While body-part detection in monocular images is possible [Chen and Yuille 2014; Newell et al. 2016; Tompson et al. 2014; Toshev and Szegedy 2014; Wei et al. 2016] we leave full integration of such methods for future work. Instead we utilize body-worn markers allowing for simple detection and unique labeling of joints. Our setup consists of 13 active IR-LED markers attached to a morphsuit (see Fig. 4).

A band-pass filter was added to the camera lenses, removing daylight but letting IR illumination pass (Fig. 5). This allows for outdoor use of the system. Each marker is composed of two LEDs, one illuminated permanently and the other displaying a unique temporal pattern. Markers are segmented from the background via simple image processing operations. The temporal pattern creates varying image intensities which are converted into a bit-stream which is used to uniquely identify markers and to track them over time (see Appendix E for details.)

8 EXPERIMENTS

Our method enables motion capture in scenarios that are difficult or entirely infeasible with traditional techniques. Hence, direct quantitative evaluation of accuracy is difficult. Furthermore, the accuracy is affected by the placement of markers on the body and processing of the resulting images as well as camera calibration. We demonstrate the feasibility and robustness of our proof-of-concept implementation in five experimental evaluations where we continuously reconstruct full body pose of a subject during fast movements, moving through large-scale scenes, in difficult to reach locations, indoors and outdoors.

Fast Motion (Ex 1): In a first experiment, a participant performs jumping jacks. The sequence in Fig. 6 shows one half-cycle of a jumping jack (duration: 0.15 seconds). The maximal joint velocity is limited by the camera sampling rate (30Hz in our experiment). The reconstructed joint positions are indicated in yellow, the skeleton is projected into the image and rendered in green. Please also refer to the accompanying video.

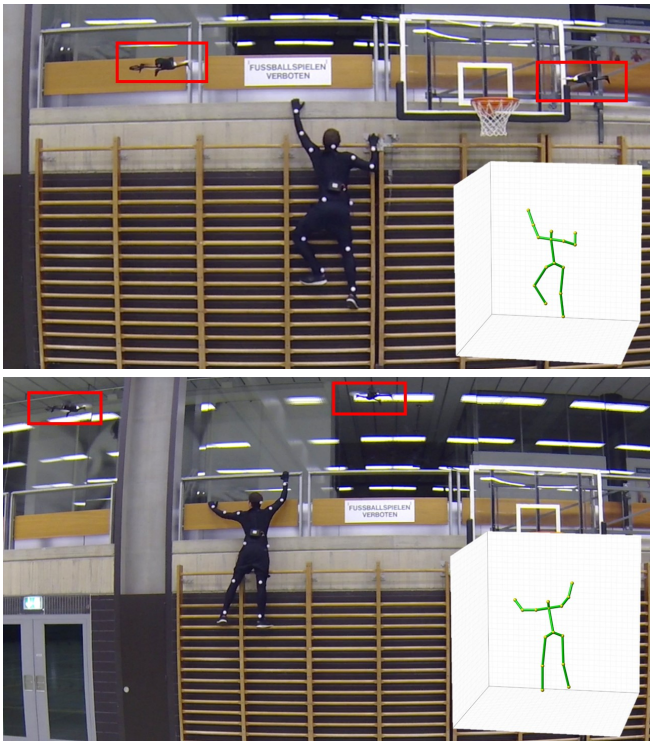

Fig. 7. Experiment 2: A subject is climbing up a wall. The drones follow the subject over different elevations and locations. The markers are indicated in yellow, the estimated skeleton in green. The drones are indicated in red.)

Climbing (Ex 2): To demonstrate the location independence we show results form our system being deployed in a difficult to reach location. A subject climbs up, across, and down a climbing wall,

while the drones track the position. Extracts from this sequence can be seen in Fig. 7. The drone positions (indicated in red) are optimized to see all markers and hence automatically follow the subject, adjusting their height above ground without external control.

