DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Robotics, Cognition, Intelligence

Event-based Non-Rigid 3D Tracking

Yuxuan Xue

DEPARTMENT OF INFORMATICS

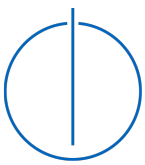
TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Robotics, Cognition, Intelligence

Event-based Non-Rigid 3D Tracking

Ereignis-basierte Nichtstarre 3D Verfolgung

Author:	Yuxuan Xue
Supervisor:	Prof. Dr. Stefan Leutenegger
Advisor:	Dr. Jörg Stückler, Haolong Li, M.Sc
Submission Date:	15.06.2022



I confirm that this master's thesis in robotics, cognition, intelligence is my own work and I have documented all sources and material used.

Munich, 15.06.2022 Yuxuan Xue

Acknowledgments

I would like to express my gratitude to my advisors, Dr. Jörg Stückler and Haolong Li, who offered me the attractive topic at MPI-IS and guided me thoroughly throughout this project. I would also like to thank my supervisor at TUM, Prof. Stefan Leutenegger, for actively having the discussion and giving valuable feedback. I would also like to thank my family and friends who supported me and offered deep insight into the study.

Abstract

Event-based cameras have received much attention in recent years due to their bio-inspired properties (e.g. high temporal resolution, high dynamic range, etc.). However, few researchers have addressed the problem of non-rigid object tracking using event-based cameras. Thus there remains a need for an approach that can reconstruct the deformation of objects using an asynchronous event stream.

In this project, we first extend existing event stream simulators to generate more realistic event data more efficiently. Afterward, we report that the event data can be classified into contour events and texture events. We use the dot product between the event bearing vector and the mesh face normal to model the contour probability and distinguish the contour events and texture events. For contour events, we consider a novel system of associating the event to corresponding mesh faces using the expectation maximization algorithm. For texture events, we use the contrast maximization algorithm. We combine the two tracking frameworks to address the individual limitations.

The experiments and results demonstrate that the proposed contour tracking part can reconstruct the deformation of texture-less objects (e.g. hand, arm, etc.), because the generated events are mostly contour events. The combined tracking framework can perform the rigid reconstruction limited to 2D motion from contour events and texture events.

The expectation maximization contour tracking approach is our main contribution to the scientific community. This approach can perform the non-rigid motion reconstruction using contour events. This can be applied to reconstruct the motion of human arm and hand using an asynchronous event stream, when the initial template is known and the global rotation and translation of the body or hand are fixed. Besides, our proposed contrast maximization texture tracking approach provides a new perspective on event-based planar 2D motion reconstruction.

Contents

1. Introduction

1.1. Motivation

The capturing and the reconstruction of real-world environments, which enables **VR** (**V**irtual **R**eality) / **AR** (**A**ugmented **R**eality) applications, is one of the most important research fields in computer vision. Non-rigid dynamic objects are essential and frequently appearing targets in both computer vision and robotic tasks. However, the tracking and reconstruction of non-rigid objects is a challenging problem due to their complex geometric shapes and deformable surfaces.

Recently, several methods of non-rigid tracking and reconstruction from RGB(-D) have been established. However, the RGB-based methods are far from optimal when objects have extremely fast motion or have the motion in high dynamic range (HDR) scenes. When the objects have fast motion or deformation, the captured images are likely to be blurred. The existing non-rigid tracking methods perform poorly on blurred images. Furthermore, these methods cannot work when the light conditions are limited, e.g., in dark scenes. Thus, there remains a need for an efficient method to perform non-rigid tracking with event-based cameras. Event-based cameras neither suffer from the image blur, caused by the fast motion, nor the high dynamic range scenario. They have the potential to outperform conventional RGB cameras in non-rigid tracking tasks in dark scenes. Nevertheless, till date, only few researchers have addressed the problem of non-rigid tracking with event-based cameras. Therefore, 3D tracking of non-rigidly deforming objects from event-based cameras is an emerging research field.

Event-based cameras have received much attention in recent years due to their bio-inspired properties. Unlike the conventional cameras - that capture images at a fixed rate, event-based cameras asynchronously measure per-pixel brightness change, and output a stream of events that encode the spatio-temporal coordinates of the brightness change and their polarity. Event-based cameras offer a considerable number of advantages in computer vision and robotic tasks over conventional cameras, such as latency in the order of microseconds, a very high dynamic range, and very low power consumption [12]. Additionally, event-based cameras capture per-pixel data independently. These sensors do not suffer from motion blur. Although event-based cameras have tremendous advantages, they do not capture intensity frames like standard cameras. For this reason, the existing computer vision algorithms are not applicable to

events. Over the past five years, a great number of event-based approaches were carried out for applications such as feature detection and tracking, optical flow estimation, simultaneous localization and mapping (SLAM), visual-inertial odometry (VIO), image reconstruction, etc [12]. Nevertheless, there are few methods that have been proposed for the 3D non-rigid reconstruction with event-based cameras [26, 35].

1.2. Problem Statement and Contributions

In this project, we propose and implement a novel method that performs the 3D template-based non-rigid object tracking with event-based cameras. As a template-based approach, the initial 3D shape of objects and their projection on the image plane are assumed to be known. The optimization-based method iteratively updates the 3D geometry of objects until the deformed mesh correctly describes the corresponding events. In our project, we report that the events of objects can be classified into texture events and contour events. For texture-less objects, most events are generated by the motion of boundary edges. Thus, we propose a method that can track the deformation using contour events. Besides, we propose a method to deal with texture events. To the best of our knowledge, none of the previous event-based non-rigid tracking works [26, 35] distinguish texture events and contour events. Thus, they do not reconstruct the deformation using individual events. Hence our work can probably bring a novel contribution to the scientific community.

1.3. Outline

An intuitive sequential structure of the whole thesis can be found in figure 1.1. We also provide a summary of each chapter here:

The first chapter gives a brief introduction to the motivation of this work. After that, the objective and structure of the thesis are presented.

The second chapter begins with the state-of-the-art literature in event-based computer vision relevant to the objective of the thesis. Then, several non-rigid tracking and reconstruction methods based on different data modalities are presented. At the end of this chapter, we present two most relevant works to our method, which reconstruct non-rigid motion of hands using event-based cameras.

Next, the background tools deployed in our work are introduced. We present the object models that we used as templates to reconstruct the rigid and non-rigid motion. Then, several techniques related to our tracking method are introduced. Two basic algorithms are shown at the end of this chapter, which are essential in our work.

Explaining the principle and the function of our event stream simulator are a part of the fourth chapter. It explains explicitly how synthetic events are generated. Besides, we compare our simulator thoroughly with other state-of-the-art event stream simulators, and explain why our simulator is more efficient and user-friendly.

We describe our tracking framework in the fifth chapter. We propose non-rigid motion reconstruction methods and explain the principle and limitation of them in detail in the chapter.

The experiments and results in our work can be found in chapter six. We evaluate our tracking framework quantitatively on synthetic data and qualitatively on real data. Our analysis appears at the end of this chapter.

Finally, we summarize our work and give an overview of the potential future works in the last chapter.

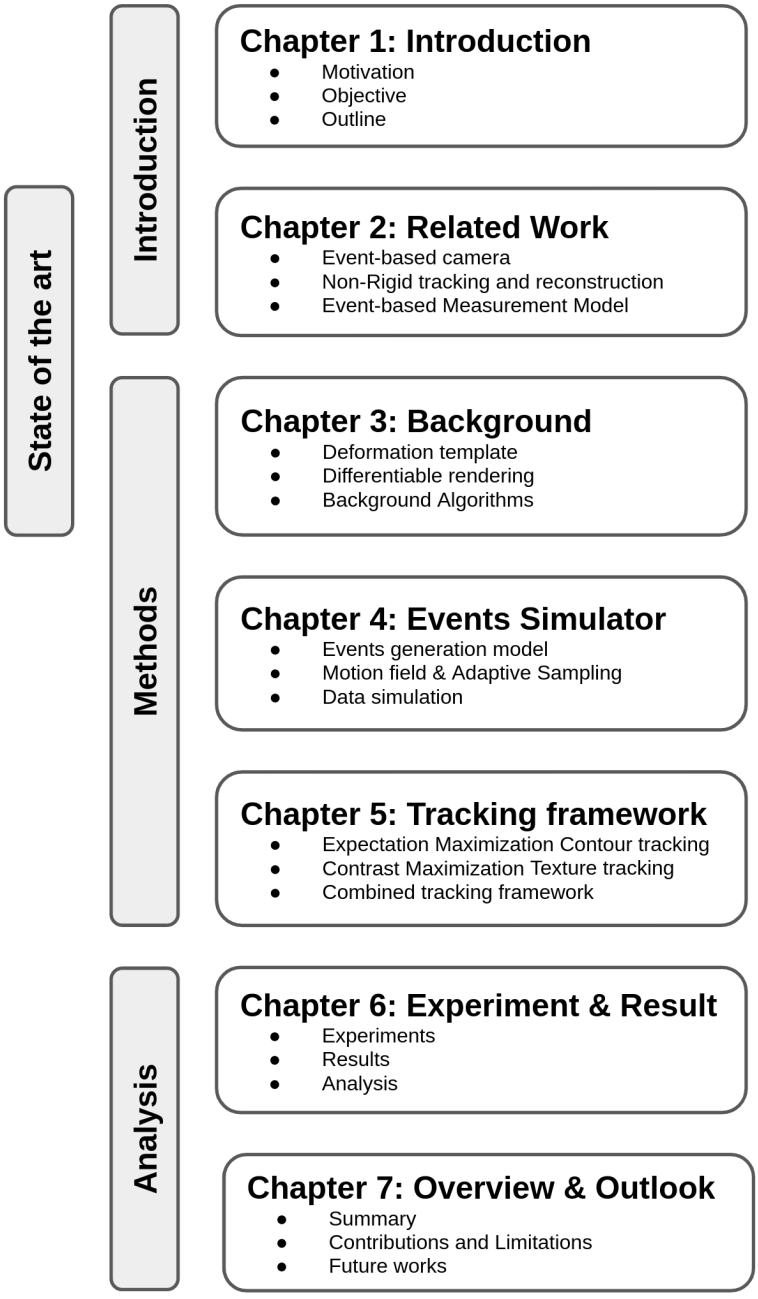


Figure 1.1.: Structure of the master thesis.

2. Related Work

2.1. Event-based Cameras

Event-based cameras arose from the discipline of neuromorphic engineering, which tries to solve a complex problem: figuring out how the brain works and then replicating it on a chip [12]. As a bio-inspired sensor, event-based cameras capture the logarithmic pixel-wise brightness change asynchronously, like how the retina works in our brain. Thus, the outputs of traditional cameras and event-based cameras are completely different: traditional cameras acquire the visual information of a scene as a stream of intensity frames at a constant rate, while event-based cameras have no notion of images since each pixel operates independently. Consequently, in a static scene with no illumination changes, a traditional camera outputs the same image on each frame. By contrast, an event-based camera produces no output at the same scene. figure 2.1 explains the output of event-based cameras intuitively: when the dot is rotating, a space-time spike event stream is captured. The event stream consists of multiple events encoded with $\{x, y, t, pol\}$, to indicate the pixel location, the timestamp, and the polarity of each event.

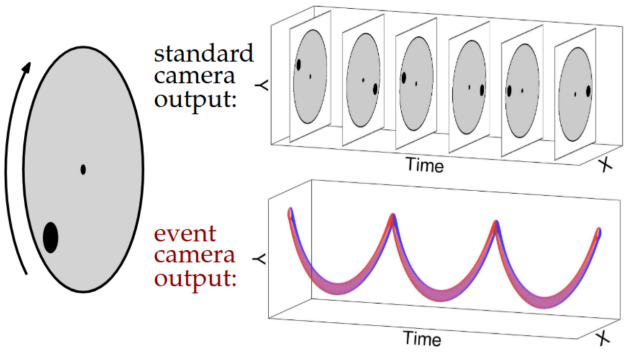


Figure 2.1.: Comparison of output between conventional cameras and event-based cameras. When the dot is rotating, conventional cameras have images at constant frame rate as output, while event-based cameras have continuous event stream as output. Image from [33].

Event-based cameras provide significant advantages over conventional cameras:

- *High temporal resolution:* Event-based cameras are data-driven sensors, where output depends on the amount of motion or brightness change in the scene. Events are timestamped with microsecond resolution (1 MHz) and are transmitted with sub-millisecond latency, which make these sensors react quickly to visual stimuli [12]. As comparison, conventional cameras usually have frequency of 30 - 60 HZ. High temporal resolution ensures that event-based cameras do not suffer from motion blur while capturing fast motion in the scenes.
- *High dynamic range:* Event-based cameras have a very high dynamic range (> 120 dB), which exceeds the 60 dB of high-quality conventional cameras. The photoreceptors of pixels operate in logarithmic scale. Therefore, event-based cameras can adapt to very dark as well as very bright stimuli, like biological retinas.
- *Low power and storage consumption:* event-based cameras only capture pixel-level brightness change, which means it avoids redundant data. Power and storage are only used to process pixel brightness changes.

2.2. Event-based Computer Vision

Since event-based cameras do not have images as output, the existing computer vision algorithms for conventional cameras cannot be directly applied to event-based cameras. Recently, a huge amount of computer vision tasks based on event cameras are solved. In addition, several event-based frameworks are proposed for computer vision tasks. Section 2.2.1 gives a brief overview of existing event-based frameworks, while several popular event-based computer vision tasks are introduced in section 2.2.2.

2.2.1. Event-based Measurement Models

Event Generative Model

Bryner et al. [8] proposed the linearized event generative model, which can predict the brightness change of a tiny time window using the image gradient and the optical flow:

$$\Delta\mathcal{L}\left(x;t\right) \simeq -\left\langle \nabla\mathcal{L}\left(x;t\right), \mathcal{V}\left(x;t\right)\right\rangle \Delta t. \tag{2.1}$$

It is derived from the brightness constancy assumption [39], and the derivation can be found in equations from 4.2 to 4.6.

The error function based on event generative model can be formulated as

$$\min_{X} \|\Delta L(\mathbf{u}) - \Delta \hat{L}(\mathbf{u}; X)\|^2_{\mathcal{L}^2(\Omega)}, \tag{2.2}$$

where $\Delta L(\mathbf{u})$ denotes the measured brightness change, which is accumulated from captured events. In addition, $\mathbf{u}$ denotes the pixel location, and Ω represents the image domain. $\Delta \hat{L}(\mathbf{u}; X)$ is the predicted brightness change according to the event generative model. In Bryner's approach [8], X is the camera pose which calculates the image gradient and optical flow. The error function is minimized over X to solve for the camera pose which generate the captured events.

In addition to the above-mentioned camera ego-motion tracking method [8], Li and Stückler [21] deployed the event generative model in 6-DoF object pose tracking. The performance proves that the event generative model generalizes well to other computer vision tasks.

It is worth mentioning that the image gradient is indispensable in event generative model: besides asynchronous event stream, an aligned intensity image frame is always required. Most dynamic vision sensors from iniVation [1] provide both events stream and gray-scale images. However, the event generative model is not applicable to event-based cameras like Prophesee cameras [2], which only output high-resolution events but no intensity frames. Thus the event generative model is not used in our approach, because we want to reconstruct the deformation only based on a pure event stream.

Contrast Maximization Principle

Gallego et al. [14] proposed a unifying contrast maximization framework for asynchronous event stream. The core idea of this framework is a general objective function for event data, which has the potential to provide the self-supervision of goal tasks. In recent years, it is widely used in self-supervised event-based computer vision tasks, such as optical flow tracking and camera ego-motion estimation [14, 43].

Recently, several studies on the sharpness quality of IWE have been published. Gallego et al. [13] discovered and compared more than 20 different focus loss functions, and concluded that variance is a computational efficient but not accurate reward function. Stoffregen and Kleeman proposed reward function **SoSA** (**S**um **o**f **S**uppressed **A**ccumulations) [38], which follows sparsity rewarding and is more robust to noise.

The contrast maximization principle inspires a part of the motion reconstruction method in our project as well. A detailed explanation of unifying contrast maximization framework can be found in section 3.6, while the contrast maximization based non-rigid objects tracking method we proposed is introduced in section 5.4.

[1] https://inivation.com/
[2] https://www.prophesee.ai/

2.2.2. Event-based Tasks

Optical Flow Estimation

Optical flow estimation using event-based cameras approximates a continuously time-varying velocity field in image coordinates. Bardow, Davidson, and Leutenegger [2] jointly reconstructed intensity images and estimated flow based on events by minimizing their objective function. However, the accuracy of their approach heavily relies on the quality of the reconstructed image. Gallego et al. [14] and Zhu et al. [43] proposed unsupervised optical flow estimation based on the contrast maximization principle. Gehrig et al. [15] proposed E-RAFT to estimate the dense optical flow from events with a fully-supervised learning method.

