Factor Graph-Based Optical-IMU Motion Capture for Single Marker Tracking in the Presence of Occlusion

Andy Brinkmeyer[*] Sebastian Friston[†]
Department of Computer Science
University College London

Wolfgang Stuerzlinger[‡]
School of Interactive Arts and Technology
Simon Fraser University

Simon J. Julier[§]
Department of Computer Science
University College London

Abstract

Optical tracking of individual body-mounted markers is widely used in content-production and virtual reality. When viewed by at least two cameras, a marker's position can be estimated with sub-millimetre accuracy hundreds of times per second. However, markers can often become occluded by the environment, or by the objects or people being tracked. This causes tracking loss. Using more cameras can reduce the likelihood of occlusion, and post-capture smoothing can fill gaps. Yet, using more cameras increases cost and does not guarantee occlusion-free capture. Smoothing algorithms can provide continuous trajectories, but do not operate in real-time.

Another way to overcome this issue is to complement the optical markers with Inertial Measurement Units (IMUs). These sensors operate at high rates and do not require external references. Therefore, they can provide measurements even if a marker is not visible from any camera. The use of IMUs is common in HMD and controller tracking. However, we are not aware that this approach has been tried with individual optical markers.

In this paper, we develop a method to fuse IMU data with optical data for a single marker, based on factor graphs. Factor graphs express inference as an optimization problem. They are highly efficient and robust in the presence of significant non-linearities. By varying the length of the graph, they directly range between Kalman filters, sliding windows, and whole trajectory smoothers. We implement our system with GTSAM, using pre-integration and custom factors to model optical sensor types.

The performance of our single-marker tracking system is evaluated using a custom simulation pipeline which uses the Unity game engine to simulate trajectories and sensor measurements. We compare the performance of our estimator with camera-only triangulation, factor graphs of different lengths, in experiments that simulate different camera resolutions, sampling rates, and varying degrees of occlusion. We show that our method for visual-inertial tracking runs in real-time and can successfully fill in gaps of between 1 to 4 seconds, depending on the estimation method.

Index Terms: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Virtual reality

1 Introduction

Camera-based tracking has many applications. One of these is marker-based motion capture, which is extensively used in film, TV, and video game production. There, multiple point markers are attached to a person or object and tracked in 3D space. Their positions can then be used to resolve a human or rigid body pose. These systems are also used in VR and AR, for example to perform full body tracking for real-time control of an avatar in a virtual environment.

Yet, the accuracy of these systems is highly dependent on the number of cameras a marker can be detected in. An individual marker must be seen by at least two cameras in order to compute its position in 3D space. Further views improve the quality of the tracking through reducing noise. However, capture systems with high numbers of cameras are expensive and can be challenging to set up. Further, even with elaborate camera setups occlusions are still often inevitable. For example, markers on the inside of an arm could be blocked by the torso when swinging the arm back and forth while walking.

For content generation purposes, post-hoc interpolation techniques such as Kalman smoothers can recover or estimate the occluded parts of trajectories offline. In the worst case, missing sections can be filled in manually by an artist. Such methods are not perfect however. Further, they cannot be used for VR and AR as these applications often require high quality tracking in real-time.

Additional measurements are required to provide reliable constraints when no camera information is available. One type of sensor that is commonly used in robotics and aerospace applications to fill in gaps in between low-rate positional measurements are Inertial Measurement Units (IMUs). These are already used in head and hand trackers to reduce latency and to provide a backup when optical tracking is lost. In a backup role, their use is often restricted to orientation-only because position cannot be estimated reliably by low-cost IMUs.

In this work we investigate the use of inertial measurement fusion for *single markers* with a commercial motion capture system, the PhaseSpace Impulse X2E. We present a graph-based model for fusing the inertial data with raw optical linear detector measurements, which can be used for both real-time tracking and post-processing. Further, our approach allows estimating states using the IMU readings even if no camera measurements are available. To test our estimator and compare it to conventional optical tracking we developed a pipeline for simulating the modified motion capture system.

1.1 Contributions

We present two major contributions: a visual-inertial estimator for single markers based on factor graphs for better gap-filling during occlusion and a simulation pipeline for quickly and easily generating marker trajectories, IMU measurements, and camera observations.

We demonstrate the effect of camera resolution and framerate on our visual-inertial estimator using two distinct scenarios, one with uniform, the other with rapid movements. Each individual test case is estimated using conventional camera-only tracking, real-time visual-inertial tracking, and visual-inertial post-processing. Furthermore, we show how such a capture system can be used to improve *single-marker* estimates under occlusion.

We outline a Unity-based approach to simulating marker trajectories from skeletal animations or physics engine simulations. Also, we present a method to deal with noisy derivatives resulting from insufficient floating point precision of the Unity game engine.

[*]e-mail: ucabrin@ucl.ac.uk
[†]e-mail:sebastian.friston@ucl.ac.uk
[‡]e-mail:w.s@sfu.ca
[§]e-mail:s.julier@ucl.ac.uk

1.2 Structure of the Paper

The structure of this paper is as follows. In the next section, we present the motion capture problem of interest and describe previous approaches. Section 3 introduces factor graphs. These are generalized, efficient and highly scalable approach for handling estimation problems. In Section 4 we describe the implementation we use which combines pre-integration with optical data. The simulation pipeline is introduced in Section 5. The analysis of the algorithm are carried out in Sections 6 and 7. Our analysis explores several properties such as the effect of different camera resolutions and dropout. Our result shows significant improvements. Finally, we discuss our results and conclude our findings in Section 8 and Section 9.

2 Background

2.1 Optical Motion Capture Systems

A number of camera-based tracking techniques are employed in VR and AR. In this paper we consider point-marker based systems of the kind often used in film and TV. These systems capture human or rigid body motion in two stages. In the first, a camera system resolves the positions of a set of point-markers in 3D space. Then, the positions are used to estimate the pose of a person or rigid body. Figueroa et al. [11] describe the implementation of a representative marker tracking system.

The first step is to identify marker locations on a camera's image plane. This can be done through, for example, feature extraction on 2D images [37]. Different systems have different ways to improve the robustness of this stage. For example, making markers retro-reflective and placing IR filters over the cameras to reduce false positives, as done by OptiTrack. The location on the image plane can be found with subpixel accuracy by matching the centroid of a Gaussian, or other area matching methods [36].

Next, the system must identify correspondences between features across cameras. Once multiple 2D observations are associated with a marker, the 3D location can be estimated through triangulation [19]. One way to do this is through motion correspondence and epipolar geometry [35, 41]. Correspondences must also be made across frames to persistently label each marker, so they can be assigned to a rigid body, or bone, for pose estimation [40]. These two correspondence problems may be combined [5]. Some systems, such as the PhaseSpace, use active markers. These markers communicate their identity to each viewpoint unambiguously, allowing immediate matching between both cameras and frames.

Finally, the set of triangulated points can be used by an optimiser to estimate a human pose or rigid body transform (e.g [3]).

In this paper we are concerned with the tracking of the points in the set. We assume that where the optical system can resolve marker locations, it is generally accurate. However there are a number of ways in which it can fail to resolve at all. The most common is the case of occlusion; at least two views of a marker are needed to triangulate it. The system may not have two views due to true occlusion, or because one or more views cannot be matched. Even after dis-occlusion, matching may take time, and the marker may become mislabelled.

2.2 Inertial Motion Capture Systems

Given initial conditions, the pose of a rigid body can be computed by integrating its accelerations and angular velocities in a process known as dead-reckoning. These can be measured by Inertial Measurement Units IMUs. The process of using such inertial measurements to compute position and pose of a body is known as Inertial Navigation and is often used as one component of more complex navigation systems in aerospace and robotics [18, 38].