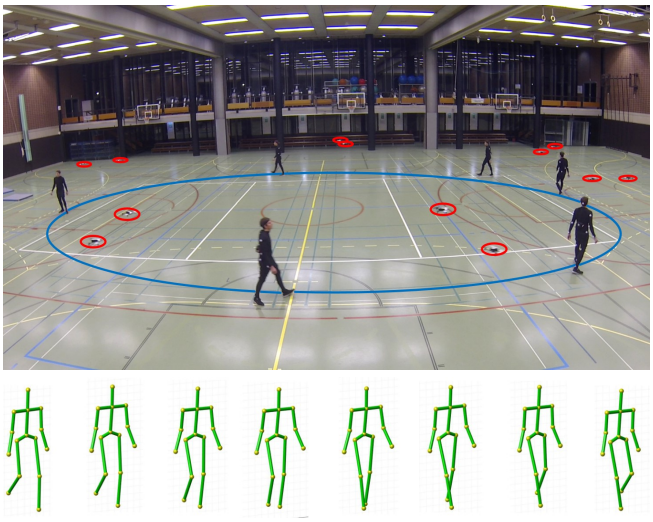

Fig. 8. Experiment 3: Top: Subject walking over a long distance and time period in circles - indicated in blue - with a walking speed around $1.5\,\frac{m}{s}$ (top). The drones follow the subject and always position themselves to optimally observe the markers, mounted on the back of the subject. Bottom: reconstructed gait cycle recorded during the experiment.

Long trajectory (Ex 3): Long-range trajectories are a particularly challenging scenario for traditional motion capture approaches. To demonstrate the environment independence of our approach, we ask a participant to walk in large circles in an area that exceeds typical motion capture spaces significantly. Fig. 8, top shows different snapshots from the sequence, drones highlighted in red. Fig. 8, bottom illustrates the corresponding estimated skeletal configurations. Note that in this experiment the system tracks the user over time period of 3 min and over a trajectory length of 170 meters. We observed an absolute position drift of about 2m, caused by the integrating nature of the optical flow estimates. In Fig. 9 we show the cumulated joint positions (red), relative to the person's center of mass over the long walking sequence.

Ground-truth comparison (Ex 4): To compute the expected accuracy of our method we performed an experiment with a motion capture system. In particular, we compare the estimated distance between the hand of the subject and one of the drone cameras with that obtained from the vicon based ground truth. In Fig. 10 we show the distance from the left hand to the first drone camera. Over a 30 second sequence We obtained a standard deviation of 2.2cm.

Outdoor Test (Ex 5): We assess the environment independence of the proposed approach via an additional outdoor experiment. Fig. 11 shows a motion sequence with the live reconstruction of the skeleton (green) in overlay. The drones are highlighted in red for better visibility.

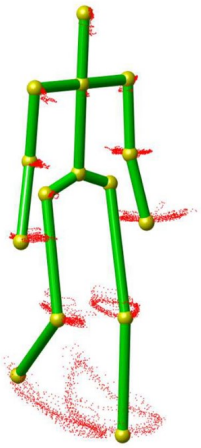

Fig. 9. The joint positions (yellow) of Experiment 3 (see Fig. 8) are plotted over time (red) with respect to the center of mass. The noise distribution is 2cm with respect to the mean joint-trajectory.

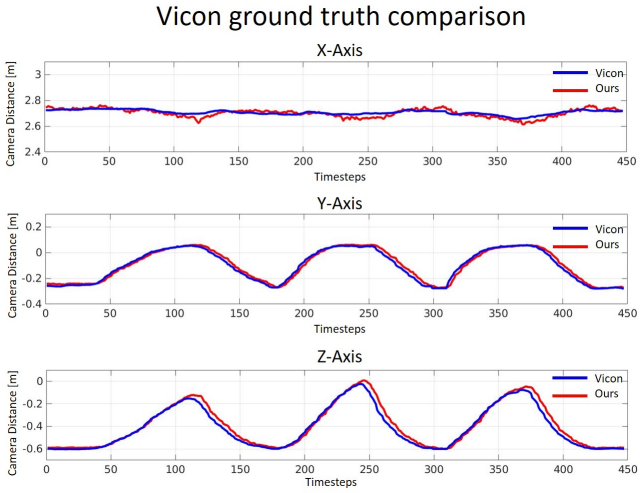

Fig. 10. Experiment 4: Ground truth comparison between the hands and the camera while walking. In the plot we show the relative distance (x top, y middle and z bottom) between the left hand and the first camera as a representative result. The ground truth is blue, our estimate is red. The standard deviation is 2.2cm.

Computational Complexity. In principle, the our method could track multiple sets of markers (multiple subjects). However, the complexity of the algorithm grows cubically with the number of measurements and quadratically with the number of states. In our proof of concept implementation the number of states is $(6n + 3m)$ and the number of measurements is $(2m \times n)$. Yet, the current image frame-rate (30Hz) is well below the filter update speed (100Hz), providing enough margin to increase the number of drones or subjects tracked.

Fig. 11. Experiment 5: Our system works indoors and outdoors. Here we show extract from a long walking trajectory in varied terrain. Best viewed in the accompanying video.