Visual Odometry / SLAM

Taking into account above-mentioned advantages of low latency, low power consumption, and high dynamic range, event-based cameras are suitable to be deployed on robotics platform for real-time SLAM applications. Kim, Leutenegger, and Davidson [17] simultaneously reconstructed the scene and tracked the 6-DoF camera pose using Kalman filters. Rebeq et al. [32] fused event stream and IMU measurements to perform visual-inertial odometry. Rosinol et al. [40] combined event stream, intensity images, and IMU measurement to perform SLAM and to control a quadrotor. The result shows that the SLAM system using event-based cameras outperforms conventional systems in HDR and high-speed scenarios.

Object Pose Estimation

Unlike two previous event-based computer vision tasks, which have been explored and well studied, object pose estimation using event-based cameras is an emerging research field. Li and Stückler [21] proposed a novel approach that tracks the 3D motion of objects in a combined way from measurements of event and frame-based cameras. It deployed the event generative framework introduced in 2.2.1.

2.3. Non-Rigid Tracking and Reconstruction

Methods for 3D reconstruction of non-rigid objects can be categorized into **SfT** (**S**hape **f**rom **T**emplate) and **NRSfM** (**N**on-**R**igid **S**tructure **f**rom **M**otion). SfT methods reconstruct the deformed shape of an object from a single image and the object's textured 3D model. The 3D shape of the object, so-called template, is assumed to be known in advance. The SfT methods can be categorized into analytical and energy-based

methods. Among analytical methods, the focus is on the isometric deformation, which implies that the geodesic distance between any two points on the surface remains constant, or the conformal deformation where angles are preserved on the surface of the shape [4]. Energy-based methods [27] jointly minimize the shape energy and the reprojection error obtained from the image correspondences. These optimization methods are appropriate to handle sequential data association with robust kernels to deal with outliers. NRSfM methods reconstruct the non-rigid surface of the objects and the corresponding camera poses from monocular image sequences using multi-frame 2D correspondences computed among the input views. In NRSfM as [28], the rigidity constraint of the normal SfM method is replaced by constraints on the object's deformation model.

Lamarca et al. [20] proposed DefSLAM, which is a parallel algorithm composed of a front-end SfT deformation tracking thread running at a higher frame rate and a back-end NRSfM to compute SfT template running at a slower frame rate. The DefSLAM estimates the camera pose and the deformation of the scene simultaneously. While exploring unseen areas, the DefSLAM estimates a template for the new zone. Besides, it periodically re-estimates the template to adapt it better to the observed scene. It is worth mentioning that the DefSLAM performs in deforming scenes in real-time.

Recently, several neural non-rigid tracking and reconstruction methods have been proposed. Sidhu et al. [37] formulated a fully differentiable dense neural NRSfM approach with an auto-decoder-based deformation model based on monocular RGB input. Božic et al. [7] proposed an end-to-end learnable, differentiable RGB-D based non-rigid tracker in a self-supervised manner. The non-rigid tracker deployed a correspondence prediction network to predict the dense correspondence from the source to the target image frame. A differentiable optimizer solves for the deformation parameter according to the pixel correspondences. However, none of these neural-network-based methods perform non-rigid tracking in real-time.

2.4. Event-based Non-Rigid Tracking

2.4.1. Event Simulation Framework

Nehvi et al. [26] proposed a template-based non-rigid tracking framework using the event generative model (Sec. 2.2.1). Given the known previous hand parameter θ^{t-1} and the desired hand parameter θ^{t}, two images can be rendered. The thresholding function $g(x)$ can generate an event for each pixel according to the difference of two images in a smooth and differentiable manner. The generated event stream are compared with captured events stream. The difference between the input event stream and the generated event stream is penalized to optimize the desired hand parameter. The

gradient is propogated back to the pose parameter θ^t using differentiable rendering. As shown in figure 2.2, the generated event stream and input event stream are stacked to frame, which allows the pixel-wise comparison.

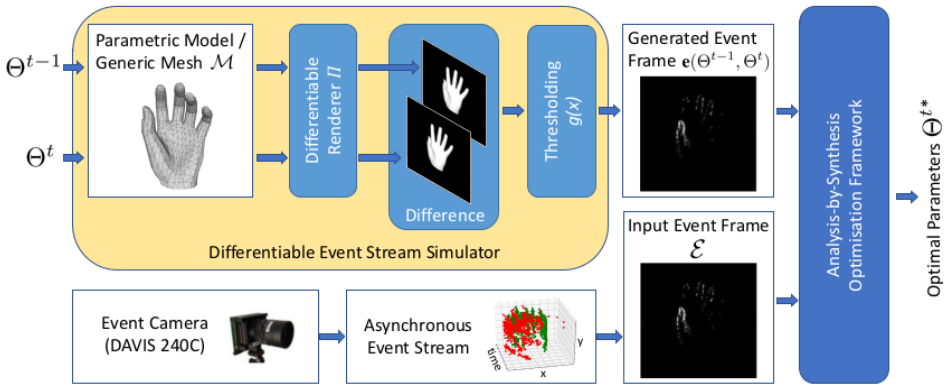


Figure 2.2.: Overview of Nehvi's non-rigid objects tracking framework. It uses a differentiable event stream simulator to generate events and compares with the captured events. The difference is minimized to optimize the objects pose θ. Image from [26].

The bottleneck of Nehvi's method [26] is the quality of the generated event. As illustrated in figure 2.2, the generated events are calculated from two rendered images. As introduced in section 3.2, the rendering process takes lights and the texture map of the mesh as inputs. However, these two properties are difficult to adjust for scenarios where events are captured. Besides, the simulator renders images with a black background and generates events according to that feature. The most essential drawback is, when the capturing scenario doesn't have the pure black background, the captured events are not the same as the generated events, which leads to non-robustness in the tracking. Unlike the Nehvi's approach [26] which only uses plain 2D event information, our method introduced in Chapter 5 relies on the 3D geometry which is inferred from events. Thus, it doesn't have above-mentioned limitations.

2.4.2. Learning-based Approach

Rudnev et al. [35] proposed a fully supervised deep learning framework for events-based hand tracking. Since it is difficult to obtain the ground-truth hand parameter label, they used a synthetic event stream to train the neural network model. The process is shown in figure 2.3: the input event stream is represented with LNES (**L**ocally-**N**ormalised **E**vent **S**urfaces), which encodes all events within a fixed time

window as an image $I \in \mathbb{R}^{W \times H \times 2}$. A ResNet-18 network is trained with the event input representation I to regress the hand parameter θ. A constant-velocity Kalman filter is deployed on the raw network output to perform the temporal filtering and to ensure the smoothness of the tracking.

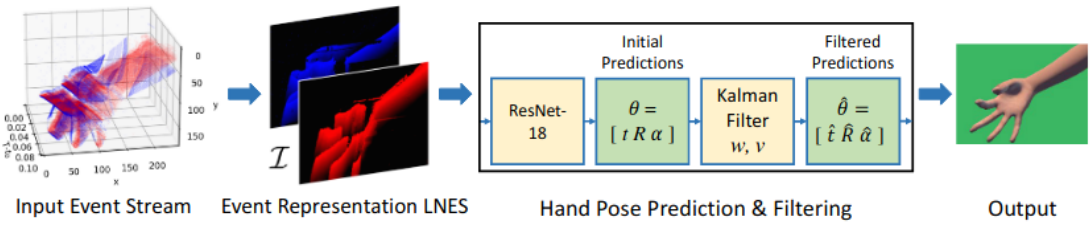


Figure 2.3.: Overview of Rudnev's *EventHand* framework. Asynchronous events stream is represented using LNES and fed into the trained hand pose prediction network to predict hand pose parameters. Image from [35].

The main limitation of Rudnev's method [35] is that the network can only be trained on synthetic data. Additionally, it is only applicable for MANO-based (Sec. 3.1.1) hand tracking. Thus, it does not generalize to other non-rigid object tracking. In contrast, our method presented in chapter 5 has a well-defined objective function. As an optimization-based method, it does not require the training on the synthetic data and is not only restricted to the MANO hand model [34].

2.5. Event Representation

As introduced in section 2.1, an event stream contains multiple asynchronous events encoded with $\{x, y, t, pol\}$. Usually, several events in a spatio-temporal window are stacked to formulate event images, which could be used as inputs of neural networks [35, 43] or reduce the computational complexity [32, 40].

Recently, several event representations are proposed. Rudnev et al. [35] presented LNES, which encodes asynchronous events within a fixed time window as an two channel image. Positive events and Negative events are recorded in separate channels so that the polarity of events are preserved. Zhu et al. [43] discretizied the time domain to represent events as a three-dimensional volume. The evens are inserted into the volume using a linearly weighted accumulation. Rebeq et al. [32] and Vidal et al. [40] stacked the fixed number of events into each spatio-temporal window. The temporal size of each window is inversely proportional to the event rate. to the event rate. In our work, we stacked the fixed number of events into each spatio-temporal window too.

3. Background

This chapter introduces the methods, tools, and algorithms which are used in the project. The 3D object template model is presented in section 3.1. Section 3.2 introduces how the gradient is propagated in rendering and section 3.3 explains the hyperparameter tunning framework we deployed. Besides, the computational efficiency is important in both data simulator and non-rigid motion reconstruction processes. The tools which are deployed to increase computational efficiency are introduced in section 3.4. Section 3.5 presents the analytical geometry background which is essential in our work. In section 3.6, the contrast maximization principle for event data is explained. At the end of the chapter, a brief introduction of the expectation maximization algorithm is given, which is an essential machine learning algorithm. Both contrast maximization and expectation maximization principles are elaborated in chapter 5.

3.1. Non-Rigid Object Model

As a template-based non-rigid method, the template model is one of the most essential component. In our project, we performed the deformable motion reconstruction on both parametric 3D human body model, SMPL-X, and general triangle mesh models.

SMPL (A **S**kinned **M**ulti-**P**erson **L**inear Model) [25] is a parametric human body model that realistically represents a wide range of human body shapes and can be posed with natural pose-dependent deformations. The template body mesh model has $n = 6890$ vertices and $k = 23$ joints. Shape parameter $\boldsymbol{\beta}$ controls the shape of the body. Pose parameter $\boldsymbol{\theta}$ is defined by 72 parameters, which denote the axis-angle representation for all joints and the root orientation.

Figure 3.1 describes intuitively the SMPL model expression. Begin with a canonical body model $\overline{\mathbf{T}}$, adding the shape blend shapes and pose blend shapes controlled by the the shape and pose parameters can express arbitrary reasonable human body.

MANO (hand **M**odel with **A**rticulated and **N**on-rigid def**O**rmations) [34] is a differentiable hand model that can map the hand pose parameter and shape parameter into a 3D hand mesh. In MANO, the pose and shape are defined as the linear combination

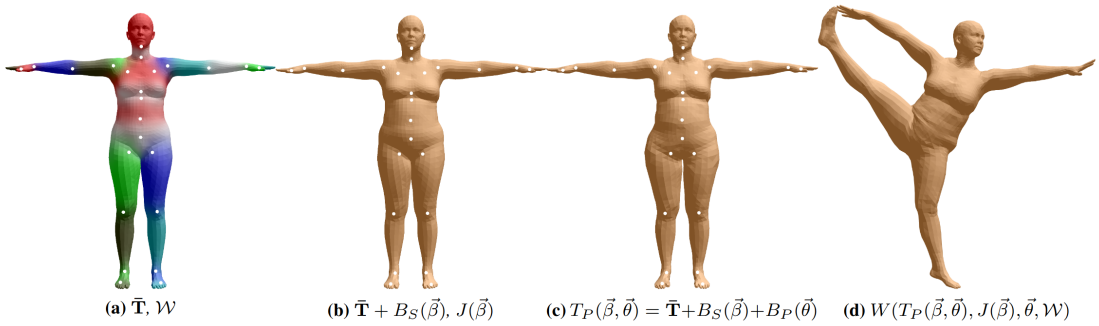


Figure 3.1.: Template, shape blend space, and pose blend space of SMPL model. Image from [25].

of a set of vertex offsets. The shape blend parameter is computed from a set of registered hand shaped, and normalized to the zero pose using **PCA** (**P**rinciple **C**omponent **A**nalysis). Each hand posture is parameterized by a set of principle components coefficients that map a differentiable low-dimensional manifold. The effect of the first ten principle components in the pose space are shown intuitively in figure 3.2.

In MANO, the hand surface is represented by a manifold triangle mesh $M \equiv (V, F)$ with $n = 778$ vertices $V = \left\{v_i \in \mathbb{R}^3 \mid 1 \leq i \leq n\right\}$ and 1538 faces F. The face F indicates the connection of the vertices in the hand surface, where the face topology is considered to be fixed. Given the mesh topology, a set of $k = 15$ joints $J = \left\{j_i \in \mathbb{R}^3 \mid 1 \leq i \leq k\right\}$ can be directly inferred from the hand mesh.

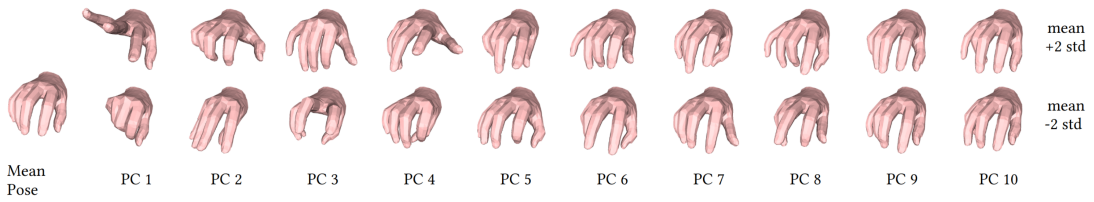


Figure 3.2.: PCA pose space. The left-most image presents the mean pose. The effect of the first ten principle components are shown in the rest of the columns. Image from [34].

FLAME (**F**aces **L**earned with an **A**rticulated **M**odel and **E**xpressions) [22] is a parametric head model. In SMPL-X, the shape of the head is defined in the joint shape parameter $\boldsymbol{\beta}$. We use facial expression parameters $\boldsymbol{\psi}$ to present the rich facial expression.

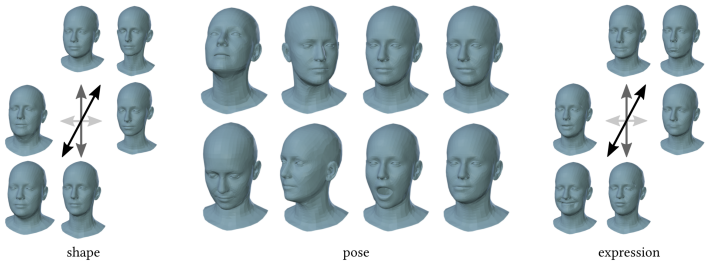


Figure 3.3.: Shape, pose, and expression of FLAME head model. Image from [22].

3.1.1. SMPL-X Human Body Model

SMPL-X (**SMPL** e**X**pressive) [29] is a 3D model of the human body that extends SMPL body model with the fully articulated hands and an expressive face. As a parametric human body model, SMPL-X has enormous essential advantages: compatibility with graphics software, simple parametrization, small size, efficiency, differentiable, etc. SMPL-X is the combination of the SMPL, the MANO, and the FLAME model, which are introduced in following paragraphs.

SMPL-X uses standard vertex-based linear blend skinning with learned corrective blend shapes, has $n = 10,475$ vertices and $k = 54$ joints. SMPL-X includes joints for the neck, jaw, eyeballs and fingers, etc. SMPL-X is parametrized by pose parameter $\boldsymbol{\theta}$, shape parameter $\boldsymbol{\beta}$, and facial expression parameter $\boldsymbol{\psi}$. The pose parameter $\boldsymbol{\theta}$ consists of the $\boldsymbol{\theta}_f$ for jaw joints, the $\boldsymbol{\theta}_h$ for finger joints, and the $\boldsymbol{\theta}_b$ for the remaining body joints. The shape parameter $\boldsymbol{\beta}$ controls the shape of the body, the face, and both hands. The facial expressions parameter $\boldsymbol{\psi}$ expresses the different facial expression. As illustrated in figure 3.4, the body, face, and hands parameters in pose and shape space can model the human accurately.

3.1.2. General Mesh Model

Two common file formats for storing single meshes are `.obj` and `.ply` files. The `obj` file has a standard way to store extra information about a mesh. In PyTorch3D [30], an `obj` file can be extracted to variables `verts`, `faces`, and `aux`. The `verts` indicates the 3D coordinates of vertices, represented with a `(V,3)`-tensor. The `faces.verts_idx` is an `(F,3)`-tensor of the vertex-indices of the corners of the faces. `V` and `F` denote the number of vertices and faces, respectively. The `aux` is an object contains normals, uv coordinates, material colors and textures if they are available in the loaded `obj` file. Using the extracted vertices, faces, and textures information, a textured mesh can be generated and used in PyTorch3D.

Figure 3.4.: SMPL-X can jointly model the human body, face, and hands. Image from [29].