For human motion capture, a number of systems have been developed around wearable inertial micro-sensors. These are used in a similar manner to traditional optical systems, in which multiple points on the body are tracked - though in six-dof, rather than three [15]. Indeed, there are now more pure-inertial commercial systems than pure-optical ones [31]. Filippeschi et al. [12] and López-Nava et al. [28] provide detailed surveys of inertial human motion tracking, along with a comparison of a number of techniques with traditional optical marker based systems.

A motivation of these is to avoid the infrastructure and cost requirements of traditional optical motion capture. Most notably, as egocentric sensors, IMUs do not require any external reference so high dimensional pose data can be captured in a small space, without discontinuities or mis-labelling. However, a well-known problem with dead-reckoning is that the continuous integration leads to increasing error over time, also known as drift. Authors have attempted to tackle drift in inertial systems in different ways. Some methods improve the actual measurements through dynamically correcting calibration during use [2]. Others use highly constrained models to minimise the impact of noise in existing data [10, 33, 39]. A kinematic chain or segment model at least is almost always used in inertial human motion capture, but models can go further with assumptions about the application, a common example being footstep detection in Pedestrian Dead Reckoning [22]. Drift is a very challenging problem however and no current model based technique will prevent divergence building up over time. Accordingly, other authors have proposed augmenting inertial sensors with other egocentric sensors [27], or even re-introducing external references such as Ultra-Wide Band radio localisation [42].

2.3 Hybrid Optical / Inertial Tracking Systems

To produce more reliable estimates, often more than one sensor is used to measure the pose of an object. The process of fusing multiple measurements is called *Sensor Fusion*. However, in the context of inertial navigation and tracking we often refer to the specific application as *Pose Estimation*.

The historically best known approach for pose estimation, specifically fusing inertial and GPS measurements, is the Extended Kalman Filter (EKF) [6], which takes advantage of the complementary error characteristics of the data sources. In the IMU-GPS example, the short-term IMU error is small but drifts over time. GPS measurements are sparse but accurate. The filter can combine these in an optimal way to provide an estimate better than either alone [17].

The same approach can be applied to combining similarly complementary data sources in VR and AR, e.g., [9, 20], and indeed is very commonly used in tracking HMDs and controllers. This includes both outside-in systems comparable with traditional motion capture, such as Oculus' Constellation [32], and inside-out systems such as SLAM, used to track standalone HMDs and platforms such as mobile phones [23].

Visual-inertial systems are used for direct human motion capture as well, though they often continue to focus on fusing complete rigid-body estimates. For example, Li et al. [26] used a complementary filter to correct for drift in an inertial microsensor system with six-dof transforms from mounted SteamVR trackers. Kobayashi et al. [25] used a traditional motion capture system to track a rigid-body, then switched between this pose, or one estimated from a coincident IMU, depending on whether the optical pose was available. While visual-inertial systems are common in a number of forms, we are not aware of any system that has combined optical and inertial tracking for individual markers in a point-based system.

3 Factor Graphs for Estimation

In this section we introduce factor graphs. Factor graphs are a general way to formulate tracking and estimation problems. We begin by introducing basic notation and discuss a filtering context. We then extend this to the full general case.

3.1 Filtering Formulations

We seek to estimate the state $\mathbf{x}_k$ of a marker at time step k given a sequence of observations from a set of cameras. We model the system using the familiar two-step structure of time prediction followed by measurement update.

The prediction step models how the marker state evolves over time. This is expressed by the process model

$$\mathbf{x}_k = \mathbf{f}\left[\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{v}_k\right], \tag{1}$$

where $\mathbf{u}_k$ are the control inputs and $\mathbf{v}_k$ is the process noise. This can also be described using the state transition density

$$p\left(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{u}_k\right). \tag{2}$$

In the update step, the marker is observed by one or more cameras. First consider the case of a single camera. The observation from the jth camera is given by

$$\mathbf{z}_k^j = \mathbf{f}\left[\mathbf{x}_k, \mathbf{c}^j, \mathbf{w}_k^j\right], \tag{3}$$

where $\mathbf{c}^j$ is the pose of the jth camera, and $\mathbf{w}_k^j$ is the observation noise. This arises from, for example, the errors in centroiding algorithms. We model this as the measurement likelihood

$$l\left(\mathbf{x}_k; \mathbf{z}_k^j, \mathbf{c}^j\right) = p\left(\mathbf{z}_k^j | \mathbf{x}_k, \mathbf{c}^j\right). \tag{4}$$

To handle multiple markers we need to introduce additional notation. Suppose at time step k the marker is observed by C_k cameras. The detection system provides the multicamera observation structure $\mathbf{Z}_k$. This structure contains a C_k dimensional vector $\mathbf{c}_k$ which indexes the cameras, and the measurements. Therefore, the general expression for the likelihood is

$$l\left(\mathbf{x}_k; \mathbf{Z}_k\right) = \prod_{j=1}^{C_k} p\left(\mathbf{z}_k^{c_k^j} | \mathbf{x}_k, \mathbf{c}^{c_k^j}\right). \tag{5}$$

Defining the sets $\mathbf{U}_{1:k} = \{\mathbf{u}_1, \ldots, \mathbf{u}_k\}$ and $\mathbf{Z}_{1:k} = \{\mathbf{Z}_1, \ldots, \mathbf{Z}_k\}$ to be the collection of all control inputs and all measurements over time, the marker tracking problem is to estimate the properties of the probability distribution

$$p\left(\mathbf{x}_k | \mathbf{U}_{1:k}, \mathbf{Z}_{1:k}\right). \tag{6}$$

In filtering algorithms this is achieved in two steps: prediction followed by update. The prediction is computed using the Chapman-Kolmogorov equation

$$p\left(\mathbf{x}_k | \mathbf{U}_{1:k}, \mathbf{Z}_{k-1}\right) = \int p\left(\mathbf{x}_k | \mathbf{x}', \mathbf{u}_k\right) p\left(\mathbf{x}' | \mathbf{x}_{k-1}, \mathbf{u}_{k-1}\right) \mathrm{d}\mathbf{x}'. \tag{7}$$

The update is achieved using Bayes Rule

$$p\left(\mathbf{x}_k | \mathbf{U}_{1:k}, \mathbf{Z}_{1:k}\right) = \frac{l\left(\mathbf{x}_k; \mathbf{Z}_k\right) p\left(\mathbf{x}_k | \mathbf{U}_{1:k}, \mathbf{Z}_{1:k-1}\right)}{p\left(\mathbf{Z}_k | \mathbf{U}_{1:k}, \mathbf{Z}_{1:k-1}\right)} \tag{8}$$

where

$$p\left(\mathbf{Z}_k | \mathbf{U}_{1:k}, \mathbf{Z}_{1:k-1}\right) = \int l\left(\mathbf{x}'; \mathbf{Z}_k\right) p\left(\mathbf{x}' | \mathbf{U}_{1:k}, \mathbf{Z}_{1:k-1}\right) \mathrm{d}\mathbf{x}' \tag{9}$$

In many practical situations, (7), (8) and (9) do not have closed form solutions and approximations of some kind must be used. The Kalman filter is one of the best-known and widely-used algorithms. It propagates only the first two moments of $p\left(\mathbf{x}_k | \mathbf{U}_{1:k}, \mathbf{Z}_{1:k}\right)$ and uses a linear least squares error update rule. Although this has been successfully applied in many motion tracking systems, the fact that it only stores the first two moments means that it cannot represent nonlinear dependency structures such as those which result from large angular errors accurately. These structures arise, for example, when using IMUs in very aggressive manoeuvres. Several approaches such as particle filters and Gaussian mixture models can be used. However, these can be complicated and time consuming to implement.