9 LIMITATIONS AND CONCLUSIONS

In this paper we proposed a marker based real-time method for the infrastructure-free estimation of articulated human motion. The approach leverages a swarm of camera-equipped flying robots (MAVs) and *jointly* optimizes the swarm's and skeletal states, including 3D joint positions and a set of bones, in real-time. The problem is phrased as a non-linear recursive filtering estimation, namely IESKF, allowing us to naturally link state estimates and measurements over time. Furthermore, a robo-centric formulation minimizes accumulation of error due to linearization around the last state and uncertainty about the global transform. The method provides robust long-term predictions of the global pose of the multi-robot swarm and the human skeletal configuration.

In this paper, we demonstrated the proposed method in a number of challenging settings where traditional multi-view methods are not applicable. In our proof of concept system, we use a minimum working example of two flying drones. Although the markers can be put arbitrary on the subject, the current marker locations on the back of the subject were chosen for visibility with two drones. If the movement of the person is very fast, visibility could be lost with the current solution. This can be solved by employing more drones, e.g., two additional ones in front of the person.

Our work lies the foundation for a host of exciting avenues for future work. Foremost we currently rely on active LED-markers to detect 2D joint locations. The framework would naturally admit 2D detections stemming from a deep-learning method that extracts these joint detections from natural images alone (e.g., [Toshev and Szegedy 2014; Wei et al. 2016]) or are even directly from videos (e.g., [Song et al. 2017]). However, note that our method requires accurate tracking of the human and makes small-motion assumptions, hence integration of a deep-learning approach into our pipeline would have to address several interesting challenges and would require strict real-time performance. Furthermore, environment features could be automatically extracted and tracked to enable even more accurate localization of the MAVs and tracking of the human. Another interesting aspect is to extend our method to work with learning-based approaches that directly predict 3D-pose from images (e.g., [Mehta et al. 2017]). This would require changes to the formulation of the skeletal estimation algorithm but could be a very fruitful direction for future work. Another interesting challenge is to incorporate our method into a pipeline that capture dense surface deformation via model-fitting or related approaches (e.g., [Rhodin et al. 2015; Robertini et al. 2016]). Finally, we are keen to explore applications of our method in graphics, AR/VR and bio-mechanics.

REFERENCES

2015. Parrot SDK. (2015). http://developer.parrot.com/.

Javier Alonso-Mora, Eduardo Montijano, Tobias Nägeli, Otmar Hilliges, Mac Schwager, and Daniela Rus. 2018. Distributed multi-robot formation control in dynamic environments. *Autonomous Robots* (July 2018).

Luca Ballan, Aparna Taneja, Jürgen Gall, Luc Van Gool, and Marc Pollefeys. 2012. Motion capture of hands in action using discriminative salient points. *Computer Vision–ECCV 2012* (2012), 640–653.

Meysam Basiri, Felix Schill, Dario Floreano, and Pedro Lima. 2013. Audio-based relative positioning system for multiple micro air vehicle systems. In *Robotics: Science and Systems RSS2013*.

Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Gehler, Javier Romero, and Michael J Black. 2016. Keep it SMPL: Automatic estimation of 3D human pose and shape from a single image. In *European Conference on Computer Vision*. Springer, 561–578.

Christoph Bregler and Jitendra Malik. 1998. Tracking people with twists and exponential maps. In *Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on*. IEEE, 8–15.

Pierre-Jean Bristeau, François Callou, David Vissière, and Nicolas Petit. 2011. The Navigation and Control technology inside the AR.Drone micro UAV. *IFAC Proceedings Volumes* 44, 1 (2011), 1477 – 1484. https://doi.org/10.3182/20110828-6-IT-1002.02327 18th IFAC World Congress.

J A Castellanos, Jose Neira, and Juan Domingo Tardos. 2004. Limits to the consistency of EKF-based SLAM. (2004). https://doi.org/10.1109/TAC.2000.880989

Xianjie Chen and Alan L Yuille. 2014. Articulated pose estimation by a graphical model with image dependent pairwise relations. In *NIPS*. 1736–1744.