YCB Benchmarks (**Y**ale-**C**MU-**B**erkeley Benchmarks) is an object and model set which consists of objects of daily life with different shapes, sizes, textures, weight and rigidity [9]. The set consists of 77 objects divided into 5 categories: Food items, kitchen items, tool items, shape items, and task items. An overview of all objects can be found in figure 3.5. For each object, RGB-D images, high-resolution RGB images, segmentation masks, calibration information, and textured 3D mesh models are available. Thus, we choose objects in YCB Benchmarks as the general mesh model in our project.

Figure 3.5.: YCB Benchmarks - object and model set. Image from [9].

3.1.3. Mesh Model Representation

A polygon mesh model consists of geometric vertices and face elements. Usually, mesh objects are saved in `.obj` file, which is a geometry definition file format first developed by Wavefront Technologies for its Advanced Visualizer animation package. In this project, the canonical SMPL-X model and YCB Benchmarks models are both provided in `.obj` format.

A `.obj` file may contain geometric vertices, vertex normals, and polygonal faces. For the texture of the model, texture coordinates, materials, and the corresponding texture map are required. A textured mesh can be constructed from the given geometric information and the corresponding texture data.

Figure 3.6 intuitively illustrates the process of loading a mesh model from the `.obj` file and the texture map. The `.obj` provides geometric vertices `v`, faces information `f`, and texture coordinates `vt` of the SMPL-X model in canonical shape and pose. Vertices `v = [x, y, z]` provide the location of a vertex. Faces `f = [v1/v2, v2/v3, v1/v3]` indicate the three edges which determine the face, while each edge is determined by two vertices. With the vertices location and faces information, a geometrical mesh model can be constructed as in the left figure in 3.6. Texture coordinates `vt = [u, v]` provide the UV coordinate of each vertex. The RGB value of each vertex is determined by its UV coordinate on the corresponding texture map. A texture map for SMPL-X model is shown in the right figure in 3.6. Examples of textured mesh models can be found in Appendix B.1, and B.3.

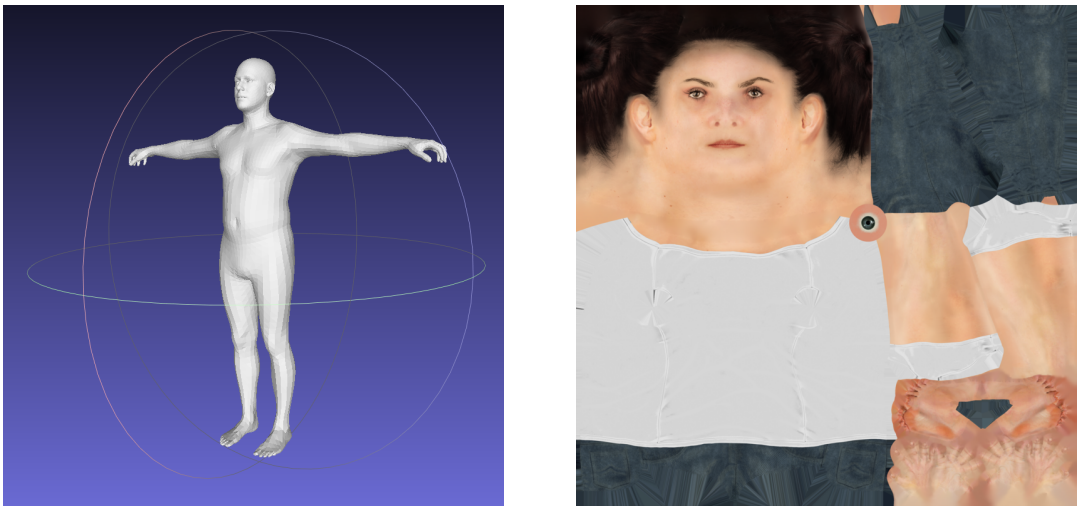


Figure 3.6.: SMPL-X geometric mesh model (left) and a texture map in the corresponding UV coordinate (right). Image of the texture map from [11].

3.2. Differentiable Renderer

Rendering is a core part of computer graphics that converts 3D models into 2D images. It's a natural way to bridge the gap between 3D scene properties and the pixels of a 2D image. Traditional rendering engines are not differentiable. For this reason, they can't be incorporated into deep learning pipelines. Recently, Liu et al. has shown how to build a differentiable renderer that integrates with deep learning [24].

In this project, we use PyTorch3D, which is an efficient, modular differentiable renderer. As a differentiable renderer, it automatically accumulates gradients to all inputs. The following figure describes all the components of the rendering pipeline.

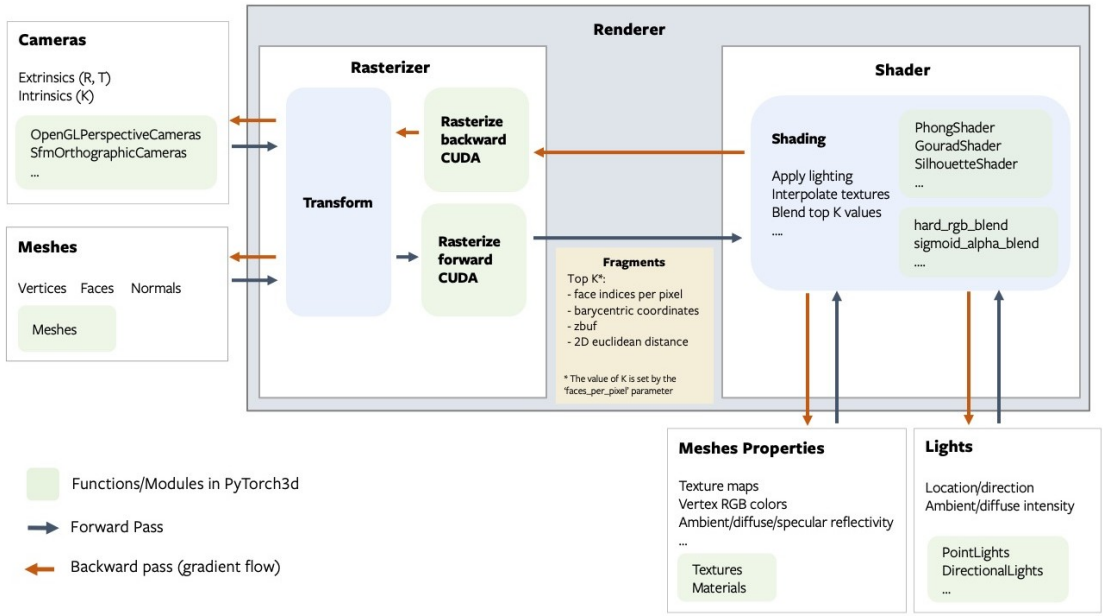


Figure 3.7.: Architecture of a PyTorch3D renderer. Image from [30].

A renderer in PyTorch3D takes the camera, the mesh, and the light as input, can generate a rendered image which contains the gradient to all inputs as output. As illustrated in figure 3.7, a renderer in PyTorch3D is composed of a rasterizer and a shader. The rasterizer is thoroughly explained in section 3.2.1, while the shader is introduced in section 3.2.2.

3.2.1. Rasterizer

Rasterization is the task of taking an image described in a vector graphics format and converting it into a raster image [39]. Compared to ray tracing, rasterization is a extremely fast process of computing the mapping from scene geometry to pixels. However, it does not prescribe a particular way to compute the color of those pixels. Figure 3.8 shows how PyTorch3D rasterizes a polygon face of a polygon mesh to pixels on 2D image plane.

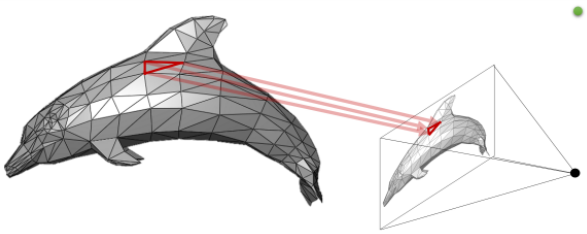


Figure 3.8.: Rasterize a triangle of a polygon mesh. Image from [30].

In PyTorch3d, the rasterizer takes mesh model as input, produces 4 outputs of desired image size:

- `pix_to_face`: gives the indices of the nearest faces at each pixel.
- `zbuf`: gives the NDC z-coordinates of the nearest faces at each pixel.
- `barycentric`: gives the barycentric coordinates in NDC units of the nearest faces at each pixel.
- `pix_dists`: gives the signed Euclidean distance (in NDC units) in the x/y plane of each point closest to the pixel.

3.2.2. Shader

As illustrated in figure 3.7, a shader takes outputs of rastarization, mesh texture properties, and lights as input, applies shading and outputs a rendered image of the mesh. Phong shader is the most commonly used shader which applies per pixel shading [41]. It first interpolates the vertex normals and vertex coordinates using the barycentric coordinates to get the position and normal at each pixel. Then, it computes the illumination for each pixel. The pixel color is obtained by multiplying the pixel textures by the ambient and diffuse illumination and adding the specular component.

3.2.3. Gradient Flow Map

Thanks to the differentiable parametric mesh model, rasterizer, and shader, the gradient of outputs w.r.t. all desired inputs are automatically accumulated and available. In our non-rigid hand tracking task, the gradient w.r.t. pose parameter, global rotation, and global translation is available.

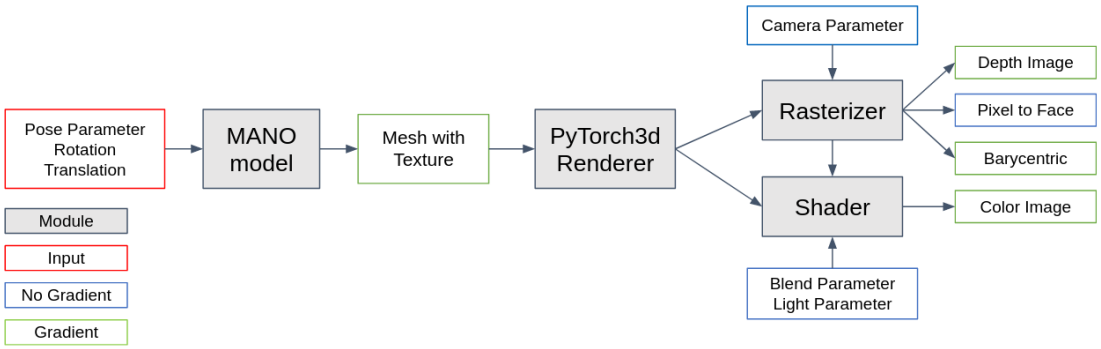


Figure 3.9.: Gradient Map from input MANO parameters to renderer outputs.

As illustrated in figure 3.9, gradient is first propagated over the MANO model, which means all mesh vertices contain gradient w.r.t. input hand parameters. PyTorch3D renderer takes hand mesh as input, and generates 4 outputs. The gradient of renderer outputs w.r.t. input hand parameters θ can be determined using chain rule:

$$\frac{\partial \text{ output}}{\partial \theta} = \frac{\partial \text{ output}}{\partial \text{ mesh}} \cdot \frac{\partial \text{ mesh}}{\partial \theta}. \tag{3.1}$$

It is worth mentioning that the output `pixel_to_face` contains integer indices, so it can't have gradients. Besides, all outputs have automatic gradient w.r.t. inputs, which requires no manual gradient calculation and therefore makes optimization simplified.

3.3. Hyperparameter Tuning

As an optimization-based tracking framework, finding the optimal hyperparameters is a key to the accurate result. In our project, we deployed optuna [1] to tune hyperparameters in the tracking framework. Optuna implements sampling algorithms such as Tree-Structured of Parzen Estimator[5]. It can determine which hyperparameter values to try next based on a historical record of trials. We present how optuna is used in our work in section A.1.

3.4. Computational Efficiency

In this project, we constantly look for ways to accelerate the computationally intensive tasks, such as the event simulator and the non-rigid object tracking. Most of our tasks involve image processing, which means that we are particularly interested in anything that makes matrix computations — sometimes over fairly large tensors easier and faster. We used two methods to increase the computational efficiency, namely JIT-compilation in section 3.4.1 and parallel programming using GPGPU in section 3.4.2.

3.4.1. JIT Compilation

Since Python is commonly used in deep learning and differentiable rendering, we use Python as the main programming language in our project. However, Python by nature is not a language with a compiler. The primary factor in Python's short-comings is usually its speed when working with large amounts of data and recursively training machine-learning models.

JIT (**J**ust-**I**n-**T**ime) compilation is a compiler feature that allows a language to be interpreted and compiled during runtime, rather than execution. That means a JIT compiler will be compiling the language as it is executing the logic before the code is compiled.

Numba [19] is an open-source JIT compiler that uses the standard **LLVM** (**L**ow **L**evel **V**irtual **M**achine). It translates a subset of Python and NumPy code into fast machine code: first analyses Python code, then turns it into an LLVM intermediate representation, and finally creates bytecode for the selected architecture. In our project, we implemented several frequently-called functions with Numba to increase efficiency.

3.4.2. Parallel Programming

CUDA (**C**ompute **U**nified **D**evice **A**rchitecture) is a parallel computing platform and API that allows software to use certain types of GPU for general purpose processing with an approach called **GPGPU** (**G**eneral **P**urpose computation on **G**raphics **P**rocessing **U**nit). CUDA is a software layer that gives direct access to the GPU's virtual instruction set and parallel computational elements, for the execution of computing kernels.

Our project is mainly implemented in PyTorch, which uses GPU to accelerate the process. All PyTorch tensor operations are available on CPU (implemented in C++) and on GPU (implemented in CUDA). In our project, all events are processed processed parallely with the implementation in CUDA operation with PyTorch.

3.5. Analytic Geometry

Analytic geometry plays an essential role in our project. This section provides the background knowledge of analytic geometry, which is applied in the tracking algorithms in chapter 5.

3.5.1. Distance between Line and Point

The smallest distance between a line and a point in 3D can be considered as the problem of finding the projection of the point on that line. The projection can be computed using the dot product.

The direction of vector **s** of the line spanned by points **a1a2** is the difference of **a1** and **a2**, divided by their length,

$$\mathbf{s} = \frac{(\mathbf{a2} - \mathbf{a1})}{||\mathbf{a2} - \mathbf{a1}||_{L2}}. \tag{3.2}$$

A vector **q** from point **b1** to **a1** is determined by

$$\mathbf{q} = \mathbf{b1} - \mathbf{a1} \tag{3.3}$$

The projection **p** of point **b1** on line **a1a2** can be calculated from the dot product between the vector **a1b1** and vector **a1a2**,

$$\mathbf{p} = \mathbf{a1} + (\mathbf{q}^\top \mathbf{s}) \cdot \mathbf{s}. \tag{3.4}$$

Finally, the minimal distance between the point and line is

$$d = ||\mathbf{p} - \mathbf{b1}||_{L2} \tag{3.5}$$

3.5.2. Distance between Line and Line Segments

As illustrated in the right figure in 3.10, there are two cases of the minimal distance between two lines in 3D: the closest point is inside of the line segment (as in **b1b2**) and the closest point is outside of the line segment (**c1c2**). Considering our method, namely calculating the minimal distance between an unprojecting ray and an edge of a triangle in 3D, we only perform the clamping for line segments (**b1b2** and **c1c2**, considered as mesh face edges) but not for the line (**a1a2**, considered as a bearing vector).

The vector of line **a1a2** and line segment **b1b2** can be given as

$$\mathbf{s} = \frac{(\mathbf{a2} - \mathbf{a1})}{||\mathbf{a2} - \mathbf{a1}||_{L2}}, \mathbf{q} = \frac{(\mathbf{b2} - \mathbf{b1})}{||\mathbf{b2} - \mathbf{b1}||_{L2}}. \tag{3.6}$$

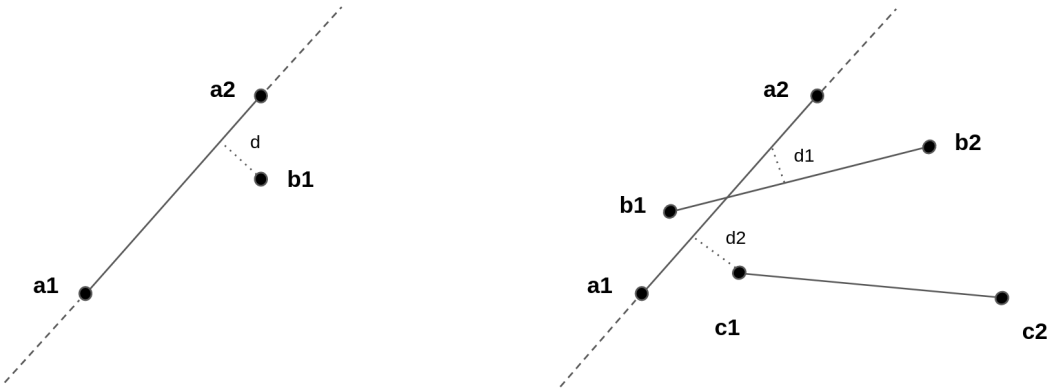


Figure 3.10.: Point-Line-Distance (left) and Line-Line-Segments-Distance (right).