A second issue is that a filtering algorithm only maintains the current state of the marker. Although this is appropriate for real-time tracking, in offline motion capture the entire history of data is used.

Recently in robotics there has been a development using factor graphs. These generalize filters to batch estimation problems and are now widely used in target tracking,

3.2 Factor Graph Representation

Factor graphs are a way to describe and carry out inference over complicated probabilistic networks of random variables. They have found widespread use in robotics and target tracking. Their success is due to the surprising fact that filtering algorithms are not the most efficient way to address challenging estimation problems from either a theoretical or a practical perspective.

Factor graphs are used to compute the *joint* density of the target state over time. In other words, the goal is to compute

$$p\left(\mathbf{x}_{1:k} | \mathbf{U}_{1:k}, \mathbf{Z}_{1:k}\right). \tag{10}$$

rather than just the current time step k. The reason why this is more efficient lies directly in the Markov structure of (2) and (5).

First consider the case in which a marker moves and no observations are available. The joint density $p\left(\mathbf{x}_{1:k} | \mathbf{U}_{1:k}\right)$ can be decomposed using the probabilistic chain rule into

$$p\left(\mathbf{x}_{1:k} | \mathbf{U}_{1:k}\right) = p\left(\mathbf{x}_k | \mathbf{x}_{1:k-1}, \mathbf{U}_{1:k}\right) p\left(\mathbf{x}_{1:k-1} | \mathbf{U}_{1:k}\right) \tag{11}$$

Substituting from (2), this simplifies to

$$p\left(\mathbf{x}_{1:k} | \mathbf{U}_{1:k}\right) = p\left(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{u}_k\right) p\left(\mathbf{x}_{1:k-1} | \mathbf{U}_{1:k-1}\right) \tag{12}$$

By recursion,

$$p\left(\mathbf{x}_{1:k} | \mathbf{U}_{1:k}\right) = p\left(\mathbf{x}_1\right) \prod_{i=2}^{k} p\left(\mathbf{x}_i | \mathbf{x}_{i-1}, \mathbf{u}_i\right) \tag{13}$$

In other words, the joint distribution is given by multiplying the distributions of each transition density on its own. This expression has several advantages. First, it is exact. Second, no integration is required. Finally, it is specified purely in terms of the local transition model rather than the global uncertainty.

Observations are included by applying Bayes rule and using the Markov structure. It can be shown that this leads to,

$$p\left(\mathbf{x}_{1:k} | \mathbf{U}_{1:k}, \mathbf{Z}_{1:k}\right) \propto p\left(\mathbf{x}_1\right) \prod_{i=2}^{k} p\left(\mathbf{x}_i | \mathbf{x}_{i-1}, \mathbf{u}_i\right) \prod_{i=2}^{k} l\left(\mathbf{x}_i; \mathbf{Z}_k\right). \tag{14}$$

Again, this expression is *exact*.

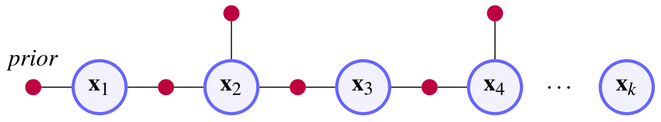


Figure 1: Factor graph representation of the tracking problem. Variable nodes are the blue circles. Red dots correspond to the factors, which are the conditional probabilities linking state variables together. In this example $\mathbf{x}_1$ is constrained by a prior, $\mathbf{x}_2$ and $\mathbf{x}_4$ are observed, and state $\mathbf{x}_2$ and $\mathbf{x}_k$ are not observed.

Figure 1 shows a factor graph representation of (14). As the figure shows, missing measurements are modelled by the graph not containing measurement factors at a given timestep.

3.3 Inference on Factor Graphs

Factor graphs provide an efficient, closed-form way to write down the joint density of the marker state over time. However, to carry out tracking, a single state value must be extracted. One option would be to compute the mean of $\mathbf{x}_k$ from (14). However, this requires integrating over the joint distribution and will be impractical. An alternative is to use the Maximum A-Posteriori (MAP) estimate. The MAP estimate is given by

$$\mathbf{x}^*_{1:k} = \underset{\mathbf{x}_{1:k}}{\arg\max}\, p\left(\mathbf{x}_{1:k}|\mathbf{U}_{1:k},\mathbf{Z}_{1:k}\right) \tag{15}$$

The argmax value is unaffected by normalisation constants. Further simplifications arise from the fact that the argmax of $\log p\left(\mathbf{x}_{1:k}|\mathbf{U}_{1:k},\mathbf{Z}_{1:k}\right)$ is the same as that for $p\left(\mathbf{x}_{1:k}|\mathbf{U}_{1:k},\mathbf{Z}_{1:k}\right)$. Substituting from (14),

$$\mathbf{x}^*_{1:k} = \underset{\mathbf{x}_{1:k}}{\arg\max} \left(\log p\left(\mathbf{x}_1\right) + \sum_{i=2}^{k} \log p\left(\mathbf{x}_i|\mathbf{x}_{i-1},\mathbf{u}_i\right) + \sum_{i=2}^{k} \log l\left(\mathbf{x}_i;\mathbf{Z}_k\right)\right). \tag{16}$$

If it is assumed that the prior, the transition densities, and likelihood functions are Gaussian, inference becomes the solution of a nonlinear least squares optimization problem, which can be solved exceptionally efficiently. Using nonlinear optimisation methods including Gauss-Newton or Levenberg-Marquardt, libraries such as GTSAM, Ceres, and g2o are able to solve extremely large optimization problems in real time. In this paper we chose to use the iSAM2 algorithm from GTSAM for real-time tracking. This is a highly efficient system optimizer designed for systems which incrementally grow over time. For example, Vadim et al. demonstrated in 2012 that iSAM2 could be used for optical inertial visual tracking on a drone at more than 1kHz [21].

Factor graphs provide a very efficient, general and flexible formulation for addressing estimation problems. We now look at how to formulate the single marker tracking problem in a factor graph.

4 Factor Graph Formulation for IMU Motion Capture

4.1 visual-inertial Capture System

Our visual-inertial capture system consists of two components: a conventional optical marker tracking system and IMUs that are attached to the rigid bodies the markers are placed on. The optical system produces a set of raw camera measurements for each marker, i.e., raw positions of intensity spikes on the camera sensor. The IMUs provide linear acceleration and angular velocity measurements at rates higher than the camera system. Here we present a graphical model that uses these two measurement sources to model the underlying inference problem and compute pose estimates.

4.2 Graphical Model

As outlined in the previous sections, factor graphs are well suited to model nonlinear inference problems. A general graph design is adapted from [21] where IMU and GPS measurements are used to estimate an objects pose and velocity as well as the IMU biases. The state of the marker consists of its position, orientation and angular velocity. In addition, we estimate the IMU bias states. Our graph is shown in Figure 2.