Edilson de Aguiar, Carsten Stoll, Christian Theobalt, Naveed Ahmed, Hans-Peter Seidel, and Sebastian Thrun. 2008. Performance Capture from Sparse Multi-view Video. In *ACM SIGGRAPH 2008 Papers (SIGGRAPH '08)*. ACM, New York, NY, USA, Article 98, 10 pages. https://doi.org/10.1145/1399504.1360697

N. de Palézieux, T. Nägeli, and O. Hilliges. 2016. Duo-VIO: Fast, light-weight, stereo inertial odometry. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2237–2242. https://doi.org/10.1109/IROS.2016.7759350

Mingsong Dou, Sameh Khamis, Yury Degtyarev, Philip Davidson, Sean Ryan Fanello, Adarsh Kowdle, Sergio Orts Escolano, Christoph Rhemann, David Kim, Jonathan Taylor, Pushmeet Kohli, Vladimir Tankovich, and Shahram Izadi. 2016. Fusion4D: Real-time Performance Capture of Challenging Scenes. *ACM Trans. Graph.* 35, 4, Article 114 (July 2016), 13 pages. https://doi.org/10.1145/2897824.2925969

Ahmed Elhayek, Edilson de Aguiar, Arjun Jain, J Thompson, Leonid Pishchulin, Mykhaylo Andriluka, Christoph Bregler, Bernt Schiele, and Christian Theobalt. 2017. MARCOnI—ConvNet-Based MARker-Less Motion Capture in Outdoor and Indoor Scenes. *IEEE transactions on pattern analysis and machine intelligence* 39, 3 (2017), 501–514.

B. Friedland. 1969. Treatment of bias in recursive filtering. *IEEE Trans. Automat. Control* 14, 4 (August 1969), 359–367. https://doi.org/10.1109/TAC.1969.1099223

Varun Ganapathi, Christian Plagemann, Daphne Koller, and Sebastian Thrun. 2012. Real-time human pose tracking from range data. In *European conference on computer vision*. Springer, 738–751.

Christoph Gebhardt, Benjamin Hepp, Tobias Nägeli, Stefan Stevšić, and Otmar Hilliges. 2016. Airways: Optimization-Based Planning of Quadrotor Trajectories According to High-Level User Goals. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 2508–2519. https://doi.org/10.1145/2858036.2858353

Christoph Gebhardt, Stefan Stevsic, and Otmar Hilliges. 2018. Optimizing for Aesthetically Pleasing Quadrotor Camera Motion. *ACM Trans. Graph.* 37, 4, Article 90 (2018), 11 pages.

Bruce P. Gibbs. 2011. *Advanced Kalman filtering, least-squares and modeling.* John Wiley & Sons.

Richard Hartley and Andrew Zisserman. 2003. *Multiple View Geometry in Computer Vision* (2 ed.). Cambridge University Press, New York, NY, USA.

Chien-Shu Hsieh. 2000. Robust two-stage Kalman filters for systems with unknown inputs. *IEEE Trans. Automat. Control* 45, 12 (2000), 2374–2378.

Chong Huang, Zhenyu Yang, Yan Kong, Peng Chen, Xin Yang, and Kwang-Ting Tim Cheng. 2018. Through-the-Lens Drone Filming. (2018).

Niels Joubert, Mike Roberts, Anh Truong, Floraine Berthouzoz, and Pat Hanrahan. 2015. An Interactive Tool for Designing Quadrotor Camera Shots. *ACM Trans. Graph.* 34, 6, Article 238, 11 pages. https://doi.org/10.1145/2816795.2818106

Ern J Lefferts, F Landis Markley, and Malcolm D Shuster. 1982. Kalman filtering for spacecraft attitude estimation. *Journal of Guidance, Control, and Dynamics* (1982).

Rui Li, Minjian Pang, Cong Zhao, Guyue Zhou, and Lu Fang. 2016. Monocular long-term target following on uavs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 29–37.

Hyon Lim and Sudipta Sinha. 2015. Monocular Localization of a moving person onboard a Quadrotor MAV. https://www.microsoft.com/en-us/research/publication/trajrecon/

Huajun Liu, Xiaolin Wei, Jinxiang Chai, Inwoo Ha, and Taehyun Rhee. 2011. Realtime human motion control with a small number of inertial sensors. In *Symposium on Interactive 3D Graphics and Games*. ACM, 133–140.

S. Lupashin, A. Schollig, M. Hehn, and R. D'Andrea. 2011. The Flying Machine Arena as of 2010. In *IEEE ICRA '11*. 2970–2971. https://doi.org/10.1109/ICRA.2011.5980308

Dushyant Mehta, Srinath Sridhar, Oleksandr Sotnychenko, Helge Rhodin, Mohammad Shafiei, Hans-Peter Seidel, Weipeng Xu, Dan Casas, and Christian Theobalt. 2017. VNect: Real-time 3D Human Pose Estimation with a Single RGB Camera. *ACM Transactions on Graphics* 36, 4, 14. https://doi.org/10.1145/3072959.3073596

Nathan Michael, D. Mellinger, Q. Lindsey, and V. Kumar. 2010. The GRASP Multiple Micro-UAV Testbed. *Robotics Automation Magazine, IEEE* 17, 3 (2010), 56–65. https://doi.org/10.1109/MRA.2010.937855

Thomas B Moeslund, Adrian Hilton, and Volker Krüger. 2006. A survey of advances in vision-based human motion capture and analysis. *Computer vision and image understanding* 104, 2 (2006), 90–126.