The normal of the spanned surface by two lines can be represented with the cross product,

$$\mathbf{v} = \mathbf{s} \times \mathbf{q}. \tag{3.7}$$

If two lines are parallel, the norm of cross product $\mathbf{v}$ is 0. If two lines don't overlap, there is a closest point solution. Otherwise, there are infinite closest positions having the same minimal distance. In the case of two lines are parallel, the distance between them is

$$d = ||((\mathbf{s}^\top(\mathbf{b1} - \mathbf{a1})) \cdot \mathbf{s} + \mathbf{a1}) - \mathbf{b1}||_{L2} \tag{3.8}$$

If two lines criss-cross as **a1a2** and **b1b2** in figure 3.10, the problem turns into the calculation of the projected closest points of one line on the other line. The projected closest point on segment A is

$$\mathbf{p_a} = \mathbf{a1} + \left(\mathbf{s} \cdot \frac{\left| \begin{matrix} \mathbf{b1} - \mathbf{a1} \\ \mathbf{q} \\ \mathbf{v} \end{matrix} \right|}{||\mathbf{v}||_{L2}} \right), \tag{3.9}$$

and the projected closest point on segment B is

$$\mathbf{p_b} = \mathbf{b1} + \left(\mathbf{q} \cdot \frac{\begin{vmatrix} \mathbf{b1} - \mathbf{a1} \\ \mathbf{s} \\ \vec{\mathbf{v}} \end{vmatrix}}{||\mathbf{v}||_{L2}} \right). \tag{3.10}$$

After having two closest projection points and considering all clamping cases, the minimal distance between **a1a2** and **b1b2** is

$$d = ||\mathbf{p_a} - \mathbf{p_b}||_{L2}. \tag{3.11}$$

3.5.3. Intersection between Line and Plane

As in figure 3.11, given a line spanned by two points **a1** and **a2**, and a plane spanned by three points **b1**, **b2**, and **b3**, we can determine whether and where does the line intersect the plane in 3D.

The vector of edge **b1b2** and edge **b2b3** can be given as

$$\mathbf{s} = \frac{(\mathbf{b2} - \mathbf{b1})}{||\mathbf{b2} - \mathbf{b1}||_{L2}}, \mathbf{q} = \frac{(\mathbf{b3} - \mathbf{b2})}{||\mathbf{b3} - \mathbf{b2}||_{L2}}. \tag{3.12}$$

Again, the normal of the plane is the cross product of two vectors spanning the plane,

$$\mathbf{v} = \mathbf{s} \times \mathbf{q}. \tag{3.13}$$

The normalized direction vector of the line is

$$\mathbf{r} = \frac{\mathbf{a2} - \mathbf{a1}}{||\mathbf{a2} - \mathbf{a1}||_{L2}} \tag{3.14}$$

If the plane and the line are parallel, namely the dot product between the plane normal and the line is close to 0, there won't be any intersection. Otherwise, we can calculate intersection **p** with

$$\mathbf{p} = \mathbf{a1} + \frac{\mathbf{v}^\top \mathbf{a1} + \mathbf{v}^\top \mathbf{b1}}{\mathbf{v}^\top \mathbf{r}} \cdot \mathbf{r} \tag{3.15}$$

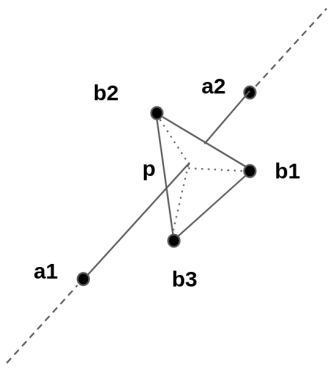


Figure 3.11.: Intersection between a line and a plane in 3D (left) and its barycentric coordinate (right).

3.5.4. Barycentric Coordinate

Having a 3D planar point **p** and 3 non-collinear points **b1**, **b2**, and **b3** on the plane, we can calculate the corresponding barycentric coordinate.

The vectors of edges of the triangle are

$$\begin{aligned} \mathbf{s} &= \mathbf{b2} - \mathbf{b1}, \\ \mathbf{q} &= \mathbf{b3} - \mathbf{b1}, \\ \mathbf{r} &= \mathbf{p} - \mathbf{b1}, \end{aligned} \tag{3.16}$$

respectively.

The barycentric coordinate of plane point P with respect to triangle (**b1**, **b2**, **b3**) is given in v, w, u, with

$$\begin{aligned} v &= \frac{(\mathbf{q}^\top \mathbf{q}) \cdot (\mathbf{r}^\top \mathbf{s}) - (\mathbf{s}^\top \mathbf{q}) \cdot (\mathbf{r}^\top \mathbf{q})}{(\mathbf{s}^\top \mathbf{s}) \cdot (\mathbf{q}^\top \mathbf{q}) - (\mathbf{s}^\top \mathbf{q}) \cdot (\mathbf{s}^\top \mathbf{s})}, \\ w &= \frac{(\mathbf{s}^\top \mathbf{s}) \cdot (\mathbf{r}^\top \mathbf{q}) - (\mathbf{s}^\top \mathbf{q}) \cdot (\mathbf{r}^\top \mathbf{s})}{(\mathbf{s}^\top \mathbf{s}) \cdot (\mathbf{q}^\top \mathbf{q}) - (\mathbf{s}^\top \mathbf{q}) \cdot (\mathbf{s}^\top \mathbf{q})}, \\ u &= 1.0 - v - w. \end{aligned} \tag{3.17}$$

Using the barycentric coordinate, point P can be accurately represented by positions of triangle corners as

$$P = v \cdot \mathbf{b1} + w \cdot \mathbf{b2} + u \cdot \mathbf{b3} \tag{3.18}$$

3.6. Contrast Maximization

Gallego et al. [14] proposed that the problem of extracting information from events can be interpreted as the data association process, i.e., establishing which events were triggered by the same scene edge. Since moving edges define point trajectories (Fig. 5.7) on the image plane, the corresponding events should be triggered along these point trajectories. As illustrated in figure 3.12, events caused by the same moving edge pattern can be drawn on a point trajectory.

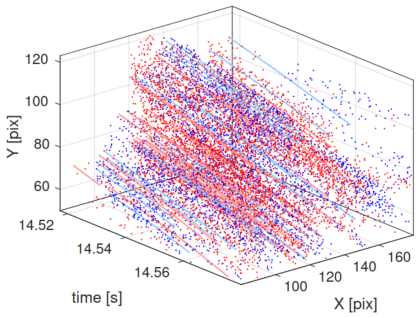


Figure 3.12.: Point trajectory of events. Image from [14].

The core idea of the contrast maximization framework is to warp all events along the point trajectory to a reference frame. Ideally, the events on the same point trajectory will be warped back to one point. Thus, the reference frame, also called **IWE** (**I**mage of **W**arped **E**vents), will have the highest sharpness when the point trajectory is estimated. Practically, point trajectories are parametrized by the desired motion parameter θ. The variance of the IWE describes the sharpness. A unifying contrast maximization framework can be formulated as

$$\max_{\theta} f(\theta) = \max_{\theta} \sigma^2(H(\mathbf{x};\theta)) \doteq \max_{\theta} \frac{1}{N_p} \sum_{i,j} \left(h_{ij} - \mu_H\right)^2, \tag{3.19}$$

where $H(\mathbf{x};\theta)$ denotes the IWE, which is parameterized by desired motion parameter θ. h_{ij} denotes the value at pixel ij on the IWE. $\sigma^2(\cdot)$ is the operation that calculates the variance. The image variance (contrast) of warped events measures how well events agree with the candidate point trajectories. It should be maximized to solve for the parameters.

According to [14], an additional benefit of the optimization-based motion compensation framework is that it produces motion-corrected event images. These motion-corrected event images are more appropriate than original event images as input to complex processing algorithms [14] such as visual-inertial data fusion, object recognition,

etc. An intuitive comparison between an input event frame and its motion-corrected frame can be found in figure 3.13: the input event frame is blurred and has low contrast. Conversely, its motion-corrected frame is sharp and has high contrast, which is more suitable for further event processing tasks.

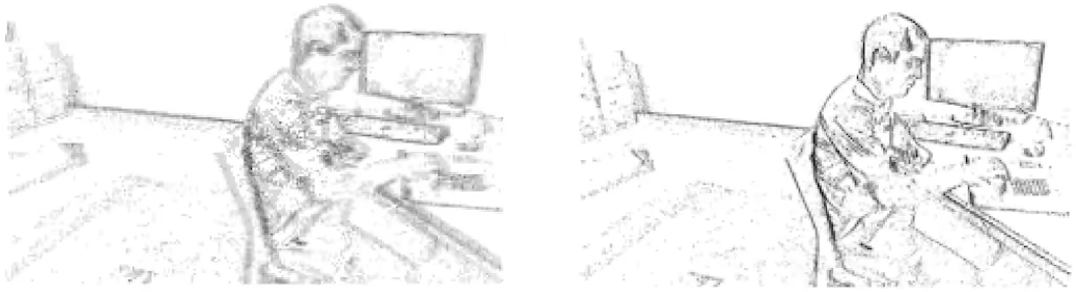


Figure 3.13.: Comparison between an input event frame (left) and its motion-corrected frame (right). Images from [14].

3.7. Expectation Maximization

Probabilistic models can have full observable variables, but can also contain latent variables, which are unknown. When a probabilistic model contains latent variables, the naive MLE (**M**aximum **L**ikelihood **E**stimation) cannot be directly deployed to estimate the model parameter.

The goal of the EM (**E**xpectation **M**aximization) algorithm is to find maximum likelihood solutions for models with latent variables [6]. The log likelihood function is stated in equation 3.20:

$$\ln P(\mathbf{X} \mid \boldsymbol{\theta}) = \ln \left\{ \sum_{\mathbf{Z}} P(\mathbf{X}, \mathbf{Z} \mid \boldsymbol{\theta}) \right\}, \tag{3.20}$$

where the set of all observable data is denoted by $\mathbf{X}$, the set of all latent variables is denoted by $\mathbf{Z}$, and the set of all model parameters is denoted by θ. For each observation in $\mathbf{X}$, the corresponding value of the latent variable $\mathbf{Z}$ is considered. In other words, the actual observed data $\mathbf{X}$ is incomplete, and introducing the latent variable $\mathbf{Z}$ makes the data set $\{\mathbf{X}, \mathbf{Z}\}$ complete.

However, the complete data set $\{\mathbf{X}, \mathbf{Z}\}$ are usually unknown, only the incomplete data $\mathbf{X}$ is given. The knowledge of latent variables $\mathbf{Z}$ is only given by the posterior distribution $P(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta})$. Because the complete-data log likelihood is not available, it is

alternatively treated as the expected value under the posterior distribution of the latent variable. The expectation of the complete-data log likelihood is written as:

$$\mathcal{Q}\left(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}\right) = \sum_{\mathbf{Z}} P\left(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta}^{\text{old}}\right) \ln P(\mathbf{X}, \mathbf{Z} \mid \boldsymbol{\theta}), \tag{3.21}$$

where we introduce a distribution $q(\mathbf{Z})$ defined over the latent variable. The following decomposition holds independently from the choice of $q(\mathbf{Z})$:

$$\ln p(\mathbf{X} \mid \boldsymbol{\theta}) = \mathcal{L}(q, \boldsymbol{\theta}) + \text{KL}(q \| p), \tag{3.22}$$

where $\mathcal{L}(q, \boldsymbol{\theta})$ and $\text{KL}(q \| p)$ are defined as

$$\mathcal{L}(q, \boldsymbol{\theta}) = \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{X}, \mathbf{Z} \mid \boldsymbol{\theta})}{q(\mathbf{Z})} \right\}, \tag{3.23}$$

$$\text{KL}(q \| p) = -\sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z} \mid \mathbf{X}\boldsymbol{\theta})}{q(\mathbf{Z})} \right\}. \tag{3.24}$$

By substituting $q(\mathbf{Z}) = p\left(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta}^{\text{old}}\right)$ into Equation 3.23, the $\mathcal{L}(q, \boldsymbol{\theta})$ has the form

$$\begin{aligned} \mathcal{L}(q, \boldsymbol{\theta}) &= \sum_{\mathbf{Z}} p\left(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta}^{\text{old}}\right) \ln p(\mathbf{X}, \mathbf{Z} \mid \boldsymbol{\theta}) - \sum_{\mathbf{Z}} p\left(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta}^{\text{old}}\right) \ln p\left(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta}^{\text{old}}\right) \\ &= \mathcal{Q}\left(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}\right) + \text{ const }. \end{aligned} \tag{3.25}$$

The expectation, denoted by $\mathcal{Q}\left(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}\right)$, is maximized to revise the parameter estimate $\boldsymbol{\theta}^{\text{new}}$. This corresponds to the M step of EM algorithm, and can be written as:

$$\boldsymbol{\theta}^{\text{new}} = \arg\max_{\theta} \mathcal{Q}\left(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}\right). \tag{3.26}$$

After updating the model parameter $\boldsymbol{\theta}^{\text{new}}$, if the log likelihood or the parameter values doesn't converge, the estimated model parameter $\boldsymbol{\theta}^{\text{new}}$ is used as the $\boldsymbol{\theta}^{\text{old}}$ to calculate the expectation in equation 3.21. Several pairs of E and M steps are performed until the convergence of the log likelihood or model parameters.

4. Non-Rigid Event Simulator

Generally, event-based cameras are scarce and expensive to get, which increases the difficulty to create event datasets. Besides, it is also strenuous to acquire the groudtruth of the non-rigid deforming object. Therefore, we developed an accurate, efficient event stream simulator, which generates synthetic event data from deforming objects. In addition to the events, the developed simulator is able to provide RGB images, depth images, ground-truth motion vectors, and surface normal images given a sequence of pose parameters.

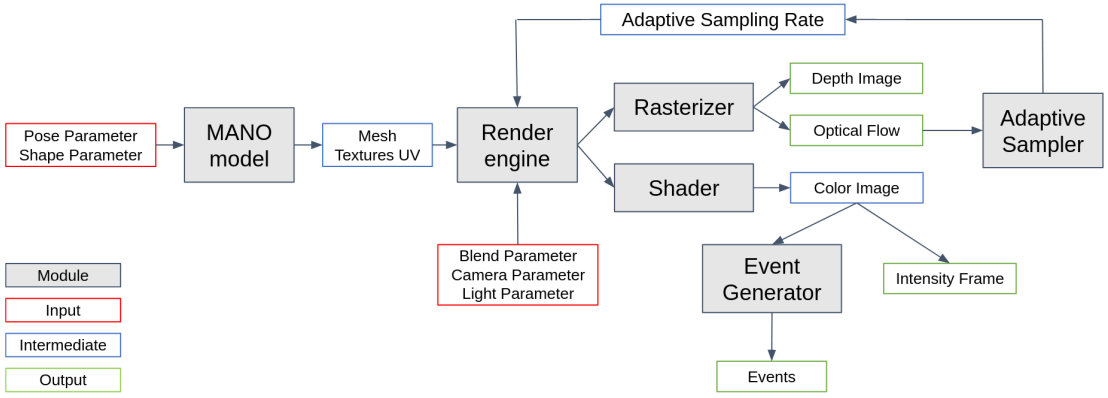


Figure 4.1.: Event Simulator Architecture. The modules are in gray, the inputs are in red, and the outputs are in green boxes.

The structure of the event simulator is given in figure 4.1. The MANO model takes the pose and shape parameters as input, and generates triangle hand mesh with textures of given parameters. The render engine renders the textured mesh into depth, motion field, and color images. The Adaptive sampler calculates the adaptive sampling rate, which models the essential asynchronous property of the event-based camera. The event generator takes consecutive intensity images and calculates synthetic events according to the event generation model introduced in section 4.1.

Our simulator effectively samples event frames using adaptive sampling strategy (Sec. 4.2). We showed the method to compute the motion field of the object (Sec 4.3), which is required in the adaptive sampling rate calculation. To simulate more realistic

data, we model noise in the event generation process (Sec. 4.4). We show all supported data modalities of our event stream simulator in section 4.5.

Our event simulator extends Nevhi's simulator [26] and ESIM [31]. Our simulator supports more data modalities (Sec. 4.5), and increases efficiency by adaptive sampling of event frames (Sec. 4.2) and parallel programming. We show the comparison with Nehvi's simulator and other event stream simulators in detail in section 4.6.

4.1. Event Generation Model

In contrast to traditional RGB cameras which record the absolute brightness of all pixels on the image plane, event-based cameras capture relative per-pixel brightness change. At a given pixel $\mathbf{x}$, an event is triggered if the relative brightness change exceeds a threshold called contrast sensitivity. Different pixels on the image plane are triggered independently, which outputs an asynchronous stream of events, when

$$|\mathcal{L}(\mathbf{x}, t) - \mathcal{L}(\mathbf{x}, t - \Delta t)| \geq C, \tag{4.1}$$

where $\mathcal{L}$ represents the brightness, $\mathbf{x}$ represents the pixel location, t represents the current timestamp, and C represents the contrast sensitivity of the event-based camera. It is well mentioning that the logarithmic irradiance value is used as the brightness value to model the HDR (High Dynamic Range) ability of the real event-based camera.