We modified [21] by replacing the GPS measurements with those from the camera system, which are similar in the sense that they provide a direct drift-free constraint of the markers position. Indeed, when first estimating the marker position from just the camera measurements, the point estimate can be added to the graph as a single loosely coupled[1] GPS-like factor. Such a factor acts on a single variable, in our case on a single pose $\mathbf{P}_i$, and constraints the markers position. This approach requires the marker to be visible from at least two cameras to produce a position estimate and a valid factor for the graph. However, when only one measurement is available no camera-based estimate can be computed and therefore no GPS-like factor can be added even though the measurement would provide some constraints on the markers position. Therefore we adapt a tightly coupled approach where we introduce raw camera measurements to the graph which is also more expressive since it keeps the raw measurement covariances of the camera sensors.

4.3 Optical Measurements

Our proposed system is based on the PhaseSpace motion capture system which uses 1D optical linear detectors instead of conventional 2D cameras. We adapted the classical pinhole camera model to this type of sensor which results in the projection equation

$$\begin{aligned}\hat{\mathbf{u}} = \begin{bmatrix}\hat{u}\\ \lambda\end{bmatrix} = \mathbf{C}\begin{bmatrix}\mathbf{x}\\ 1\end{bmatrix} = \mathbf{IRT}\begin{bmatrix}\mathbf{x}\\ 1\end{bmatrix} \\ = \begin{bmatrix}f & 0 & c\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}r_{11} & r_{12} & r_{13}\\ r_{21} & r_{22} & r_{23}\\ r_{31} & r_{32} & r_{33}\end{bmatrix}\begin{bmatrix}1 & 0 & 0 & -x_{cam}\\ 0 & 1 & 0 & -y_{cam}\\ 0 & 0 & 1 & -z_{cam}\end{bmatrix}\begin{bmatrix}\mathbf{x}\\ 1\end{bmatrix}\end{aligned} \tag{17}$$

Here, $\mathbf{I}$ is the intrinsic matrix with the focal length f and sensor centre offset c, $\mathbf{R}$ is the rotation matrix that rotates a vector from the local frame to the sensor frame, and $\mathbf{T}$ is the translation matrix with the position $\mathbf{x_{cam}}$ of the linear detector in the local coordinate system. The normalised detector measurement can now be computed as

$$u = \frac{\hat{u}}{\lambda} \tag{18}$$

Both equations can be combined into the measurement function

$$h(\mathbf{P}) = \frac{\hat{u}}{\lambda} = \frac{c_{11}x + c_{12}y + c_{13}z + c_{14}}{p_{21}x + c_{22}y + c_{23}z + c_{24}} \tag{19}$$

where c_{ij} are the entries of the projection matrix $\mathbf{C}$. Note that $h(\mathbf{P})$ is nonlinear due to the homogeneous normalisation. Also, $h(\mathbf{P})$ is optimised over the group of poses, the six-dimensional Lie Group $SE(3)$. Therefore, the special structure of $SE(3)$ needs to be respected when deriving the Jacobian. The pose increments $\boldsymbol{\xi}$ are defined in the tangent space of $SE(3)$ at the identity and mapped back on the manifold through the exponential map. Hence, an incremental change to a pose is expressed as $\mathbf{P}_0 e^{\boldsymbol{\xi}}$. The resulting Jacobian of the measurement function is defined as

$$\frac{\partial h\left(\mathbf{P}_0 e^{\xi}\right)}{\partial \boldsymbol{\xi}} = \begin{bmatrix}\mathbf{0}_{1\times 3} & \mathbf{J}_t\mathbf{R}_0\end{bmatrix} \tag{20}$$

where $\mathbf{J}_t$ is the Jacobian with respect to the position

$$\mathbf{J}_t = \begin{bmatrix}\frac{c_{11}\lambda - c_{21}\hat{u}}{\lambda^2} & \frac{c_{12}\lambda - c_{22}\hat{u}}{\lambda^2} & \frac{c_{13}\lambda - c_{23}\hat{u}}{\lambda^2}\end{bmatrix} \tag{21}$$

and $\mathbf{R}_0$ is the rotation matrix associated with the pose $\mathbf{P}_0$, a product of the Jacobian of the exponential map [1]. The measurement function in Equation 19 and its Jacobian in Equation 20 can then be used to define a new *Linear Detector Factor* for use in our factor graph.

[1] A *loosely coupled* GPS integration is one where the GPS position estimate is fused with the navigation solution. In contrast, a *a tightly coupled* integration uses the raw GPS pseudo-range and Doppler measurements in the estimator.

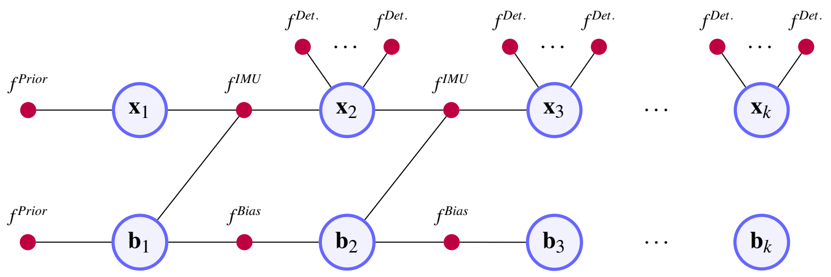


Figure 2: Factor graph used for estimating the poses and velocity of a single marker and the bias vector of the IMU.

4.4 Inertial Measurements

IMUs usually operate at rates much higher than other sensors used for navigation, e.g., GPS or cameras. To use such high-rate measurements in a filter or smoother, measurements in between keyframes (frames where additional measurements from lower rate sensors are available) must either be dropped or additional states must be added. This results in either a loss of information or significantly higher prediction rates. Another difficulty is proper initialisation of the inertial integration. The high non-linearity of the attitude equations can result in large errors or even non-convergence of the estimate in filters and smoothers with incorrect initialisation [29]. More precisely, an incorrect initial attitude leads to wrong compensation of the gravity vector and therefore large errors in the estimated body acceleration. But good initialisation is not always available or may change when new measurements provide additional information.

To avoid adding unnecessary states and to avoid committing to an initialisation point, inertial measurements can be integrated without initial conditions in between two poses to form a single pseudo inertial measurement as first proposed by [29]. This *preintegrated* measurement can then be used to constrain the movement in between two keyframes. Preintegration is especially useful in estimation methods that re-linearise previous states since the preintegrated measurement can easily be applied to new linearisation points. This avoids storing the individual IMU measurements and reintegrating them whenever re-linearisation is applied.

While the work of [29] and [30] pioneered the idea of preintegration, their method used Euler angles for representing rotations which suffer from singularities. In [13] and [14] those shortcomings were addressed by taking the manifold structure of the rotation group $SO(3)$ into account. Furthermore, they presented an improved method of uncertainty propagation and a-posteriori bias correction for IMU preintegration. Further advancements were presented in [7, 8] by deriving a closed form solution to the preintegration equations. Instead of using discrete integration as the previous works they introduce two analytical models: piecewise constant measurements and piecewise constant local true acceleration.

The preintegrated IMU measurement can then be used as a factor to constrain the movement in between two poses..

5 Simulating Motion Capture Data

To evaluate the accuracy of our estimation system we require ground truth motion data. Recording such data in a motion capture system is difficult since determining the true position of a point in free space is a hard task. Thus, we developed a simulation pipeline that generates marker trajectories, computes their corresponding IMU measurements, and simulates individual camera outputs, even for complex human motions. Simulation has been used in the past to investigate the design space of a tracking system [34]. The use of simulated data also allows us to control the noise models of the cameras and IMUs to better understand the behaviour of our system.