T. Naegeli, J. Alonso-Mora, A. Domahidi, D. Rus, and O. Hilliges. 2017. Real-time Motion Planning for Aerial Videography with Dynamic Obstacle Avoidance and Viewpoint Optimization. *IEEE Robotics and Automation Letters* 2, 3 (2017), 1696–1703. https://doi.org/10.1109/LRA.2017.2665693

Tobias Nägeli, Christian Conte, Alexander Domahidi, Manfred Morari, and Otmar Hilliges. 2014. Environment-independent formation flight for micro aerial vehicles. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, 1141–1146.

Tobias Nägeli, Lukas Meier, Alexander Domahidi, Javier Alonso-Mora, and Otmar Hilliges. 2017. Real-time Planning for Automated Multi-view Drone Cinematography. *ACM Trans. Graph.* 36, 4, Article 132 (July 2017), 10 pages. https://doi.org/10.1145/3072959.3073712

Richard A Newcombe, Dieter Fox, and Steven M Seitz. 2015. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 343–352.

Alejandro Newell, Kaiyu Yang, and Jia Deng. 2016. Stacked hourglass networks for human pose estimation. In *ECCV*. 483–499.

Iasonas Oikonomidis, Nikolaos Kyriazis, and Antonis A Argyros. 2012. Tracking the articulated motion of two strongly interacting hands. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 1862–1869.

Gerard Pons-Moll, Javier Romero, Naureen Mahmood, and Michael J. Black. 2015. Dyna: A Model of Dynamic Human Shape in Motion. *ACM Trans. Graph.* 34, 4, Article 120 (July 2015), 14 pages. https://doi.org/10.1145/2766993

Jim Pugh and Alcherio Martinoli. 2006. Relative localization and communication module for small-scale multi-robot systems. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*. IEEE, 188–193.

Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. 2009. ROS: an open-source Robot Operating System. In *IEEE ICRA Workshop on Open Source Software*.

Helge Rhodin, Nadia Robertini, Christian Richardt, Hans-Peter Seidel, and Christian Theobalt. 2015. A versatile scene model with differentiable visibility applied to generative pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision*. 765–773.

Nadia Robertini, Dan Casas, Helge Rhodin, Hans-Peter Seidel, and Christian Theobalt. 2016. Model-based Outdoor Performance Capture. In *Proceedings of the 2016 International Conference on 3D Vision (3DV 2016)*. http://gvv.mpi-inf.mpg.de/projects/OutdoorPerfcap/

Mike Roberts and Pat Hanrahan. 2016. Generating Dynamically Feasible Trajectories for Quadrotor Cameras. *ACM Trans. Graph.* 35, 4, Article 61 (July 2016), 11 pages. https://doi.org/10.1145/2897824.2925980

Daniel Roetenberg, Henk Luinge, and Per Slycke. 2007. Moven: Full 6dof human motion tracking using miniature inertial sensors. *Xsen Technologies, December* 2, 3 (2007), 8.

S. I. Roumeliotis, G. S. Sukhatme, and G. A. Bekey. 1999. Circumventing dynamic modeling: evaluation of the error-state Kalman filter applied to mobile robot localization. In *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, Vol. 2. 1656–1663 vol.2. https://doi.org/10.1109/ROBOT.1999.772597

Loren Schwarz, Diana Mateus, and Nassir Navab. 2009. Discriminative human full-body pose estimation from wearable inertial sensor data. *Modelling the Physiological Human* (2009), 159–172.

Jamie Shotton, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. 2013. Real-time human pose recognition in parts from single depth images. *Commun. ACM* 56, 1 (2013), 116–124.

Jie Song, Limin Wang, Luc Van Gool, and Otmar Hilliges. 2017. Thin-Slicing Network: A Deep Structured Model for Pose Estimation in Videos. *arXiv preprint arXiv:1703.10898* (2017).

Jonathan Starck and Adrian Hilton. 2003. Model-based multiple view reconstruction of people. In *null*. IEEE, 915.

Carsten Stoll, Nils Hasler, Juergen Gall, Hans-Peter Seidel, and Christian Theobalt. 2011. Fast articulated motion tracking using a sums of gaussians body model. In *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 951–958.