An event is usually represented as $e = (x, y, t, p)$, where x and y indicates the pixel coordinates as 2D location on the image plane, t is the timestamp when event occurs, and the discrete variable $p \in \{-1, 1\}$ is the polarity which indicates whether the brightness increases or decreases on the pixel. Usually, a set of events $\mathcal{S} = \{e_i = (x_i, y_i, t_i, p_i)\}$ within a time window T are accumulated into an event buffer image as shown in the right column in figure 4.2.

In the developed simulator, the synthetic events are generated from two consecutive images according to equation 4.1. For each pixel on the image, an event is generated when the brightness change is larger than threshold C. This process is shown intuitively in figure 4.2.

Figure 4.2.: Event Generation Principle. The two intensity images with black background are rendered from the textured mesh. The events image with white background are accumulated from all events in a temporal window between the two rendered image. Positive events are represented in red, while the negative events are represented in blue.

4.2. Adaptive Sampling

An essential advantage of event-based cameras is the continuous representation of the visual signal at every pixel. Unlike the intensity images which have a fixed frame rate and can be sampled synchronously, the events should have an asynchronous fashion. Rebeq et al. [31] proposed the adaptive sampling method in ESIM, which allows the simulator to sample frames adaptively. We take the adaptive sampling method in our non-rigid event simulator, namely adapt the sampling rate based on the predicted dynamics of the visual signal. It is worth mentioning that the predicted dynamics refers to the motion field of each pixel, introduced in section 4.3. Two possible adaptive sampling strategies are introduced in the section. As illustrated in figure 4.1, the calculated sampling rate is used to ask the render engine to render a color image. The events since the last color image are simulated using the event generation principle in figure 4.2.

Nehvi's simulator uses historical information to calculate the adaptive sampling rate, which is inaccurate. In Nevhi's simulator, the adaptive sampling rate is decided by the maximal brightness difference between two recently sampled images. However, the historical information does not describe the current motion accurately. In our simulator, we simulate motion field (Sec. 4.3) which provides the information of current motion. It leads to a more accurate adaptive sample rate.

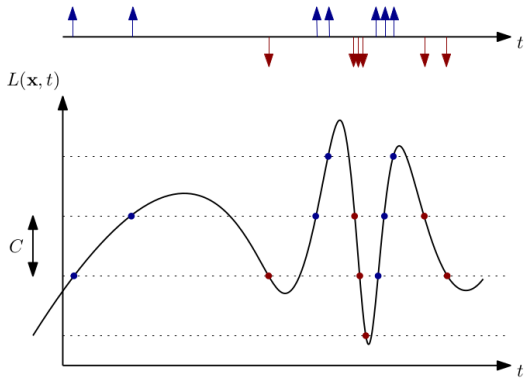


Figure 4.3.: Adaptive Sampling on a single pixel. Positive (red) and negative (blue) events are generated whenever the brightness change larger than the contrast threshold C. The Event rate grows when the brightness change rapidly. Image from [31].

Adaptive Sampling Based on Brightness Change

The idea of adaptive sampling based on brightness change is that the simulator renders when the maximum expected brightness change of arbitrary pixel on the image plane exceeds the contrast threshold C.

Under the Lambertian surfaces assumption, the brightness constancy assumption [39] can be formulated as:

$$\mathcal{L}(x, y, t) = \mathcal{L}(x + \delta x, y + \delta y, t + \delta t) \quad . \tag{4.2}$$

The approximation of first-order Taylor expansion of equation 4.2 yields:

$$\mathcal{L}(x + \delta x, y + \delta y, t + \delta t) \approx \mathcal{L}(x, y, t) + \frac{\partial \mathcal{L}}{\partial x}\delta x + \frac{\partial \mathcal{L}}{\partial y}\delta y + \frac{\partial \mathcal{L}}{\partial t}\delta t. \tag{4.3}$$

After some simple transformation, it can be rewritten as:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial x}\delta x + \frac{\partial \mathcal{L}}{\partial y}\delta y + \frac{\partial \mathcal{L}}{\partial t}\delta t = 0, \\ \frac{\partial \mathcal{L}}{\partial x}u + \frac{\partial \mathcal{L}}{\partial y}v + \frac{\partial \mathcal{L}}{\partial t} = 0. \end{aligned} \tag{4.4}$$

Thus, the original brightness constancy assumption can be represented as:

$$\frac{\partial \mathcal{L}(x;t)}{\partial t} \simeq -\left\langle \nabla \mathcal{L}(x;t), \mathcal{V}(x;t) \right\rangle, \tag{4.5}$$

where the brightness change rate can be represented by the negative dot product of the image gradient $\nabla\mathcal{L}$ and motion vector $\mathcal{V}$ at pixel x. The expected brightness change at pixel x and time t can be approximated by

$$\Delta\mathcal{L}(x;t) \simeq \frac{\partial\mathcal{L}(x;t)}{\partial t}\Delta t, \tag{4.6}$$

where the Δt indicates the given small interval of time. Equation 4.5 and 4.6 formulate the linearlized event generation model [8], which can approximate the expected brightness change during a small time interval from the motion field and the image gradient.

The adaptive next sampling time can be calculated by

$$t_{k+1} = t_k + \lambda_b C \left|\frac{\partial\mathcal{L}}{\partial t}\right|_m^{-1}, \tag{4.7}$$

where $\left|\frac{\partial\mathcal{L}}{\partial t}\right|_m = \max_{x\in\Omega}\left|\frac{\partial\mathcal{L}(x;t_k)}{\partial t}\right|$ indicates the maximum brightness change rate across image plane Ω. $\lambda_b \leq 1$ is a parameter that manually controls the trade-off between the rendering accuracy and efficiency. In our project, we adjust the λ_b by observing the simulated and real event data and close the gap between them.

Adaptive Sampling Based on Pixel Displacement

The idea of adaptive sampling based on pixel displacement is to ensure that the maximum displacement of any pixel on the image plane between consecutive rendered frames is bounded:

$$t_{k+1} = t_k + \lambda_v |\mathcal{V}|_m^{-1}, \tag{4.8}$$

where $|\mathcal{V}|_m = \max_{x\in\Omega}|\mathcal{V}(x;t_k)|$ is the maximum altitude of the motion vector across the image plane at time t_k. As before, $\lambda_v \leq 1$ is used to manually control the render rate and is adjusted using real events data.

4.3. Motion Field

In computer vision, the motion field is an ideal representation of 3D motion as it is projected onto a 2D camera image. Optical flow represents the apparent motion of brightness patterns on a 2D image. Generally, Optical flow is an approximation of the motion field. The difference is small at points with high spatial gradients under some simplifying assumptions. In this project, we estimate the motion field and use it to approximate the optical flow.

Figure 4.4.: A visualization of motion field in hand deformation sequence. Green arrows denotes where 3D points move.

Theoretically, the discrete motion field can be calculated by

$$\begin{pmatrix} u' \\ v' \end{pmatrix} = \frac{\pi\left((x_2, y_2, z_2)^T\right) - \pi\left((x_1, y_1, z_1)^T\right)}{\Delta t}, \tag{4.9}$$

where $(u', v')^T$ is the motion field in image coordinate, π is the projection function related to the camera model, and Δt denotes a small time interval. Spatial points $(x_1, y_1, z_1)^T$ and $(x_2, y_2, z_2)^T$ are the same point on the object before and after deformation in the small time interval, respectively.

Pratically, we can simplify the process because of the known pixel-to-face coordinates and barycentric coordinates from rasterization. The motion field of a pixel can be explained as where the corresponding 3D point of the pixel moves. According to pixel-to-face coordinate, we know the face index of the mesh model, which corresponds to

the pixel. Doing barycentric interpolation using vertex locations before the deformation, we can know the exact 3D point position which projects to the pixel. Obviously, we can use vertex locations of the same face after the deformation to do the barycentric interpolation. It can give us the updated 3D position of the point which projects to that pixel. Then, we can project the interpolated 3D point back to the image plane, and we get the reprojected position of that pixel. The reprojection process of a pixel is given intuitively in figure 4.5.

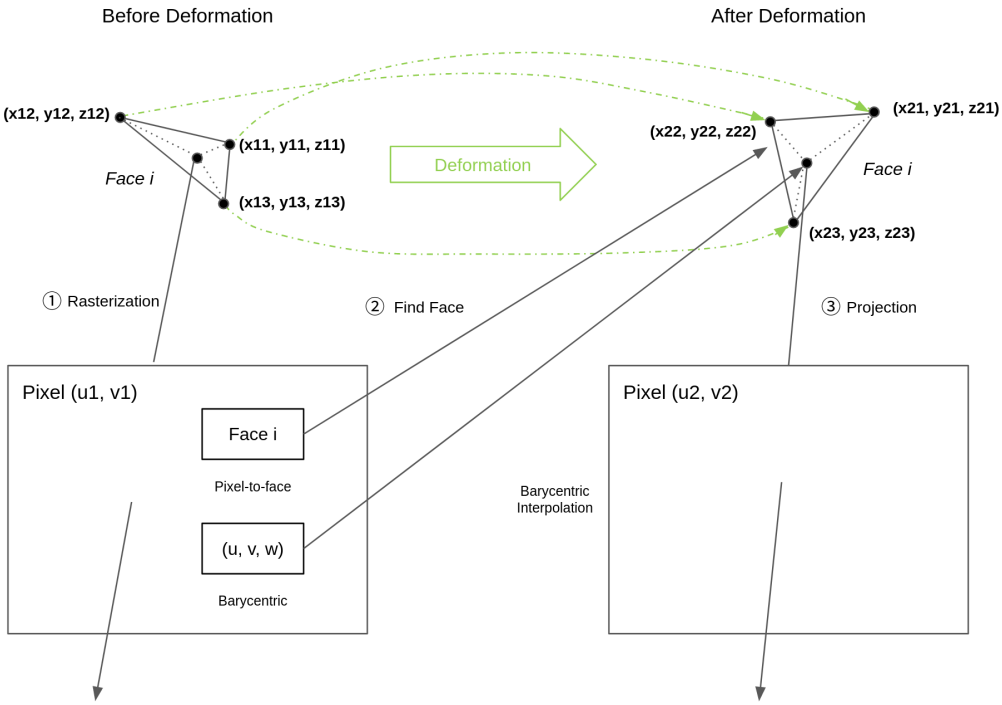


Figure 4.5.: Reprojection of a pixel during the deformation.

4.4. Noise Modelling

Event-based cameras are noisy because of the inherent shot noise in photons and from transistor circuit noise [12]. To close the sim-to-real gap of our simulator, we have noisy modelling methods that the uncertainty and background noise of real event-based cameras.

Uncertainty of Contrast Threshold

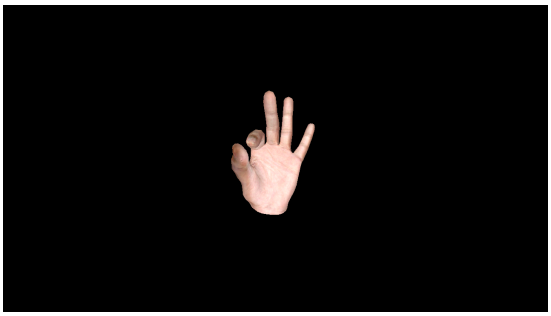
The contrast threshold C of a real event-based camera is determined by the pixel bias currents [12]. Positive and negative events can be triggered according to different corresponding thresholds, namely $C+$ and $C-$. In practice, the contrast threshold C is affected by the noise and the pixel-to-pixel mismatch. Similar to ESIM [31], we model the uncertainty of the contrast threshold by sampling it from a Gaussian distribution at each sampling time. In our simulator, we draw the contrast threshold from $\mathcal{N}(0.5, 0.0004)$.

Salt-and-Pepper Noise

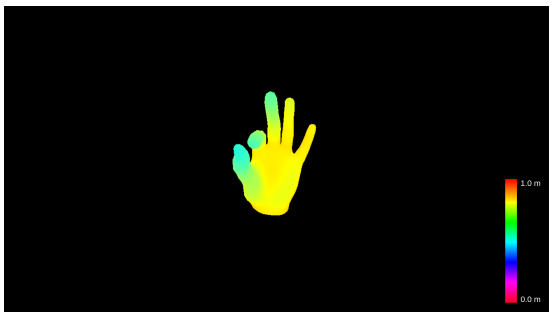
In our observation, the real event-based camera outputs a considerable amount of noisy events on the background, which are not generated by a moving edge. To make the synthetic event data more realistic, we also simulate the salt-and-pepper noise in our simulator. Here, we use a similar method as in EventHands [35]: at each sampling time, we sample a probability from a uniform distribution from $[0, 1]$ for each pixel; If the probability exceeds a predefined threshold, a salt-and-pepper noise event appears on this pixel. Rudnev et al. [35] estimated the noise event rate by placing the event-based camera towards the static background and counting the number of positive and negative recorded events. They reported that the noise consists of $\approx$ 2500 positive and $\approx$ 100 negative events per second. Thus, the threshold is adjusted to 1×10^{-5} for positive noisy events and 4×10^{-7} in our simulator.

4.5. Data Simulation

In this section, we visualize available data modalities of our event simulator, e.g. RGB image, depth image, event stream, motion vector, and surface normal. The color coding of the 2D flow vectors and 3D normal vectors are shown in the lower right corner in figures 4.6e and 4.6f, respectively.



(a) RGB intensity image

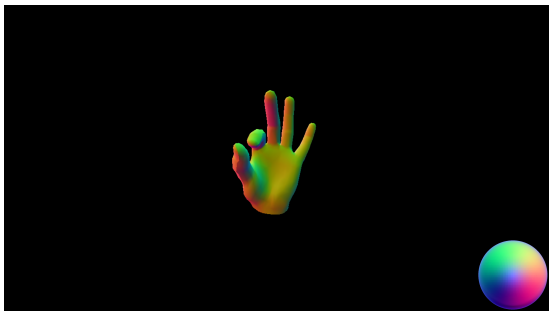


(b) Colorful depth image

(c) Image of events in a spatio-temporal window

(d) Image of accumulated events between consecutive intensity images

(e) Motion field



(f) Surface normal

Figure 4.6.: Synthetic data of our event simulator. Here is an example sequence of the MANO hand motion.

4.6. Comparison of event simulators

We compare our simulator with state-of-the-art rigid motion [31] and non-rigid motion [26, 35] event stream simulators.

ESIM

Rebeq et al. [31] proposed ESIM, which is a popular open-sourced event camera simulator for simulating events from camera ego-motion or object rigid-body transformation. ESIM uses OpenGL as the rendering engine, and is mainly implemented in C++. The architecture of ESIM can be found in figure 4.7.

For the camera ego-motion, ESIM takes the scenario data and a user-defined camera trajectory as input, returns simulated asynchronous events stream, RGB images, depth images, motion field, and IMU data. For multiple object tracking, ESIM takes the scenario, object data, and user-defined trajectories for all objects as input, returns similar outputs as the camera ego-motion simulation without the IMU data.

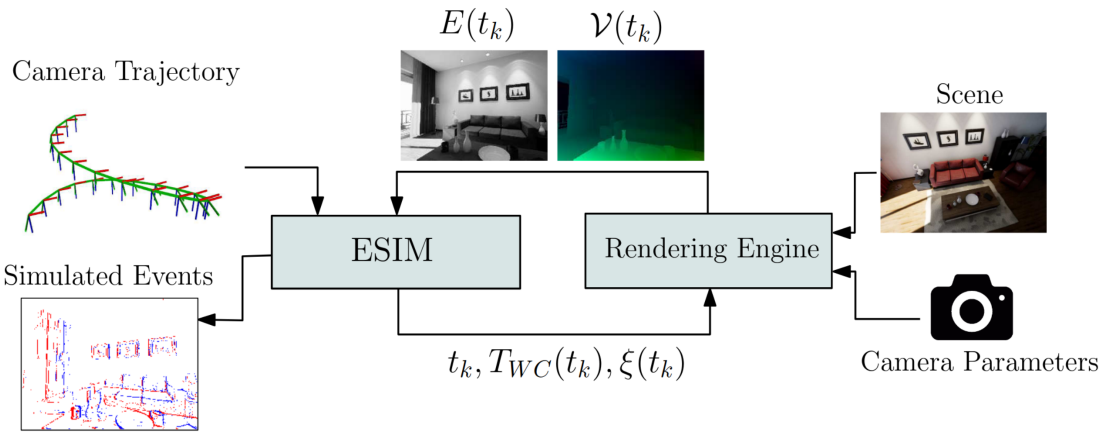


Figure 4.7.: Architecture of ESIM [31]. ESIM samples the camera pose $T_{WC}(t_k)$ and the camera twist $\xi(t_k)$ from the given trajectory, and the renderer returns a intensity image and a motion vector map. The next rendering time t_k is adaptively chosen as described in section 4.2. Image from [31].