5.1 Simulation Pipeline

Fig. 3 illustrates our simulation pipeline. We start by interactively placing the virtual capture system and markers using the Unity editor. Subsequently, we use the Unity game engine to simulate the tracked object or person through skeletal keyframe-based animations or the physics engine and compute visibility of markers from individual cameras. We then use Matlab for processing the marker poses generated by Unity. This involves reducing quantization noise, computing kinematics, and simulating IMU measurements. Next, we feed the true poses into our C++ library that simulates raw linear detector measurements. Together with the IMU measurements, these are used by our GTSAM[2]-based estimator to construct the factor graph and perform the optimisation. Finally, the estimates are compared to the ground truth.

5.2 Rigid Body Simulation

We developed a Unity-based framework to simulate optical marker trajectories and marker occlusion. Our approach makes it simple to set up a desired marker and capture configuration by using the Unity scene editor, as shown in Fig. 4. Once the camera system and the markers are set up, we move the virtual body using skeletal animations or the physics engine.

To record the motion at high sampling rates, we set the Unity capture rate to the desired frame rate, which decouples the game engine loop from the real time clock and allows Unity to run animations at the appropriate speed to achieve the selected capture rate.

For each frame we record the marker poses and determine occlusion. To simulate occlusion we use ray casting to compute the visibility of markers. Note that at this point in the simulation pipeline we simply determine if there is a direct line of sight without considering camera intrinsic properties like the field of view, which we do in the C++ library.

5.3 IMU Simulation

To simulate IMU measurements we first need to obtain the true acceleration and angular velocity of the markers. We use the rigid body simulation detailed in the previous section as a basis.

To obtain the linear acceleration and velocity we numerically differentiate the marker positions and orientations. Simply applying central differences to the marker positions results in noisy measurements when using high sample rates as shown in Figure 5.

We found that the differentiation noise is caused by the internal use of 4-byte floating point numbers for poses in the Unity engine. The high sampling rates result in small differential movements between frames that exhaust the floating point precision and in turn cause quantization errors which are amplified through differentiation. The resulting acceleration measurements are too noisy to be used for simulating the IMU.

Since the Unity-internal number format cannot be changed, the measurements need to be smoothed. We tested a method based on

[2] `https://gtsam.org/`

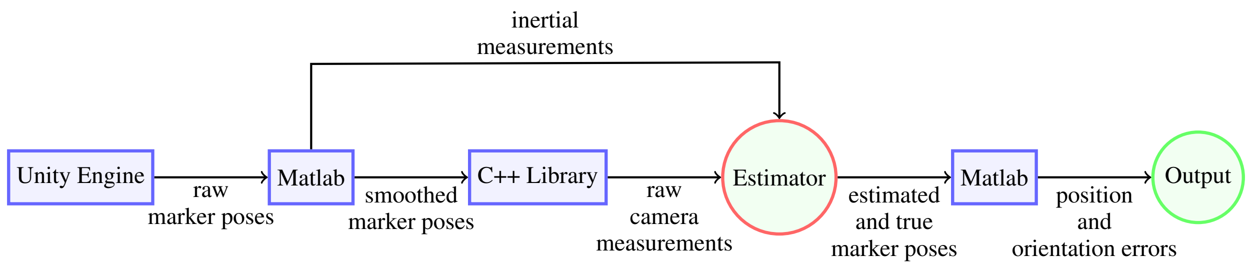


Figure 3: Visualisation of the simulation pipeline.

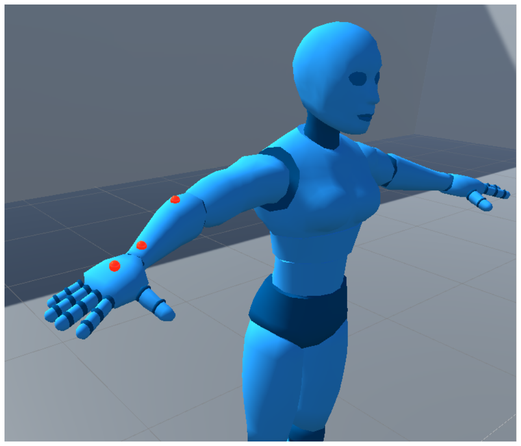


Figure 4: Three markers (red dots) attached to the right arm of a human model in Unity using the simulation framework.

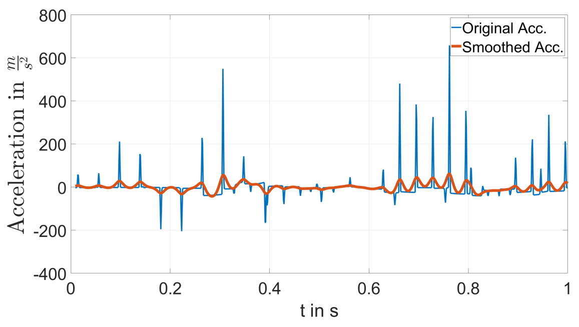


Figure 5: Original acceleration obtained from numerically differentiating (central differences) the marker position twice and acceleration obtained by analytically differentiating the smoothing spline fitted to the marker position twice. Both are measured in $\mathrm{m\,s^{-2}}$. A smoothing parameter of $p = 1 - 1 \times 10^{-6}$ was used. The position-spline fitting error (RMS) is in the order of 1×10^{-3} m.

total-variation regularisation proposed in [4] which was not able to achieve the desired results and introduced significant position errors when re-integrating the smoothed acceleration. We thus chose a different approach, where the position of each marker is approximated by three cubic smoothing splines (one for each coordinate direction). A cubic smoothing spline $s(t)$ is one that minimises the cost function

$$c = p\sum_i (x_i - s(t_i))^2 + (1-p)\int \left(s''(t)\right)^2 dt \qquad (22)$$

Here p is a smoothing parameter between 0 and 1 which controls the amount of smoothing. Larger values of p result in a tighter fit while lower values result in smaller second derivatives and therefore a looser but also smoother approximation. To compute the goodness of the fit the root mean square error (RMSE) is computed as

$$e_{RMS} = \sqrt{\frac{\sum_i \left(x_{i,true} - x_{i,spline}\right)^2}{N}} \qquad (23)$$

where x_{true} is the position (one direction of the position vector is considered at a time) obtained from the rigid body simulation, x_{spline} is the position as obtained from the smoothing spline and N is the total number of frames.

Figure Fig. 5 shows how fitting a smoothing spline to the simulated position can result in smoother accelerations without introducing large errors in the position measurements. However, long trajectories can quickly result in long computation times.

The angular velocity is obtained from the rotations and is usually sufficiently smooth without further processing.

We then simulate the IMU measurements using the computed acceleration and angular velocity. This process consists of first transforming the measurements to the IMUs' body-fixed coordinate system and adding the gravity vector to the accelerations. Then, sensor-characteristic errors, such as noise, biases, and measurement range, are added. The Matlab Navigation Toolbox provides an IMU model which can be configured to exhibit specific noise and measurement characteristics. Once configured, the IMU model is fed true linear accelerations, angular velocities, and rotations to compute the simulated IMU measurements.

5.4 Camera Simulation

To simulate the camera measurements, we use the projection matrix defined in Equation 17 to map a marker from world coordinates to the linear detector sensor. We then check if the theoretical measurement actually lies on the camera sensor.

Another aspect to consider are the measurement errors of the camera sensors, or more precisely the errors of the linear detector sensors that make up the PhaseSpace cameras. While there are several error sources that could be considered individually we assumed that they behave in the aggregate as zero-mean normal distributed noise on the sensor. In the simulation this error term is directly added to the raw sensor measurement before quantization.