Jochen Tautges, Arno Zinke, Björn Krüger, Jan Baumann, Andreas Weber, Thomas Helten, Meinard Müller, Hans-Peter Seidel, and Bernd Eberhardt. 2011. Motion reconstruction using sparse accelerometer data. *ACM Transactions on Graphics (TOG)* 30, 3 (2011), 18.

Jonathan Taylor, Jamie Shotton, Toby Sharp, and Andrew Fitzgibbon. 2012. The vitruvian manifold: Inferring dense correspondences for one-shot human pose estimation. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 103–110.

Bugra Tekin, Pablo Márquez-Neila, Mathieu Salzmann, and Pascal Fua. 2016. Fusing 2D Uncertainty and 3D Cues for Monocular Body Pose Estimation. *arXiv preprint arXiv:1611.05708* (2016).

Jonathan J Tompson, Arjun Jain, Yann LeCun, and Christoph Bregler. 2014. Joint training of a convolutional network and a graphical model for human pose estimation. In *NIPS*. 1799–1807.

Alexander Toshev and Christian Szegedy. 2014. Deeppose: Human pose estimation via deep neural networks. In *CVPR*. 1653–1660.

T. von Marcard, B. Rosenhahn, M. J. Black, and G. Pons-Moll. 2017. Sparse Inertial Poser: Automatic 3D Human Pose Estimation from Sparse IMUs. *Comput. Graph. Forum* 36, 2 (may 2017), 349–360. https://doi.org/10.1111/cgf.13131

Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. 2016. Convolutional pose machines. In *CVPR*. 4724–4732.

Lan Xu, Yebin Liu, Wei Cheng, Kaiwen Guo, Guyue Zhou, Qionghai Dai, and Lu Fang. 2017. FlyCap: Markerless motion capture using multiple autonomous flying cameras. *IEEE transactions on visualization and computer graphics* (2017).

Xiaowei Zhou, Menglong Zhu, Spyridon Leonardos, Konstantinos G Derpanis, and Kostas Daniilidis. 2016. Sparseness meets deepness: 3D human pose estimation from monocular video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4966–4975.

Michael Zollhöfer, Matthias Nießner, Shahram Izadi, Christoph Rehmann, Christopher Zach, Matthew Fisher, Chenglei Wu, Andrew Fitzgibbon, Charles Loop, Christian Theobalt, et al. 2014. Real-time non-rigid reconstruction using an RGB-D camera. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 156.

A NOTATION

The following coordinate frames are used throughout this paper $\mathbf{W}$ – the inertial world frame; $\mathbf{O}$ – the origin frame; A_j – anchor frames; $\mathbf{C}$ – the camera frame; $\mathbf{I}$ – the IMU frame.

We follow the standard notation proposed in literature. Translation vectors between two frames A and B, expressed in frame A, are denoted by $\mathbf{t}_{AB}$. Rotation matrices performing rotations from frame A to frame B are denoted by $\mathbf{R}_{BA} = \mathbf{R}(\bar{q}_{BA}) \in SO(3)$, where $\bar{q}_{BA}$ is the corresponding quaternion. We adhere to the JPL quaternion definition and denote a quaternion by $\bar{q} = \left[q_x\mathbf{i} + q_y\mathbf{j} + q_z\mathbf{k} + q_w\right] = [\mathbf{q}, q_w]^T$. Quaternion multiplication is denoted by $\otimes$.

Expected or estimated values of a variable x are denoted by $\mathbb{E}[x] = \hat{x}$, errors are written as δx. Orientation errors are described

in $so(3)$, the tangent space of $SO(3)$, and are written as $\delta\theta$. Measurements of a quantity x affected by white Gaussian noise are written as $z = x + \nu$ with $\nu \sim \mathcal{N}(\sigma)$. Due to the many degrees-of-freedom of the problem discussed in this paper, the resulting notation is rather verbose.

In various parts of the algorithm, state estimates are changed from one value to another, where the new value is often a function of the previous state estimate. We denote the update of a variable with an arrow $\leftarrow$. For example, incrementing a variable x by 1 is written $x \leftarrow x + 1$.