ESIM is popular in event-based VIO (**V**isual-**I**nertial-**O**dometry) tasks, since it has perfect simulated IMU data, RGB image, and events stream. ESIM also supports in publishing data as ROS (**R**obot **O**perating **S**ystem) topics, which is also an essential advantage for robotic vision development.

The main drawback of ESIM is that it doesn't support non-rigid objects. Compared to ESIM, our non-rigid event simulator support deforming objects. Besides, our simulator is mainly implemented in Python, which no longer requires us to calculate the gradient manually thanks to the automatic differentiation engine in PyTorch.

Differentiable Event Simulator

Nehvi et al. [26] proposed a differentiable event simulator, which simulates intensity images and events of the motion of MANO [34] hand pose. Thanks to the technology of differentiable rendering introduced in section 3.2, each pixel on the image screen is differentiable with respect to hand pose parameters. Literally, the differentiable event simulator proposed by Nehvi has the similar structure as our simulation framework in figure 4.1. However, there are several limitations of the Nehvi's simulator compared to our simulator.

First of all, Nehvi's simulator has less data modality: it can only output asynchronous events. As an reference, our simulator outputs intensity RGB image, asynchronous events, depth image, surface normal and discrete motion field. We believe the motion field is an essential output of an event simulator, since event-based optical flow estimation is a popular task in event-based computer vision [14] [43].

Secondly, because of the lack of the motion field information, Nehvi's method has a bad adaptive sampling strategy. The sampling time is inverse proportional to the maximal brightness difference between the last two rendered images among all pixels, which is based on the historical motion and cannot react to the next motion. As a reference, our simulator uses the adaptive sampling strategy introduced in section 4.2, which is based on a linearlized event generation model and optical flow. Thanks to the motion field, the simulator can select the sampling time according to the current motion of objects.

Last but not least, the event generation process is not parallel processed. The Nevhi's simulator loops over each pixel and calculates the brightness change, which is extremely inefficient and limits the simulation of large image sizes. As a reference, our simulator processes event generation on each pixel simultaneously using parallel programming introduced in section 3.4.2.

In the simulation, different speed of simulated motion leads to different sampling times because of the adaptive sampling strategy (Sec. 4.2). It takes 47.17 seconds to simulate a one-second synthetic sequence, which contains 30 RGB images, depth images, surface normal images, motion field images, and 220 sampled event frames (Sec. 4.5). As a comparison, the Nehvi's simulator [26], which doesn't use parallel processing operation, takes 3717.56 seconds to simulate a pure events stream of the same sequence. Our simulator is 78-times faster than Nevhi's simulator, which is an

enormous advantage while generating enormous data.

EventHands

Rudnev et al. [35] proposed an event simulator for MANO hand in EventHands. Rudnev's simulator is mainly implemented in C++, which is faster than the implementation in Python due to compilation mechanism. The event generation process is implemented in CUDA, which means all pixels are parallely processed.

However, same as Nevhi's simulator [26], Rudnev's simulator also only has events stream as output. Moreover, because of the lack of motion field, it doesn't have the adaptive sampling strategy. It samples every `0.001` seconds, which is inefficient when the motion is small. Nevertheless, Rudnev's simulator is not implemented in PyTorch. Therefore, it is not differentiable with respect to inputs same as ESIM [31].

Summary

This section compares our simulator with other event simulators in different metrics. Table 4.1 represents the differences between event simulators intuitively. In summary, our simulator supports non-rigid objects, parallel processing in the event generation process, and automatic differentiation, which are advantages over the well-known event simulator, ESIM [31]. In addition to that, our simulator has up to 5 data modalities, and supports adaptive sampling using simulated motion field. These features are not achieved by non-rigid event simulators in [26] [35]. Last but not least, our simulator integrates SMPL-X model [29], which is the only events simulator that simulate events generated by the motion the of body and facial expression.

	Non-Rigid Objects	**Data Modality**	**Adaptive Sampling**	**Parallel Processing**	**Auto Gradient**
Our Simulator	Body, Hand, Expression	5	√	√	√
ESIM [31]	×	5	√	×	×
Nehvi's Simulator [26]	Hand	1	×	×	√
EventHands [35]	Hand	1	×	√	×

Table 4.1.: Comparison between our event simulator and other event simulators in different metrics.

5. Event-based Non-Rigid Tracking

This chapter explains our proposed methods and frameworks in the non-rigid object tracking task. We propose that events can be distinguished as contour events and texture events (Sec. 5.1). Then, we propose an expectation maximization contour tracking approach for contour events (Sec. 5.3) and a contrast maximization texture tracking approach for texture events (Sec. 5.4). We explain the advantage and the limitation of each approach in their sections. Then, we introduce an event-based non-rigid tracking framework in 5.5, which combines advantages and addresses issues of CM-texture-tracker and EM-contour-tracker. At the end of this chapter, we introduce the sliding window optimization applied in our tracking framework.

5.1. Contour and Texture Events

In our project, we first proposed that events can be classified into contour events and texture events. Contour event means the event is generated by the motion of contour boundary of objects and texture event means the event are generated by the motion of texture at each pixel.

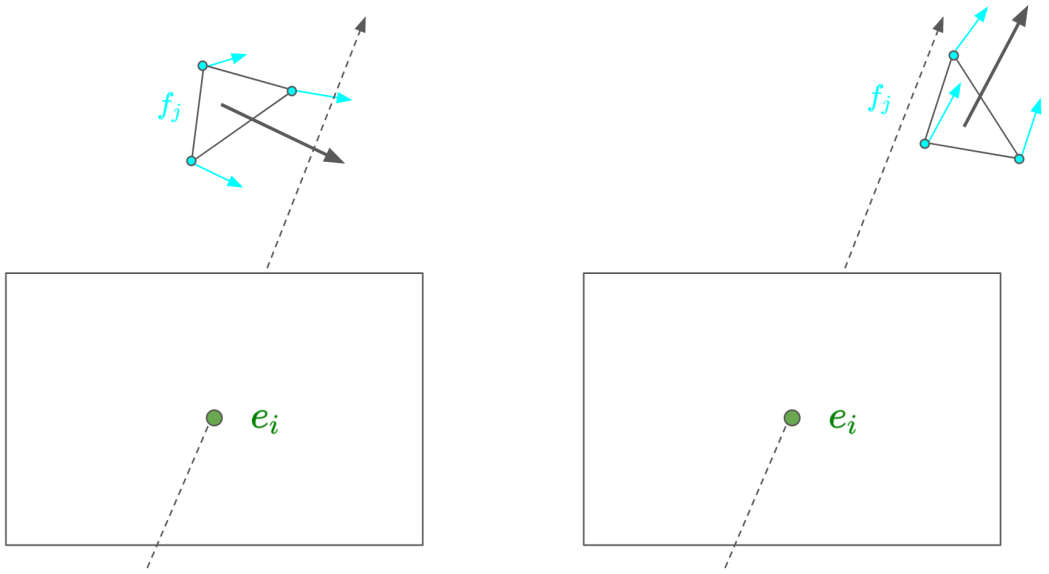
Figure 5.1 intuitively shows that our decision strategy of a mesh face as contour face or texture face for an event can be obtained from the relationship between the unprojection ray through the event and the normal of the mesh face: if they are orthogonal to each other, the mesh face is a contour face w.r.t. the event; if they are parallel to each other, the mesh face is a texture face w.r.t. the event.

Thus, we compute the dot product between the normal of mesh face f_j and the unprojection ray of event e_i, and formulate the contour probability of the mesh face f_j w.r.t. event e_i from the absolute dot product using the exponential function as

$$P_{contour} = \exp(-\frac{|N(f_j) \cdot b(e_i)|}{\gamma}), \tag{5.1}$$

where $N(\cdot)$ denotes the normal of mesh face f_j while $b(\cdot)$ represents the bearing vector of event e_i. γ is the sharpness control parameter.

In our project, we define a hard decision boundary for the contour probability: if the calculated probability is higher than the threshold, we treat event e_i as that it is



(a) Unprojection ray is orthogonal to the normal of mesh face.

(b) Unprojection ray is parallel to the normal of mesh face.

Figure 5.1.: The normal of contour face f_j is orthogonal (a) to the unprojection ray of event e_i, while the normal of texture face f_j is parallel (b) to the unprojection ray of events e_i. Cyan arrows are the normal of vertices.

caused by contour mesh face f_j; Otherwise, it is considered as that it is caused by texture face f_j. According to this, we can split events $\{e_k, e_{k+1}, ..., e_{k+N-1}\}$ into contour events group $\{E_{contour}\}$ and texture events group $\{E_{texture}\}$. Then, we perform the tracking framework using contrast maximization principle (Sec. 3.6) on texture events group $\{E_{texture}\}$, and perform the tracking framework using expectation maximization algorithm (5.3) on contour events group $\{E_{contour}\}$.

We show an intuitive visualization of the distinguished contour mesh faces (Fig. 5.2a) and texture mesh faces (Fig. 5.2b) of a mesh model using proposed contour probability (Eq. 5.1). In this example, we assume the event e_i is at the image center. We calculate the dot product between the bearing vector of event e_i and normals of all mesh faces. Then, we formulate the contour probability from the dot product using Equation 5.1. If the contour probability of a mesh face is larger than the predefined hard boundary, the mesh face is a contour mesh face w.r.t. event e_i.

(a) Contour mesh faces. (b) Texture mesh faces.

Figure 5.2.: Visualized Contour mesh faces and Texture mesh faces.

Figure above shows intuitively that our method can distinguish contour mesh faces and texture mesh faces. Theoretically, an event caused by a contour mesh face is a contour event and an event caused by a texture mesh face is a texture event. We proposed expectation-maximization contour events tracking approach (Sec. 5.3) for contour events and perform contrast-maximizaton-based texture events tracking approach (Sec. 5.4) for texture events.

5.2. General Detail in Tracking Framework

Spatio-temporal windows of events:

we split the stream of asynchronous events into a set of spatio-temporal windows as in figure 5.3 to increase the computational efficiency. We assume that the time difference of the window is so small that all events in the window can be considered as having one mesh pose. Thus, we estimate the mesh pose for an event buffer instead of for a single event, which reduces computational complexity. The k^{th} window is defined as the set of events $W_k = \{e_k, \ldots, e_{k+N-1}\}$, where N is the window size parameter. It is worth mentioning that the number of events is fixed in each window, which means the start time $t_k^f := t_k$ and duration of each window Δt_k^f are controlled by the events. This preserves the data-driven nature of the sensor. Obviously, the selection of window

size N is a trade-off: for the same events stream, a larger N means a lower number of windows and estimation of fewer parameters. A smaller N means more windows, but also less events will be considered as having the same mesh pose. In summary, larger N means lower computation time, smaller N means higher accuracy.

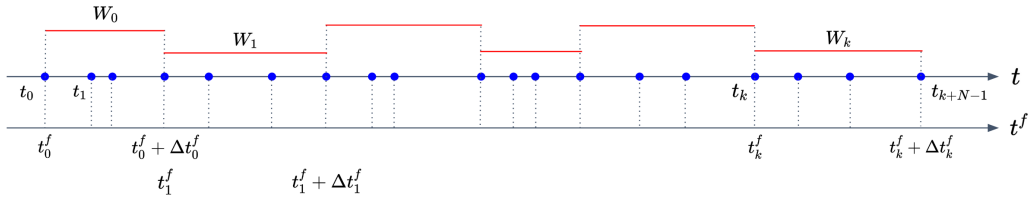


Figure 5.3.: Spatio-temporal windows of events. Events are depicted as blue dots on the timeline. The windows $\{W_k\}$ are marked in red ($N = 4$ here). Note that the temporal size of each window Δt_k^f is automatically adapted to the event rate.

Initialization of desired mesh pose:

For the initial frame, we start from a known mesh pose. Regarding subsequent event frames, we use the estimation result of the last spatio-temporal window θ_{k-1} and the previous mesh pose velocity v_{k-1} to initialize the estimated mesh pose of current spatio-temporal window θ_k. The pose velocity v_{k-1} and the initialization of θ_k can be formulated as

$$v_{k-1} = \frac{\theta_{k-1} - \theta_{k-2}}{\Delta t_{k-1}} \tag{5.2}$$

$$\theta_k = \theta_{k-1} + v_{k-1} \cdot \Delta t_k \tag{5.3}$$

5.3. Expectation Maximization for Contour Tracking

In this section, we propose an event-based non-rigid tracking method using the expectation maximization framework which process contour events. The main idea is to maximize the expectation of the measurement likelihood of the association between contour events and contour mesh faces of objects. Note that event e_i in this part denotes contour event.

We obtain the association relationship between the events and mesh faces using the rasterization of objects. The rasterization process introduced in section 3.2.1 provides

the association between events and mesh faces. However, if an event is outside of the 2D contour of the mesh model, no corresponding mesh face is available. Inspired by soft rasterizer [24], we can assign events to mesh faces in a soft manner, where each face has its probability to cause the event. In other words, an event e_i in the spatio-temporal window W_k (Sec. 5.2) can be caused by any mesh face f_j of the model. Thus, we formulate the measurement likelihood $P(e_i|f_j, \theta_k)$, which means the likelihood that the event e_i is caused by mesh face f_j under the current mesh pose θ_k. We construct a probability map D with the size of $[N, F]$ for all events in the spatio-temporal window W_k, where N is the number of events in the window and F is the number of triangle faces of the mesh model. The computation of the measurement likelihood $P(e_i|f_j, \theta_k)$ can be described in following detailed steps:

1. We unproject from the 2D event location $e_i = (x, y)$ to compute the 3D bearing vector $\mathbf{b_i} = (m_x, m_y, m_z)$ using the pinhole camera model:

$$\begin{pmatrix} m_x \\ m_y \\ m_z \end{pmatrix} = \begin{pmatrix} \frac{x-c_x}{f_x} \\ \frac{y-c_y}{f_y} \\ \left(1 + m_x^2 + m_y^2\right)^{-\frac{1}{2}} \end{pmatrix}, \tag{5.4}$$

   where (f_x, f_y) is the focal length and (c_x, c_y) is the image principle point. After multiplying two arbitrary depths value with bearing vector b_i, we obtain two 3D points, P_A and P_B, lying on the unprojection ray through the event e_i.

2. We calculate the lateral distance from the unprojection ray to each individual mesh face. Intuitively, the closer the mesh face to the unprojection ray, the higher the probability that the mesh face causes the corresponding event. We have the following method to calculate the lateral distance:

   - Line-Edge-distance: minimal distance between the unprojection ray and edges of the mesh face. An event can be caused by an arbitrary point on the mesh face. Therefore, we introduce the method (Fig. 5.4) which calculates the distance between the unprojection ray and the closest edge of the mesh face. Here, we use the formula in section 3.5.2: the line of unprojection ray is defined by points p_A and p_B, while three edges of the mesh face are line segments defined by vertices $(\overline{x}_{1j}, \overline{y}_{1j}, \overline{z}_{1j})^T$, $(\overline{x}_{2j}, \overline{y}_{2j}, \overline{z}_{2j})^T$, and $(\overline{x}_{3j}, \overline{y}_{3j}, \overline{z}_{3j})^T$ (in cyan in Fig. 5.4). Afterwards, the minimal distance of three edges is used as the lateral distance.

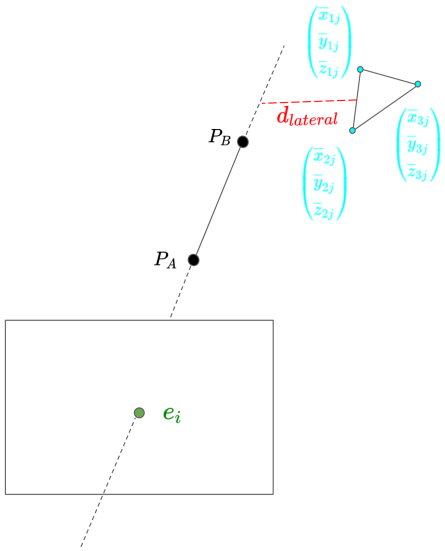


Figure 5.4.: Line-Edge lateral distance between the unprojection ray of event e_i and mesh face f_j.

The lateral probability from the distance between the event unprojection ray and the edge has two cases: If the unprojection ray doesn't goes through the mesh face, the smaller the lateral distance, the higher the association probability; If the unprojection ray go through the mesh face, the larger the lateral distance, the further the unprojection ray from the boundary of the mesh face, and therefore the higher the association probability. Inspired by SoftRasterizer [24], we use a sigmoid function to model the lateral probability from the line-edge-distance:

$$P_{lateral} \propto sigmoid(\delta_j^i \cdot \frac{f(d_{lateral}(i,j))}{\alpha}), \tag{5.5}$$

where the α is the sharpness control parameter. $f(\cdot)$ is the Charbonnier robust kernel function [3], which is more robust compared to L2-norm. δ_j^i is a sign indicator that $\delta_j^i = +1$ if the unprojection ray of event e_i intersects with mesh face f_j and $\delta_j^i = -1$ if they don't intersect. Here, we use the formula introduced in section 3.5.3 to determine whether the unprojection ray defined by points P_A and P_B intersects with a 3D triangle face spanned by $(\bar{x}_{1j}, \bar{y}_{1j}, \bar{z}_{1j})^T$, $(\bar{x}_{2j}, \bar{y}_{2j}, \bar{z}_{2j})^T$, and $(\bar{x}_{3j}, \bar{y}_{3j}, \bar{z}_{3j})^T$.