6 Evaluation

To evaluate our estimator we set up a virtual capture system, as shown in Figure 6, with 8 cameras evenly spaced around a cubic capture volume of about 10 m x 10 m x 3 m. We captured data for two types of motion:

1. A cube rotating around its vertical axis moving up and down with a marker placed on one corner. This motion has uniform linear accelerations and angular velocities that gener-

ally conform to the assumptions of simple linear predictors and smoothers, e.g., Eulers method. The maximum linear acceleration was $16\,\mathrm{m\,s^{-2}}$ and the maximum angular velocity $1.57\,\mathrm{rad\,s^{-1}}$.

2. A human dancing with a marker placed on the back of their hand. This motion is characterized by rapid, organic movements with large higher order terms that generally do not conform to the assumptions of simple predictors. Maximum linear accelerations were about $130\,\mathrm{m\,s^{-2}}$ and maximum angular velocities about $17\,\mathrm{rad\,s^{-1}}$.

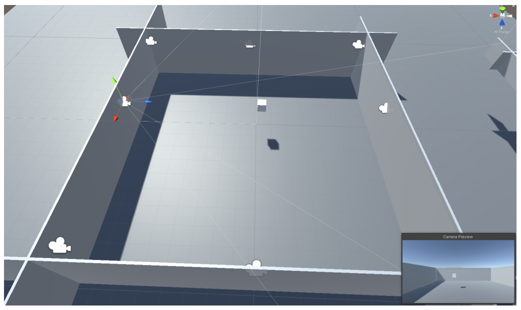


Figure 6: Unity capture setup as used for evaluating the estimator performance.

Both sequences were driven by keyframe animations and lasted approximately 15 s. For the IMU parameters, we used the specification of the Bosch BMI263 IMU with a sampling rate of 1 kHz.

We computed synthetic sensor data for the sequences (Section 5.1), then estimated motion from this using three methods: (i) conventional camera-only tracking that minimises the algebraic projection error (*Camera*), (ii) real-time incremental fixed lag smoothing based on iSAM2 [24] with a 10-states sliding window (*Ours*), and (iii) post-processing through full-trajectory batch optimisation using a Levenberg-Marquardt optimizer (*Batch*). By adjusting properties of the virtual system, we investigated the sensitivity of each method to camera resolution and frame rate, and robustness to occlusion. Performance was measured through the error, computed as the Euclidean distance between the method's estimate and the ground truth position, for each frame of the capture sequence.

7 Results

7.1 Resolution & Framerate

We first investigated the effect of camera resolution. The frame rate was held constant at 960 FPS while we varied the effective resolution of the virtual linear detectors. To account for sub-pixel processing, we assumed that these methods result in a tenfold increase in effective resolution over the actual physical sensor resolution [16].

The results are shown in Figure Fig. 7. As expected, error generally decreased with higher resolution. Error was also dependent on motion, with Sequence 1 showing generally lower errors than Sequence 2. The effect of motion type was most pronounced for our method. While the visual-inertial measurements lead to better estimates for Sequence 1, they caused higher mean errors and larger variances for Sequence 2. Batch optimisation always performs better than fixed lag smoothing. In all cases, the system achieved a sub-millimeter mean error.

Next, we kept the effective resolution constant at 36000 pixels (3600 pixels sensor resolution) and tested the system with six frame rates: 960, 420, 360, 120, 60, and 30 fps. These rates are commonly found capture rates, as advertised by various motion capture system and camera manufacturers.

The results are shown in Fig. 8. Again, Sequence 1 had generally lower error than Sequence 2 for all estimation methods. The visual-inertial estimates degrade with lower capture rates. Again, Sequence 1 has lower error than Sequence 2. There is a marked increase in outliers as frame rate drops. Also, while the batch method is able to compensate fairly well for the loss of frame rate for the predictable motion (Sequence 1), it does not perform much better than the fixed lag method for the dancing motion (Sequence 2).

7.2 Occlusion

Sensitivity to occlusion was tested by removing camera measurements for a maximum of 4 s. We compared our system against a constant velocity interpolation as shown in Figure Fig. 9a for Sequence 1 and Fig. 9c for Sequence 2.

The results show that for occlusion of up to 4 s, our estimator is able to perform significantly better than constant velocity interpolation in a real-time setting. Especially for short periods of occlusion, below 1 s, we can show that the estimator error stays below 10 mm for Sequence 1 and below 110 mm for Sequence 2. However, the amount of positional drift that accumulates over a few seconds is still significant and quickly reaches multiple tens of centimetres.

The full-trajectory batch optimisation can recover the trajectory to within 3 mm for Sequence 1 and 53 mm for Sequence 2 (Figures 9b & 9d).

8 Discussion

Our results show that when there is occlusion, the use of IMU data significantly reduces the errors, even if batch smoothing of the whole trajectory is used. However, the results also show that, that when a marker is detected by the camera the tracking performance can be slightly worse.

We believe that this counter-intuitive behaviour can be partially ascribed to the simulation system used. Our simulation setup assumes a nearly perfect capture system in such that the cameras are perfectly calibrated and produce without any exception reliable results within the predefined raw sensor variance. Furthermore, we do not consider any optical distortion. This is in contrast to real world systems that suffer from calibration inaccuracies and processing errors that reduce the quality of position estimates. Also, we assume a consistent tenfold increase in effective resolution through methods like matching the centroid of a Gaussian. All these assumptions make our simulated camera system more robust and reliable than a real world counterpart. We will address this in future work when we apply these algorithms to a real system.

9 Conclusions & Future Work

We presented a design for a single-marker visual-inertial capture system and an estimator that is capable of dealing with short periods of occlusion. Our estimator is based on an expressive graph model, commonly used in robotics and aerospace, that allows simple and intuitive "plug-and-play" capability for adding several camera and inertial measurements without additional modifications. We developed a simulation pipeline based on Unity and Matlab for quickly generating optical marker trajectories based on real-world or artificial motion data. Our results show that visual-inertial sensor fusion *at the single marker level* is a promising approach to deal with short periods of occlusion and significantly outperforms constant-velocity interpolation. Especially during sub-second occlusion our approach can be used to reliably fill in the gaps at high sampling rates. In situations where visibility is good, our system is not able to improve the camera-only estimates. We therefore suggest a dual architecture: conventional tracking when the optical system can produce an estimate and our visual-inertial approach when a marker is occluded.

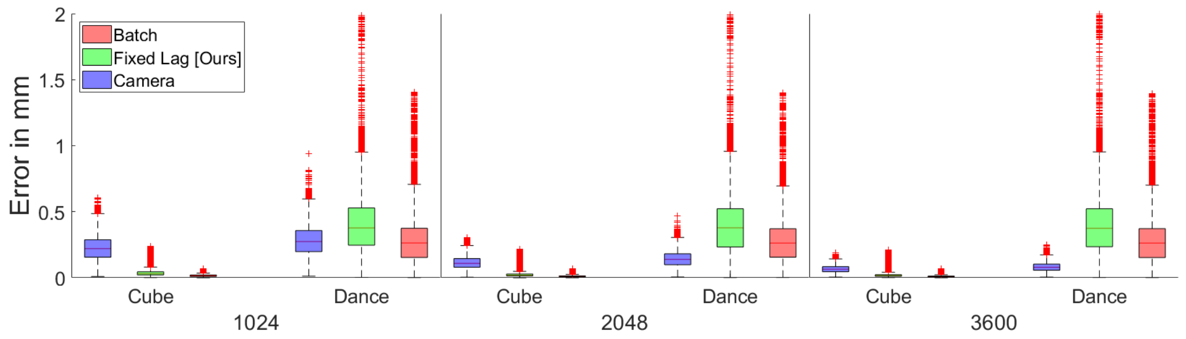


Figure 7: Errors of the estimates compared to the ground truth for different camera resolutions in pixels. The effective resolution (resulting from sub-pixel processing methods) is ten times the camera resolution.