B QUADROTOR DYNAMICS MODEL

The state of the quadrotor is given by its position $\mathbf{p}_q \in \mathbb{R}^3$, its velocity $\dot{\mathbf{p}}_q \in \mathbb{R}^3$ and its orientation in $SO(3)$, i.e. roll Φ_q, pitch Θ_q and yaw ψ_q. The camera is attached to the robot via a pan-tilt gimbal (in case of the Bebop this is a software gimbal). The state of the camera is given by its position $\mathbf{p}_c$ (rigid body transformation from $\mathbf{p}_q$), the velocity $\dot{\mathbf{p}}_q$ and the gimbal states θ_g, ψ_g. For ease of explanation we assume $\mathbf{p}_q - \mathbf{p}_c$. We denote the state of the system, consisting of quadrotor and gimbal, by

$$\mathbf{x}_d = [\mathbf{p}_q, \dot{\mathbf{p}}_q, \Phi_q, \Theta_q, \psi_q, \theta_g, \psi_g] \in \mathbb{R}^{11}. \tag{14}$$

Following the Parrot Bebop 2 SDK, the control inputs to the system are given by the vector

$$\mathbf{u} = [v_z, \phi_q, \theta_q, \omega_{\psi_q}, \omega_{\theta_g}, \omega_{\psi_g}] \in \mathbb{R}^6, \tag{15}$$

where v_z is the velocity of the quadrotor in the body-z axis, ϕ_q and θ_q are the desired roll and pitch angles of the quadrotor, respectively, ω_{ψ_q} is the angular speed around the body-z axis and $\omega_{\theta_g}, \omega_{\psi_g}$ are the pitch and yaw rates of the camera gimbal. The horizontal velocities are not directly controlled.

We employ a first order low-pass Euler approximation of the quadrotor dynamics, as follows. The translational dynamics are then given by $\dot{\mathbf{p}}_q = [\dot{\mathbf{p}}_{qx,y}, v_z]$ and $\ddot{\mathbf{p}}_q = [\ddot{\mathbf{p}}_{qx,y}, 0]$, with

$$\ddot{\mathbf{p}}_{qx,y} = \mathbf{R}_{\psi_q}(\psi_q) \begin{bmatrix} -tan(\Phi_q) \\ tan(\Theta_q) \end{bmatrix} g - C\dot{\mathbf{p}}_{qx,y}, \tag{16}$$

where $g = 9.81\frac{m}{s^2}$ is the earth's gravity, $\mathbf{R}_{\psi_q}(\psi_q) \in SO(2)$ is the rotation matrix only containing the yaw rotation of the quadrotor and C is the drag coefficient at low speeds. The rotational dynamics of the quadrotor are

$$\dot{\Phi}_q = \tau_a(\phi_q - \Phi_q), \quad \dot{\Theta}_q = \tau_a(\theta_q - \Theta_q) \text{ and } \dot{\psi}_q = \omega_{\psi_q}, \tag{17}$$

and the gimbal pitch rate is given by $\dot{\theta}_g = \omega_{\theta_g}$.

C ERROR STATE FILTERING

Rotation quaternions have a unit norm constraint $\|\bar{q}\| = 1$ and thus have only three degrees of freedom, like any other orientation parameterization. Due to the fact that a quaternion uses 4 dimensions to describe 3 degrees of freedom, a quaternion's covariance matrix is singular. This issue is avoided with the error state or indirect Extended Kalman Filter formulation [Lefferts et al. 1982].

With this formulation, rather than estimating total states, as is the case for the direct Kalman filter, errors are estimated. Thus, we differentiate between the *total state* $\hat{\mathbf{x}}$, and the *error state* $\delta\mathbf{x}$. Usually, the error between two quantities is defined as the arithmetic difference between the two, which is how we define the error in estimated values which are linear, such as positions:

$$\delta\mathbf{x} := \mathbf{x} - \hat{\mathbf{x}}$$

For orientation errors however, the arithmetic difference is not suitable. We define the error of an orientation using an error quaternion $\delta\bar{q}$, a small rotation between the estimated and true orientation. This error is multiplicative, rather than additive:

$$\bar{q}_{AB} = \delta\bar{q}_{A\hat{A}} \otimes \bar{q}_{\hat{A}B} \quad \Longleftrightarrow \quad \delta\bar{q}_{A\hat{A}} = \bar{q}_{AB} \otimes \left(\bar{q}_{\hat{A}B}\right)^{-1}$$

The error quaternion $\delta\bar{q}_{A\hat{A}}$ can be assumed to be small and the small angle approximation can be made:

$$\delta\bar{q}_{A\hat{A}} \approx \begin{bmatrix} \frac{1}{2}\delta\boldsymbol{\theta} \\ 1 \end{bmatrix}$$

Using $\delta\boldsymbol{\theta}$ to represent orientations in the Kalman filter reduces their dimensionality to 3. This is both computationally advantageous and circumvents the issues with a 4×4 orientation covariance matrix.