3. In addition to the lateral distance, we also have to consider the longitudinal distance from the mesh face to the image plane. Normally, the unprojection ray

will go through two mesh faces on the front and the back side of the model. Taking the longitudinal distance into consideration helps us to avoid assigning the event to the mesh face, which has a small lateral distance but is far away from the image plane. We also have the following method to calculate the longitudinal distance:

- Length along the unprojection ray: We consider the projected mesh face distance on the event projection ray as the longitudinal distance. We construct a right triangle connecting the image center, the mean face location, and the closest point to the mean face location on the unprojection ray. We use mean vertices location $(\overline{x}_j, \overline{y}_j, \overline{z}_j)^T$ (in cyan in Fig 5.4) to indicate the location of face f_j. P_A and P_B are two arbitrary 3D points on the unprojection ray, and they determine a 3D line along the unprojection ray through the event e_i. Here, we use the formula introduced in section 3.5.1 to calculate the distance $d_{e_i}^{f_j}$ between the unprojection ray of event e_i and the mean location of the mesh face f_j. According to the Pythagorean theorem [16], the distance from the image center to the closest point on the unprojection ray can be formulated as

$$d_{longitudinal} = \left((\overline{x}_j, \overline{y}_j, \overline{z}_j) \begin{pmatrix} \overline{x}_j \\ \overline{y}_j \\ \overline{z}_j \end{pmatrix} - d_{e_i}^{f_j} * d_{e_i}^{f_j} \right)^{\frac{1}{2}}. \tag{5.6}$$

  In the data association step, if several mesh faces have the similar lateral distance w.r.t. an event unprojection ray, the mesh face with smaller longitudinal distance should have higher likelihood to be associated with the event. Thus, we model this effect by the longitudinal probability formulated as

$$P_{longitudinal} \propto e^{(-\frac{d_{longitudinal}(i,j)}{\beta})}, \tag{5.7}$$

  where β is the sharpness control parameter.

4. The contour probability $P_{contour}$ is formulated using the dot product between the unprojection ray and mesh face normals as explained in Equation 5.1.

5. Given the individual probability term $P_{lateral}$, $P_{longitudinal}$, $P_{contour}$, we can formulate the data measurement likelihood. It is worth mentioning that the data likelihood term should be formalized differently for the expectation step and in the maximization step. In the expectation step, we consider longitudinal term $P_{longitudinal}$ because the closer mesh face has higher probability to cause the event. Besides, we consider lateral term $P_{lateral}$ and contour term $P_{contour}$ as well. It means the

mesh face which is closer and more orthogonal w.r.t. the event unprojection ray have higher probability to be associated with the event. Thus, the data likelihood for the expectation step is formulated as

$$P\left(e_i|f_j,\theta_k\right) \propto P_{lateral} \cdot P_{longitudinal} \cdot P_{contour}. \tag{5.8}$$

However, the longitudinal term $P_{longitudinal}$ should be dropped in the maximization step, since the object pose should not be optimized to have smaller longitudinal distance. The lateral term $P_{lateral}$ and the contour term $P_{contour}$ formulates the data likelihood for the maximization step:

$$P\left(e_i|f_j,\theta_k\right) \propto P_{lateral} \cdot P_{longitudinal}. \tag{5.9}$$

Now we have association data likelihood $P(e_i|f_j,\theta_k)$ between events and all mesh faces under current mesh pose. Now, the hard association correspondence using rasterization can be replaced by soft association process. Then, we propose an expectation maximization framework, which uses the association between events and mesh faces as the hidden variable.

The background of the expectation maximization algorithm is explained in section 3.7. As stated in equation 3.20, the analytical formulation of likelihood $P(\mathbf{X}|\boldsymbol{\theta})$ is difficult since there is no observable relation between measurement $\mathbf{X}$ and model $\boldsymbol{\theta}$ available. Therefore, people introduce hidden variable $\mathbf{Z}$ to rewrite the original likelihood into following marginal likelihood:

$$P(\mathbf{X} \mid \boldsymbol{\theta}) = \sum_{\mathbf{Z}} P(\mathbf{X}, \mathbf{Z} \mid \boldsymbol{\theta}). \tag{5.10}$$

We use the associations as the hidden variables, which is a novelty in our tracking framework. In our project, measurements $\mathbf{X}$ are all events $\{e_k, ..., e_{k+N-1}\}$ in a spatio-temporal window W_k. The desired model $\boldsymbol{\theta}$ means the mesh model defined by the pose parameter θ_k. The posterior distribution of the hidden variable $P(\mathbf{Z}|\mathbf{X},\boldsymbol{\theta})$ in our project is the posterior distribution of the associations given events and mesh pose.

The original expectation of the complete-data log likelihood is stated in equation 3.21. We replace the hidden variable $\mathbf{Z}$ and measurement $\mathbf{X}$ with variable a and e, respectively. a indicates the probability that mesh faces associates to event e. Then, we get the expectation of the logarithmic likelihood (LL) in our tracking framework using expectation maximization algorithm:

$$E(LL(f(\theta_i) \mid e, a)) = \sum_{a_j \in faces} P\left(a_j \mid e, f(\theta^{old})\right) \cdot \ln P\left(e, a_j \mid f(\theta)\right), \tag{5.11}$$

where $f(\cdot)$ indicates a pipeline like MANO which takes pose parameters and outputs a mesh model. Obviously, the hidden variable a_j is marginalized with $\sum_{a_j \in f}$. It can be explained that for a single event e, summing up the association probability of all mesh faces is equivalent to the marginalization of the association variable. $P(a_j|e, f(\theta^{old}))$ is the hidden state distribution. It is worth mentioning that in the expectation step, we always use the fixed previous model parameter θ^{old} to compute the hidden state distribution. In other words, we estimate how the hidden variables distribute according to our previous knowledge of the model θ. $\ln P(e, a_j|f(\theta_i))$ is the logarithmic likelihood term. We compute the logarithmic likelihood based on all states of the hidden variable, and calculate the expectation of the log-likelihood according to the individual states distribution of the hidden variable.

The hidden variable distribution $P(a_j|e, f(\theta))$ can be calculated from measurement likelihood $P(e|a_j, f(\theta))$ based on Bayes' theorem [10]:

$$\begin{aligned} P\left(a_j \mid e, f(\theta)\right) &= \frac{P\left(a_j, e \mid f(\theta)\right)}{P(e \mid f(\theta))} \\ &= \frac{P\left(e \mid a_j, f(\theta)\right) \cdot P\left(a_j \mid f(\theta)\right)}{P(e \mid f(\theta))} \\ &= \frac{P\left(e \mid a_j, f(\theta)\right) \cdot P\left(a_j \mid f(\theta)\right)}{\sum_{a_j} P\left(a_j, e \mid f(\theta)\right)} \\ &= \frac{P\left(e \mid a_j, f(\theta)\right) \cdot P\left(a_j \mid f(\theta)\right)}{\sum_{a_j} P\left(e \mid a_j, f(\theta)\right) \cdot P\left(a_j \mid f(\theta)\right)}, \end{aligned} \tag{5.12}$$

we assume that the association variable a_j is unconditioned on mesh $f(\theta)$ without the measurement variable e. Therefore, we assume that probability $P(a_j|f(\theta))$ has a uniform distribution. Thus, equation 5.12 turns into

$$P\left(a_j \mid e, f(\theta)\right) = \frac{P\left(e_i \mid a_j, f(\theta)\right)}{\sum_{a_j} P\left(e \mid a_j, f(\theta)\right)}, \tag{5.13}$$

where we can represent hidden variable distribution $P\left(a_j|e, f(\theta)\right)$ purely with the available measurement likelihood $P(e|a_j, f(\theta))$.

Similarly, logarithmic likelihood term $P(e, a_j|f(\theta))$ can also be calculated from measurement likelihood term $P(e|a_j, f(\theta))$ as

$$\begin{aligned} \ln P\left(e, a_j \mid f(\theta)\right) &= \ln\left(P\left(e \mid a_j, f(\theta)\right) \cdot P\left(a_j \mid f(\theta)\right)\right) \\ &= \ln\left(P\left(e \mid a_j, f(\theta)\right)\right), \end{aligned} \tag{5.14}$$

where $P(a_j|f(\theta))$ is assumed to be a uniform distribution and neglected.

Now, we can represent the expectation of the logarithmic likelihood in equation 5.11 purely with the available measurement likelihood $P(e|a_j, f(\theta))$:

$$E(LL(f(\theta_i) \mid e, a)) = \sum_{a_j \in faces} \frac{P\left(e \mid a_j, f(\theta_{i-1})\right)}{\sum_{a_j} P\left(e \mid a_j, f(\theta_{i-1})\right)} \cdot \ln\left(P\left(e \mid a_j, f(\theta_i)\right)\right), \tag{5.15}$$

where θ^{i-1} is the estimation of mesh pose in the last iteration, which is fixed and will not be optimized in current iteration. Equation 5.15 is the objective function in our tracking framework using expectation maximization algorithm. In summary, we maximize the likelihood that an event e is caused by an arbitrary mesh face to optimize over current mesh pose θ_i.

We now have the objective function for an event e_i. Our EM-based tracking framework allows non-rigid tracking using single event input. As introduced in section 2.1, events are usually timestamped with microsecond resolution. In principle, our EM-based tracking framework has the potential to estimate the mesh pose with microsecond temporal resolution as well. It fully benefits from the advantage in high temporal resolution of event cameras which is not achieved by other event-based non-rigid tracking frameworks in [35], in [26], and in section 5.4.

To increase the computational efficiency, we assume that N events $\{e_k, ..., e_k + N - 1\}$ in spatio-temporal window W_k are caused by the same mesh with pose θ_k. A visualization of events in spatio-temporal window can be found in figure 5.3. The expectation of logarithmic likelihood for multiple events can be formulated as

$$\begin{aligned} \sum_{a_k} \sum_{a_{k+1}} \cdots \sum_{a_{k+N-1}} (&P\left(a_k, a_{k+1}, \cdots, a_{k+N-1} \mid e_k, z_{k+1}, \cdots, z_{k+N-1}, f(\theta_k)\right) \cdot \\ &\ln P\left(z_k, z_{k+1}, \cdots, z_{k+N-1}, a_k, a_{k+1}, \cdots, a_{k+N-1} \mid f(\theta_k)\right)), \end{aligned} \tag{5.16}$$

where a_i is the association situation of event e_i. To be more intuitive on the dependency between variables, we generate a Bayesian network to interpret the relationship between events $\{e_k, ..., e_{k+N-1}\}$, association situation for each event $\{a_k, ..., a_{k+N-1}\}$, and desired mesh pose θ_k:

We assume that events are independent from each other. We can apply the chain rule [36] on equation 5.16 and rewrite it into

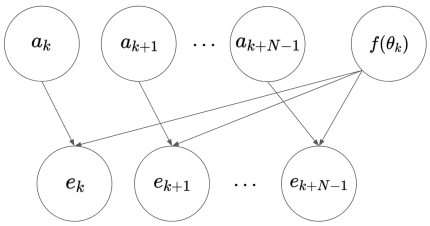


Figure 5.5.: Bayesian network with variables $\{e_k, ..., e_{k+N-1}\}$, $\{a_k, ..., a_{k+N-1}\}$, and θ_k. It is intuitively shown that each event e_i is only dependent on it parents: the association situation a_i and the mesh pose parameter θ_k.

$$
\begin{aligned}
&\sum_{a_k} \cdots \sum_{a_{k+N-1}} (P\left(a_k, a_{k+1}, \cdots, a_{k+N-1} \mid e_k, z_{k+1}, \cdots, z_{k+N-1}, f(\theta_k)\right) \cdot \\
&\qquad \ln P\left(z_k, \cdots, z_{k+N-1}, a_k, \cdots, a_{k+N-1} \mid f(\theta_k)\right)) \\
=&\sum_{a_k} \cdots \sum_{a_{k+N-1}} (P\left(a_k \mid e_k, \cdots e_{k+N-1}, f(\theta_k)\right) \cdots P\left(a_{k+N-1} \mid e_k, \cdots e_{k+N-1} \cdot f(\theta_k)\right) \cdot \\
&\qquad \ln \left((P(e_k|a_k, f(\theta_k)) \cdots P\left(e_{k+N-1} \mid a_{k+N-1}, f(\theta_k)\right)\right) \\
=&\sum_{a_k} \cdots \sum_{a_{k+N-1}} (P\left(a_k \mid e_k, f(\theta_k)\right) \cdots P\left(a_{k+N-1} \mid e_{k+N-1}, f(\theta_k)\right) \cdot \\
&\qquad \ln \left(P\left(e_k \mid a_k, f(\theta_k)\right) \cdots P\left(e_{k+N-1} \mid a_{k+N-1}, f(\theta_k)\right)\right)) \\
&\qquad \left(\ln \left(P\left(e_k \mid a_k, f(\theta_k)\right)\right) + \cdots + \ln \left(P\left(e_{k+N-1} \mid a_{k+N-1}, f(\theta_k)\right)\right)\right)) \\
=&\sum_{a_k} \cdots \sum_{a_{k+N-1}} (P\left(a_k \mid e_k, f(\theta_k)\right) \underbrace{\cdots P\left(a_{k+N-1} \mid e_{k+N-1}, f(\theta_k)\right)}_{\text{sums up to 1}} \cdot \ln \left(P\left(e_k \mid a_k, f(\theta_k)\right)\right)) \\
+&\sum_{a_k} \cdots \sum_{a_{k+N-1}} \cdots \\
+&\sum_{a_k} \cdots \sum_{a_{k+N-1}} (\underbrace{P\left(a_k \mid e_k, f(\theta_k)\right) \cdots}_{\text{sums up to 1}} P\left(a_{k+N-1} \mid e_{k+N-1}, f(\theta_k)\right) \cdot \ln \left(P\left(e_{k+N-1} \mid a_{k+N-1}, f(\theta_k)\right)\right)) \\
=&\underbrace{\sum_{a_k} (P\left(a_k \mid e_k, f(\theta_k)\right) \cdot \ln \left(P\left(e_k \mid a_k, f(\theta_k)\right)\right))}_{\text{expectation of log-likelihood for } e_k} \\
+&\cdots \\
+&\underbrace{\sum_{a_{k+N-1}} (P\left(a_{k+N-1} \mid e_{k+N-1}, f(\theta_k)\right) \cdot \ln \left(P\left(e_{k+N-1} \mid a_{k+N-1}, f(\theta_k)\right)\right))}_{\text{expectation of log-likelihood for } e_{k+N-1}} \\
=&E(LL(f(\theta_k) \mid e_k, a_k)) + \cdots + E(LL(f(\theta_k) \mid e_{k+N-1}, a_{k+N-1})).
\end{aligned}
\tag{5.17}
$$

It proves that maximizing the expectation of logarithmic association likelihood of multiple events caused by same pose θ_k is equivalent to maximizing the sum of the expectation of all events in the spatio-temporal window W_k.

Finally, given the events input e_i, we maximize the above stated objective function to optimize for pose parameter θ_k:

$$\underset{\theta_k}{\operatorname{argmax}}(E(LL(f(\theta_k) \mid e_k, a_k)) + \cdots + E(LL(f(\theta_k) \mid e_{k+N-1}, a_{k+N-1}))) \tag{5.18}$$

We provide a brief summary of the EM-based non-rigid tracking framework in algorithm 1. After the initialization process of θ_k, the tracking framework can be divided into two steps:

1. E-step: For each event in spatio-temporal window W_k, we calculate the lateral distance, the longitudinal distance, and the dot product from the event to each face of the mesh model. Then, we compute the measurement likelihood that the event is caused by the individual mesh face based on the distance. According to equation 5.15, we can calculate the hidden variable distribution and the logarithmic likelihood from the measurement likelihood. Finally, we compute the expectation of the log-likelihood. As proven in equation 5.17, we sum the ELL for all events up. It is the value of the Q-function as stated in equation 3.21.

2. M-step: we maximize the value of Q-function to solve for the desired mesh pose parameter θ_k.

The expectation maximization algorithm is an iterative algorithm. It iterates over E-step and M-step until Q-function converges. We show the algorithm of the expectation maximization contour tracking approach in Alg. 1.

As indicated in equation 5.11, hidden variable distribution $P(a|e_i, \theta^{old})$ uses the previous estimation on mesh pose parameter θ^{old}. In other words, this term doesn't propagate the gradient from the pose parameter θ, which needs to be solved. We visualize the gradient map in our EM-based non-rigid tracking framework in figure 5.6. It shows that the gradient flow is not propagated through hidden variable distribution $P(a|e_i, \theta_k)$.