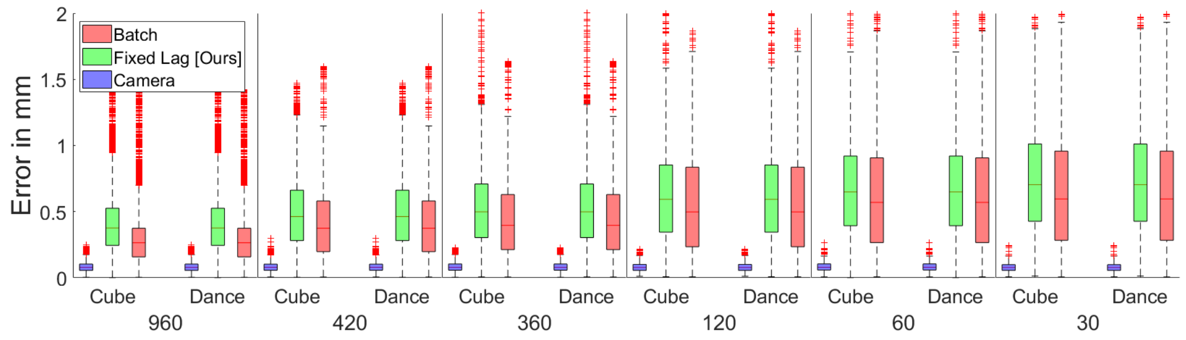
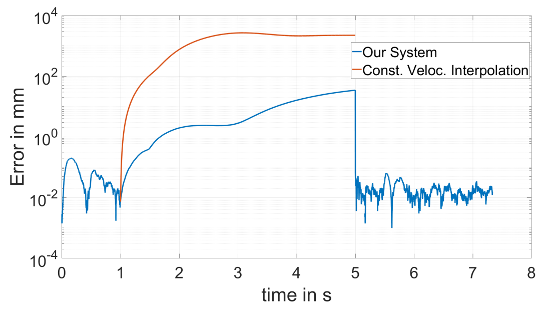
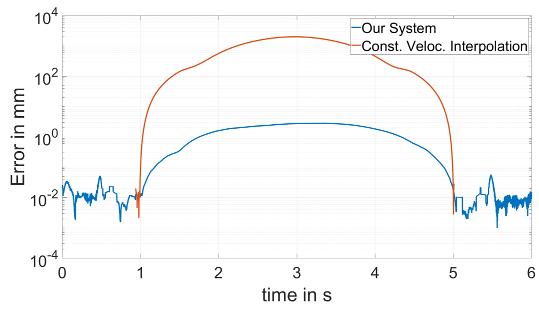


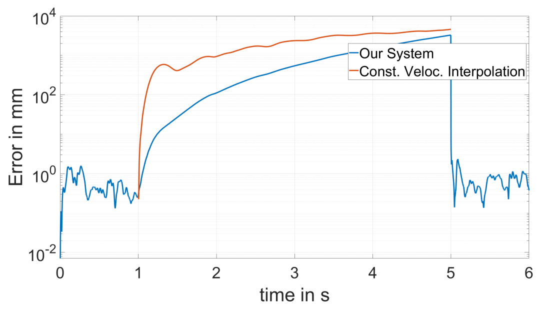
Figure 8: Errors of the estimates compared to the ground truth for different camera rates in frames per second.



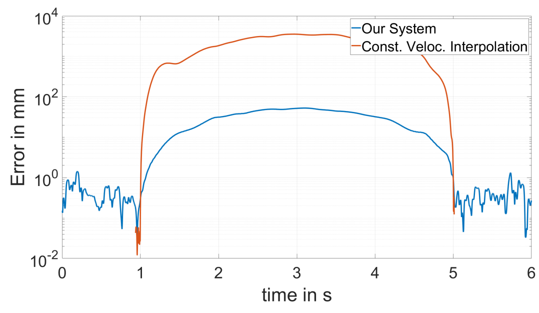
(a) Fixed lag smoother sequence 1.



(b) Full-trajectory batch estimation sequence 1.



(c) Fixed lag smoother sequence 2.



(d) Full-trajectory batch estimation sequence 2.

Figure 9: Results for fixed lag smoothing (left) and full batch estimation (right) for the sequence 1 (top) and sequence 2 (bottom). In each case, the marker was occluded for 4 s in between the 1 s and 5 s mark.

References

[1] J. L. Blanco. A tutorial on SE(3) transformation parameterizations and on-manifold optimization. Technical report, University of Malaga, 09 2010.

[2] H. T. Butt, M. Pancholi, M. Musahl, P. Murthy, M. A. Sanchez, and D. Stricker. Inertial Motion Capture Using Adaptive Sensor Fusion and Joint Angle Drift Correction. In *FUSION 2019 - 22nd International Conference on Information Fusion*. ISIF - International Society of Information Fusion, 2019.

[3] J. Cameron and J. Lasenby. Estimating Human Skeleton Parameters and Configuration in Real-Time from Markered Optical Motion Capture. In *Articulated Motion and Deformable Objects*, vol. 5098 LNCS, pp. 92–101. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. doi: 10.1007/978-3-540-70517-8_10

[4] R. Chartrand. Numerical differentiation of noisy, nonsmooth data. *ISRN Applied Mathematics*, 2011:1–11, 2011. doi: 10.5402/2011/164564

[5] M. Chen, G. AlRegib, and B.-H. Juang. Trajectory triangulation: 3D motion reconstruction with l1 optimization. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4020–4023. IEEE, may 2011. doi: 10.1109/ICASSP.2011.5947234

[6] J. Crassidis. Sigma-point kalman filtering for integrated GPS and inertial navigation. *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2005. doi: 10.2514/6.2005-6052

[7] K. Eckenhoff, P. Geneva, and G. Huang. Closed-form preintegration methods for graph-based visual–inertial navigation. *The International Journal of Robotics Research*, 38(5):563–586, 2019. doi: 10.1177/0278364919835021

[8] K. Eckenhoff, P. Geneva, and G. Huang. *High-Accuracy Preintegration for Visual-Inertial Navigation*, pp. 48–63. Springer International Publishing, Cham, 2020. doi: 10.1007/978-3-030-43089-4_4

[9] W. Fang, L. Zheng, H. Deng, and H. Zhang. Real-Time Motion Tracking for Mobile Augmented/Virtual Reality Using Adaptive Visual-Inertial Fusion. *Sensors*, 17(5):1037, may 2017. doi: 10.3390/s17051037

[10] B. Fasel, J. Sporri, J. Chardonnens, J. Kroll, E. Muller, and K. Aminian. Joint Inertial Sensor Orientation Drift Reduction for Highly Dynamic Movements. *IEEE Journal of Biomedical and Health Informatics*, 22(1):77–86, jan 2018. doi: 10.1109/JBHI.2017.2659758

[11] P. J. Figueroa, N. J. Leite, and R. M. Barros. A flexible software for tracking of markers used in human motion analysis. *Computer Methods and Programs in Biomedicine*, 72(2):155–165, oct 2003. doi: 10.1016/S0169-2607(02)00122-0

[12] A. Filippeschi, N. Schmitz, M. Miezal, G. Bleser, E. Ruffaldi, and D. Stricker. Survey of Motion Tracking Methods Based on Inertial Sensors: A Focus on Upper Limb Human Motion. *Sensors*, 17(6):1257, jun 2017. doi: 10.3390/s17061257

[13] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza. IMU preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation. *Robotics: Science and Systems XI*, 2015. doi: 10.15607/rss.2015.xi.006

[14] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza. On-manifold preintegration for real-time visual–inertial odometry. *IEEE Transactions on Robotics*, 33(1):1–21, 2017. doi: 10.1109/tro.2016.2597321

[15] E. Foxlin. Inertial Head-Tracker Sensor Fusion by a Complementary Separate-Bias Kalman Filter. In *Proceedings of the IEEE 1996 Virtual Reality Annual International Symposium*, pp. 185–194, Mar. 1996. doi: 10.1109/VRAIS.1996.490527

[16] A. Gaschler. Real-time marker-based motion tracking: application to kinematic model estimation of a humanoid robot. Master's thesis, Technical University Munich, 2011.