In the error state EKF, the error state $\delta\mathbf{x}$ is the quantity being estimated and the covariance matrix $\mathbf{P}$ describes the uncertainty of $\delta\mathbf{x}$. The total state $\hat{\mathbf{x}}$ is always updated such that the expected value of the error state $\mathbb{E}[\delta\mathbf{x}] = \mathbf{0}$. In other words, the total state $\hat{\mathbf{x}}$ always represents the best estimate of $\mathbf{x}$.

D LIE GROUP DERIVATIVES

Consider the rotation matrix $\mathbf{R} \in SO(3)$ and the vector $\mathbf{x} \in \mathbb{R}^3$. Let

$$\mathbf{y} = \mathbf{R}\mathbf{x} \qquad\qquad \mathbf{y}' = \mathbf{R}^T\mathbf{x}$$

The differentiation by $\mathbf{x}$ is straightforward:

$$\frac{\partial\mathbf{y}}{\partial\mathbf{x}} = \mathbf{R} \qquad\qquad \frac{\partial\mathbf{y}'}{\partial\mathbf{x}} = \mathbf{R}^T$$

The differentiation with respect to the rotation parameters that define $\mathbf{R}$, which we simply denote by $\frac{\partial}{\partial\mathbf{R}}$, is:

$$\frac{\partial\mathbf{y}}{\partial\mathbf{R}} = -\lfloor\mathbf{y}\rfloor_\times \qquad\qquad \frac{\partial\mathbf{y}'}{\partial\mathbf{R}} = -\mathbf{R}^T\lfloor\mathbf{x}\rfloor_\times, \tag{18}$$

where $\lfloor\mathbf{w}\rfloor_\times$ is the skew symmetric matrix of a three dimensional vector $\mathbf{w}$ and defined as

$$\lfloor\mathbf{w}\rfloor_\times = \begin{bmatrix} 0 & -w_3 & w_2 \\ w_3 & 0 & -w_1 \\ -w_2 & w_1 & 0 \end{bmatrix}.$$

E ACTIVE LED MARKERS

For reproducibility we detail how we extract marker IDs from the drones onboard camera streams.

Marker labelling. To produce labeled measurements of marker positions in the image, bright blobs corresponding to markers are segmented from the background. Each marker blinks with a distinct bit pattern and when tracked over time the pixel intensities may be converted into a bitstream indicating the on- or off-state of the modulated LED. The extracted bit pattern allows for unique identification of each marker.

Marker detection. Intensity blobs corresponding to our marker candidates are segmented from the background by applying an intensity threshold on input frame I_i. A morphological opening operation is used for noise removal and the remaining connected components are marker candidates C_i. Marker candidates are tracked via a KLT tracker and associated with the newly detected marker candidates.

Bitstream conversion. Each frame I_i provides a sample s_i^j of marker M_j's current signal bit state $b_{cur}^j \in \{0, 1, 0.5\}$, where $b_{cur}^j = 0.5$ denotes a corrupted signal.

Since apparent marker intensity depends on the current state of the blinking LED and extraneous influences, dynamic thresholding is used to classify the state as on (logical 1) or off (logical 0). This marker-specific threshold is computed as the moving average of a marker's intensities over a sample size of N_{window} frames.

Note that our system clocks are not synchronized and hence the time at which the LEDs state switches $t_{transition}$ has to be approximated by a transition window $[t_s, t_e]$. This is done by finding the frames I_i and I_{i+1} where the sample bits of multiple markers change their value, i.e. find i such that $|\{j \mid s_i^j \neq s_{i+1}^j\}| > 3$. Because the signal pattern frequency f is known, it can be assumed that $t_{transition} = t_{transition} + \frac{1}{f}$ and the transition window can be updated accordingly. The transition time window allows us grouping samples belonging to the same signal bit b_{cur}^j . Namely, we choose samples:

$$\{s_l^j, ..., s_{l+k}^j\} = \{s_i^j \mid timestamp(I_i) > t_e \wedge timestamp(I_i) < t_s + \frac{1}{f}\}$$

Having multiple samples per signal leads to increased robustness of the bit classification process. The resulting signal bit for a Marker j is computed as $b_{cur}^j = f(s_l^j, ...s_{l+k}^j))$ with

$$f(s_l^j, ...s_{l+k}^j)) = \begin{cases} 1, & \text{if } \frac{1}{K}\sum_{i=0}^{K} s_l^{j+i} > 0.6 \\ 0, & \text{if } \frac{1}{K}\sum_{i=0}^{K} s_l^{j+i} < 0.4 \\ 0.5, & \text{otherwise} \end{cases}$$

The resulting bitstream can be used to match the extracted to the known patterns which correspond to unique marker labels.