[17] M. S. Grewal, L. R. Weill, and A. P. Andrews. *Global Positioning Systems, Inertial Navigation, and Integration*. John Wiley & Sons, Inc., New York, USA, dec 2000. doi: 10.1002/0471200719

[18] P. D. Groves. Principles of GNSS, inertial, and multisensor integrated navigation systems, 2nd edition [book review]. *IEEE Aerospace and Electronic Systems Magazine*, 30(2):26–27, 2015.

[19] R. I. Hartley and P. Sturm. Triangulation. *Computer Vision and Image Understanding*, 68(2):146–157, nov 1997. doi: 10.1006/cviu.1997.0547

[20] C. He, P. Kazanzides, H. Sen, S. Kim, and Y. Liu. An Inertial and Optical Sensor Fusion Approach for Six Degree-of-Freedom Pose Estimation. *Sensors*, 15(7):16448–16465, jul 2015. doi: 10.3390/s150716448

[21] V. Indelman, S. Williams, M. Kaess, and F. Dellaert. Factor Graph Based Incremental Smoothing in Inertial Navigation Systems. In *2012 15th International Conference on Information Fusion*, pp. 2154–2161, July 2012.

[22] A. R. Jiménez, F. Seco, C. Prieto, and J. Guevara. A comparison of pedestrian dead-reckoning algorithms using a low-cost MEMS IMU. In *WISP 2009 - 6th IEEE International Symposium on Intelligent Signal Processing - Proceedings*, pp. 37–42, 2009. doi: 10.1109/WISP.2009.5286542

[23] L. Jinyu, Y. Bangbang, C. Danpeng, W. Nan, Z. Guofeng, and B. Hujun. Survey and Evaluation of Monocular Visual-Inertial Slam Algorithms for Augmented Reality. *Virtual Reality & Intelligent Hardware*, 1(4):386–410, Aug. 2019. doi: 10.1016/j.vrih.2019.07.002

[24] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert. iSAM2: Incremental smoothing and mapping using the bayes tree. *The International Journal of Robotics Research*, 31(2):216–235, 2011. doi: 10.1177/0278364911430419

[25] F. Kobayashi, K. Kitabayashi, K. Shimizu, H. Nakamoto, and F. Kojima. Human motion caption with vision and inertial sensors for hand/arm robot teleoperation. *International Journal of Applied Electromagnetics and Mechanics*, 52(3-4):1629–1636, 2016. doi: 10.3233/JAE-162140

[26] Y. Li, D. Weng, D. Li, and Y. Wang. A Low-Cost Drift-Free Optical-Inertial Hybrid Motion Capture System for High-Precision Human Pose Detection. In *2019 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, pp. 75–80. IEEE, oct 2019. doi: 10.1109/ISMAR-Adjunct.2019.00034

[27] S. Liu, J. Zhang, Y. Zhang, and R. Zhu. A wearable motion capture device able to detect dynamic motion of human limbs. *Nature Communications*, 11(5), 2020. doi: 10.1038/s41467-020-19424-2

[28] I. H. Lopez-Nava and A. Munoz-Melendez. Wearable Inertial Sensors for Human Motion Analysis: A Review. *IEEE Sensors Journal*, 16(22):7821–7834, nov 2016. doi: 10.1109/JSEN.2016.2609392

[29] T. Lupton and S. Sukkarieh. Efficient integration of inertial observations into visual SLAM without initialization. *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009. doi: 10.1109/iros.2009.5354267

[30] T. Lupton and S. Sukkarieh. Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions. *IEEE Transactions on Robotics*, 28(1):61–76, 2012. doi: 10.1109/tro.2011.2170332

[31] J. Marin, T. Blanco, and J. J. Marin. Octopus: A design methodology for motion capture wearables. *Sensors (Switzerland)*, 17(8):1–24, 2017. doi: 10.3390/s17081875

[32] A. Melim. Increasing fidelity with constellation-tracked controllers, 2019 (accessed November, 2020). https://developer.oculus.com/blog/increasing-fidelity-with-constellation-tracked-controllers/.

[33] M. Miezal, B. Taetz, and G. Bleser. On inertial body tracking in the presence of model calibration errors. *Sensors (Switzerland)*, 16(7):1–34, 2016. doi: 10.3390/s16071132

[34] K. Patel and W. Stuerzlinger. Simulation of a virtual reality tracking system. In *International Conference on Virtual Environments, Human-Computer Interfaces and Measurement Systems Proceedings*, VECIMS '11, pp. 78–83. IEEE, Sep 2011. doi: 10.1109/VECIMS.2011.6053849

[35] K. Rangarajan and M. Shah. Establishing motion correspondence. In *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 54, pp. 103–108. IEEE Comput. Sco. Press, 1991. doi: 10.1109/CVPR.1991.139669

[36] M. Shimizu and M. Okutomi. Precise sub-pixel estimation on area-based matching. *Proceedings of the IEEE International Conference on Computer Vision*, 1:90–97, 2001. doi: 10.1109/iccv.2001.937503

[37] P. Tissainayagam and D. Suter. Assessing the performance of corner detectors for point feature tracking applications. *Image and Vision Computing*, 22(8):663–679, 2004. doi: 10.1016/j.imavis.2004.02.001

[38] D. Titterton and J. Weston. *Strapdown inertial navigation technology*. Institution of Electrical Engineers, 2004.

[39] I. Weygers, M. Kok, H. De Vroey, T. Verbeerst, M. Versteyhe, H. Hallez, and K. Claeys. Drift-Free Inertial Sensor-Based Joint Kinematics for Long-Term Arbitrary Movements. *IEEE Sensors Journal*,

20(14):7969–7979, 2020. doi: 10.1109/JSEN.2020.2982459

[40] S. Xia, L. Su, X. Fei, and H. Wang. Toward accurate real-time marker labeling for live optical motion capture. *Visual Computer*, 33(6-8):993–1003, 2017. doi: 10.1007/s00371-017-1400-y

[41] R. S. Zeynalov, A. A. Yakubenko, and A. S. Konushin. The matching of infrared markers for tracking objects using stereo pairs. *Pattern Recognition and Image Analysis*, 23(4):468–473, dec 2013. doi: 10.1134/S1054661813040196

[42] S. Zihajehzadeh, P. K. Yoon, B. S. Kang, and E. J. Park. UWB-Aided Inertial Motion Capture for Lower Body 3-D Dynamic Activity and Trajectory Tracking. *IEEE Transactions on Instrumentation and Measurement*, 64(12):3577–3587, 2015. doi: 10.1109/TIM.2015.2459532