Theoretically, our proposed EM-contour-tracker is limited to process contour events. A texture event already has an intersection with a mesh face, which means it already has a high expectation of the data likelihood given the association. Thus, our EM-contour-tracker cannot perform tracking based on texture events.

Algorithm 1 EM-contour-tracking using event-based cameras

Input: events $\{e_k, \cdots, e_{k+N-1}\}$ in spatio-temporal window W_k
Output: optimized mesh pose parameter θ_k

procedure ExpectationMaximization
  $\theta_k \leftarrow$ *initialization of mesh pose*, (Eq.5.2, 5.3)
  E-step:
  $f(\theta_k) \leftarrow$ *generate mesh model given pose parameter* θ_k, (Sec.3.1)
  $obj\_func \leftarrow 0$, *initialization of objective function*
  for e_i in $\{e_k, \cdots, e_{k+N-1}\}$ **do**
    $d^i_{normal} \in [F] \leftarrow$ *dot product between event* e_i *to F faces*, (Eq.5.1)
    $d^i_{lateral} \in [F] \leftarrow$ *lateral distance from event* e_i *to F faces*, (Fig.5.4)
    $d^i_{longitudinal} \in [F] \leftarrow$ *longitudinal distance from event* e_i *to F faces*, (Eq.5.6)
    $P(e_i|a, f(\theta)) \in [F] \leftarrow$ *Likelihood of event* e_i *caused by F faces*, (Eq.5.8, 5.9)
    $E(LL(f(\theta_k|e_i, a))) \leftarrow$ *expectaion of log-likelihood of event* e_i, (Eq.5.15)
    $obj\_func \leftarrow obj\_func + E(LL(f(\theta_k|e_i, a)))$, (Eq.5.17)
  M-step:
  $\theta_k \leftarrow \underset{\theta_k}{\text{argmax}}\ obj\_func$
  if *Optimization doesn't converge* **then**
    goto *E-step*.

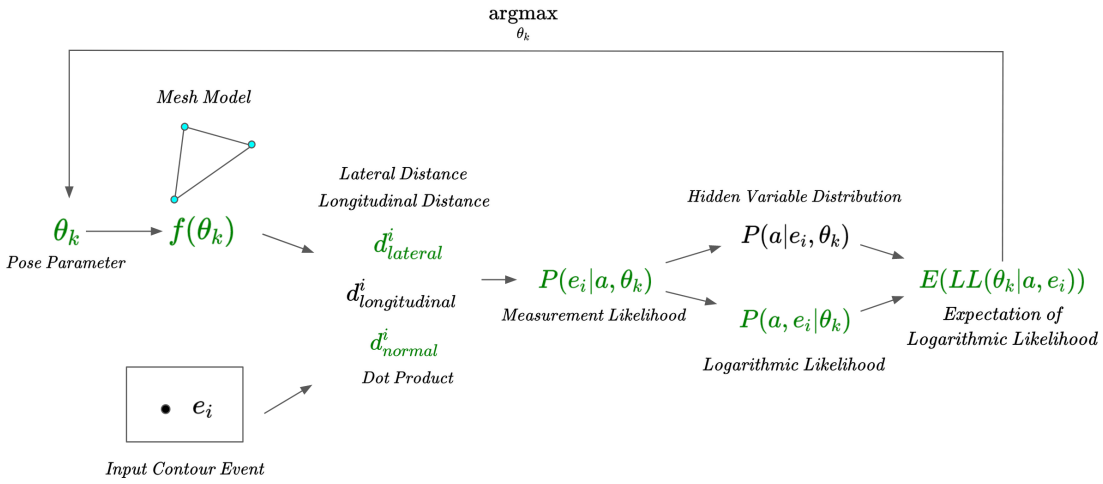


Figure 5.6.: Gradient flow map in EM-based tracking framework. Variables in green means they contain gradient flow. Variables in black means they don't contain gradient flow. In other words, gradient is not propagated through those variables.

5.4. Contrast Maximization for Texture Events

For texture events, we propose CM-texture-tracker which uses the contrast maximization principle introduced in section 3.6. As stated in [14], events triggered by the same moving edge lie on a point trajectory, which is defined by estimated motion parameters. Events on a point trajectory are warped back to the same location on a reference frame, when motion parameters define the point trajectory accurately. Thus, we can maximize the sharpness of the **IWE** (**I**mage of **W**arped **E**vents) to optimize over desired motion parameters.

The point trajectory in our non-rigid tracking task can be understood as the trajectory of the mesh face which causes those events. A point trajectory for a set of corresponding events is visualized intuitively in figure 5.7. At time t, $t+1$, and $t+2$, three events (indicated with green dots) are caused by 3D points on the same mesh face i, respectively. The positions of 3D points are determined by barycentric coordinates and the location of three corresponding vertices. Barycentric coordinates are obtained in the rasterization process. A line connecting these 3D points is the point trajectory for the set of events in figure 5.7.

Estimated motion parameters in our project are the non-rigid mesh deformation. The mesh deformation contains the spatial information of vertices for all mesh faces. Assuming the mesh deformation is correctly estimated, each event can find the correct

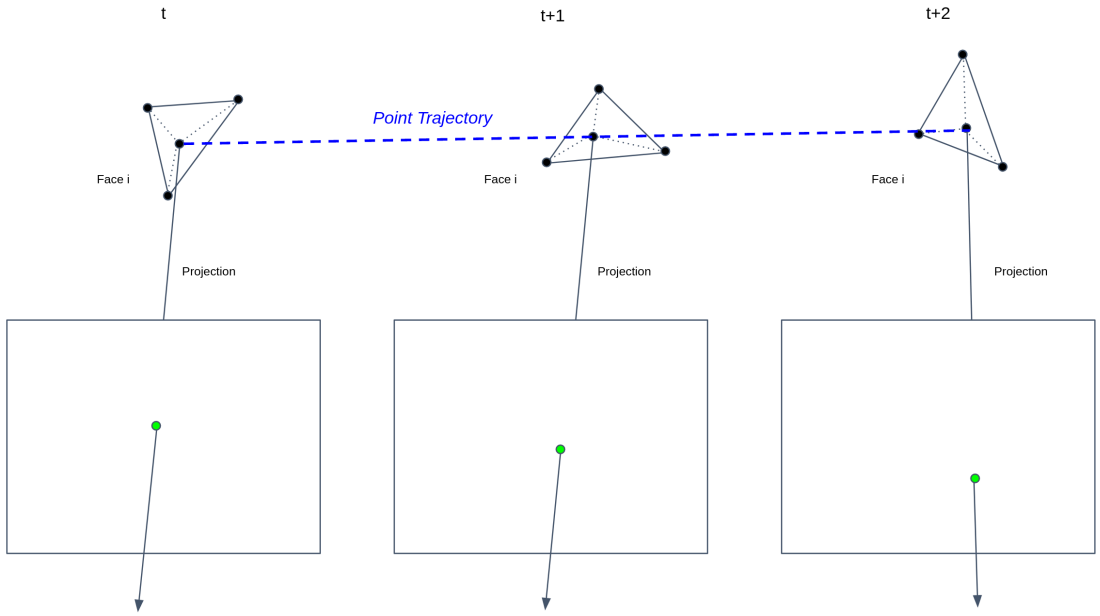


Figure 5.7.: Visualization of point trajectory. Green dots at frame t, $t+1$, and $t+2$ are corresponding events which caused by same mesh face i. Trajectory of 3D points which cause a set of corresponding events is the point trajectory of the events.

corresponding mesh face. After knowing the index of the corresponding mesh face, we can project the mesh face using mesh information in the reference frame. Thus, we warp the original event back to the reference frame. In principle, the corresponding events caused by the same mesh face will be warped back to the same position. After warping all events in a tiny spatio-temporal window, we obtain an IWE. It should have the highest sharpness if mesh information is correctly estimated and all events find their corresponding mesh faces.

The tracking method using the contrast maximization algorithm can be described in the following detailed steps:

1. **Find corresponding mesh faces for all events:** for all events in the spatio-temporal window (Sec. 5.2), we have to find the corresponding mesh face. As introduced in section 3.2.1, the rasterization process provides the pixel-to-face correspondence and the corresponding barycentric coordinate. In this step, we rasterize the mesh which is generated using the desired mesh pose θ_k. Thus, we can get the corresponding mesh face index and the barycentric coordinate for each event

in the spatio-temporal window. After doing the barycentric interpolation, we can find the corresponding 3D points of each event. These 3D points can be considered as the 3D events. In other words, we lift the 2D events to 3D events in this step.

2. **Find corresponding mesh faces at reference frame:** the core challenge in the contrast maximization principle is to warp all the events into the reference frame. In our project, we have to find the warped event position at reference frame. We choose the previous spatio-temporal window as the reference frame, which has the mesh pose θ_{k-1}. Then, we use the mesh vertex locations for the mesh face at the reference frame to do the barycentric interpolation. Thus, we find the corresponding 3D point at the reference frame for the event. It could also be described as the 3D event at the reference frame.

3. **Project mesh faces onto image plane:** after knowing the location of the corresponding mesh faces or 3D events at the reference frame, we can project them back to the 2D image plane to obtain the 2D warped events position. The warping process of 2D events is also called *reprojection*. A visualization of the reprojection process can be found in figure 4.5. Essentially, events in S spatio-temporal windows are reprojected to the reference frame to generate the IWE. In our project, we use the method described in the first step to initialize the mesh pose $\{\theta_k, ..., \theta_{k+S-1}\}$. Then, we repeat the reprojection process with individual mesh poses. The final IWE contains the warped events in spatio-temporal window W_k to W_{k+S-1}.

4. **Perform contrast maximization on IWE:** the corresponding events caused by the same mesh face should be warped back to the same position. We maximize the sharpness of the generated IWE to optimize over mesh poses $\{\theta_k, ..., \theta_{k+S-1}\}$. As introduced in the last step, we use individual mesh poses to warp events in different spatio-temporal window to the reference frame. Thus, the IWE contains the gradient w.r.t. all mesh poses. We can perform gradient-based optimization to solve for mesh poses. There are several reward functions mentioned in section 2.2.1 to measure the sharpness of IWE. In our project, we use the variance

$$r_{\sigma^2}(H) = \frac{1}{N_p} \sum_{i,j} \left(h_{i,j} - \mu_H\right)^2 \tag{5.19}$$

and **SoSA** (**S**um **o**f **S**uppressed **A**ccumulations) [38]

$$r_{\text{SoSA}}(H) = \sum_{i,j} e^{(-h_{i,j} * p)} \tag{5.20}$$

as the reward function, where $h_{i,j}$ is the number of reprojected events on pixel at *i-th* row and *j-th* column on the IWE.

An intuitive visualization of all steps is available in figure 5.10. The reprojection process (step 1 to 3) is similar to the motion field generation in our simulator, which is visualized in figure 4.5.

Similar to our EM-contour-tracker, the proposed CM-texture-tracker also has limitations:

1. Event correspondences can be wrongly processed in the contrast maximization because of self-occlusion and non-rigid deformation. The contrast maximization framework framework implicitly handles data association between events, which is a central problem in event-based vision [14]. However, the contrast maximization framework doesn't work when the event correspondences are wrongly determined. Because of the self-occlusion and non-rigid deformation, events generated close to each other can be caused by different mesh faces. However, those events will be considered as corresponding events and be tried to warp back to the same position in the contrast maximization framework.

   An example of wrong event associations because of the self-occlusion is shown in figure 5.8. The bottom subfigures show the hand before and after the deformation, respectively. The upper subfigure shows the generated events during the deformation. It is clear that the events caused by the motion of the small finger tip are caused by different mesh faces. Our experiments proves that contrast maximization method doesn't perform well for texture-less objects tracking.

   In our observation, this effect happens more frequently when objects have less texture. Our conclusion is that the events for texture-rich objects are mostly caused by texture, while most events for texture-less objects are caused by the boundary. Obviously, the boundary of objects is highly likely to suffer from self-occlusion in the non-rigid deformation. Thus, our CM-texture-tracker cannot perform tracking based on contour events.

2. During our experiments we found that the CM-texture-tracker is biased towards some specific motion. As introduced in this section, we use sharpness measurement of the IWE as the reward function. Figure 5.9 shows that the valid size of IWE has strong impact on the final reward function: we reproject events in a spatio-temporal window on to a further image plane (in 5.9a) and a closer image plane (in 5.9b). Intuitively, the further image plane has a larger projected IWE and the closer image plane has a smaller projected IWE. The larger IWE has a smaller variance of 309.3432, while the smaller IWE has a larger variance of 446.1528.

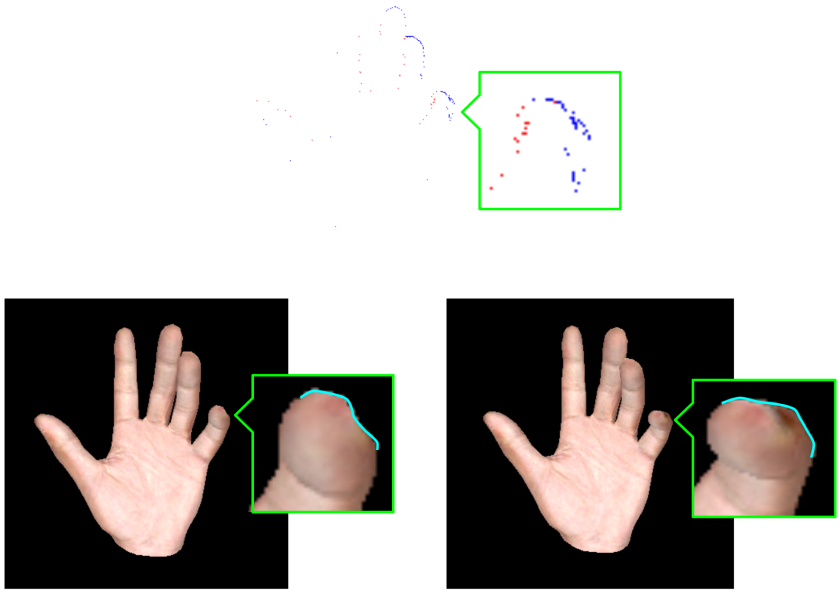
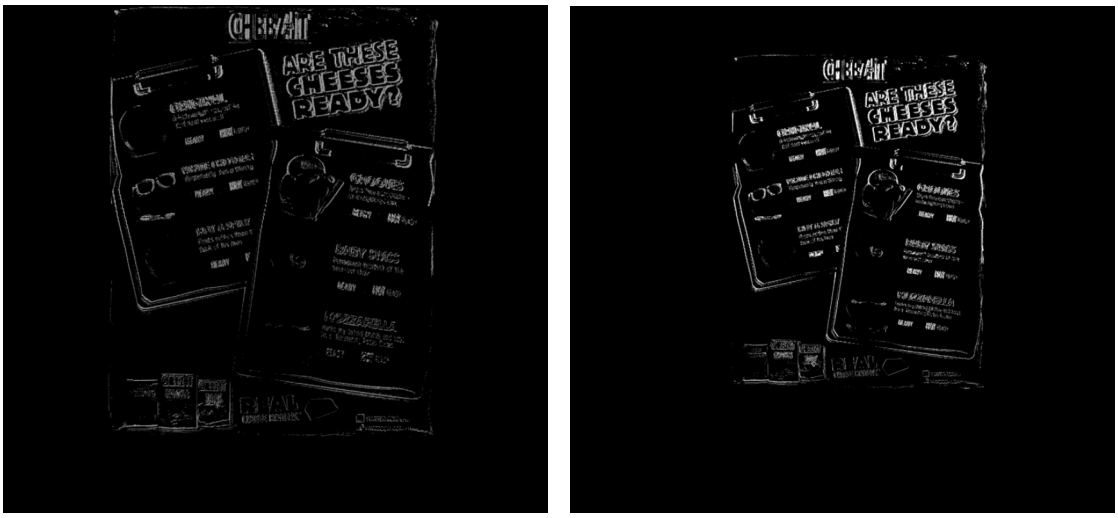


Figure 5.8.: Wrong events association because of the self-occlusion. Curves in cyan indicate the mesh faces causing events around the small finger tip. Those events caused by different faces will be considered as associated events in contrast maximization framework, which is obviously wrong.

Instead of having a closer image plane, moving object far away from image plane can also cause a smaller projection. We also show a landscape heatmap of reward function value in section 6.2.2 to prove that contrast maximization framework has bias in specific motion direction in our event-based objects tracking task.



(a) Larger IWE, variance = 309.3432. (b) smaller IWE, variance = 446.1528.

Figure 5.9.: Reprojection of same events onto a further (a) and a closer (b) image plane. The valid size of the IWE has strong impact on the reward function.

Given above mentioned limitations, our CM-texture-tracker can only process texture events. For event streams containing both contour events and texture events, we combine our EM-contour-tracker and CM-texture-tracker to deal with different type of events. The full tracking framework is introduced in section 5.5.

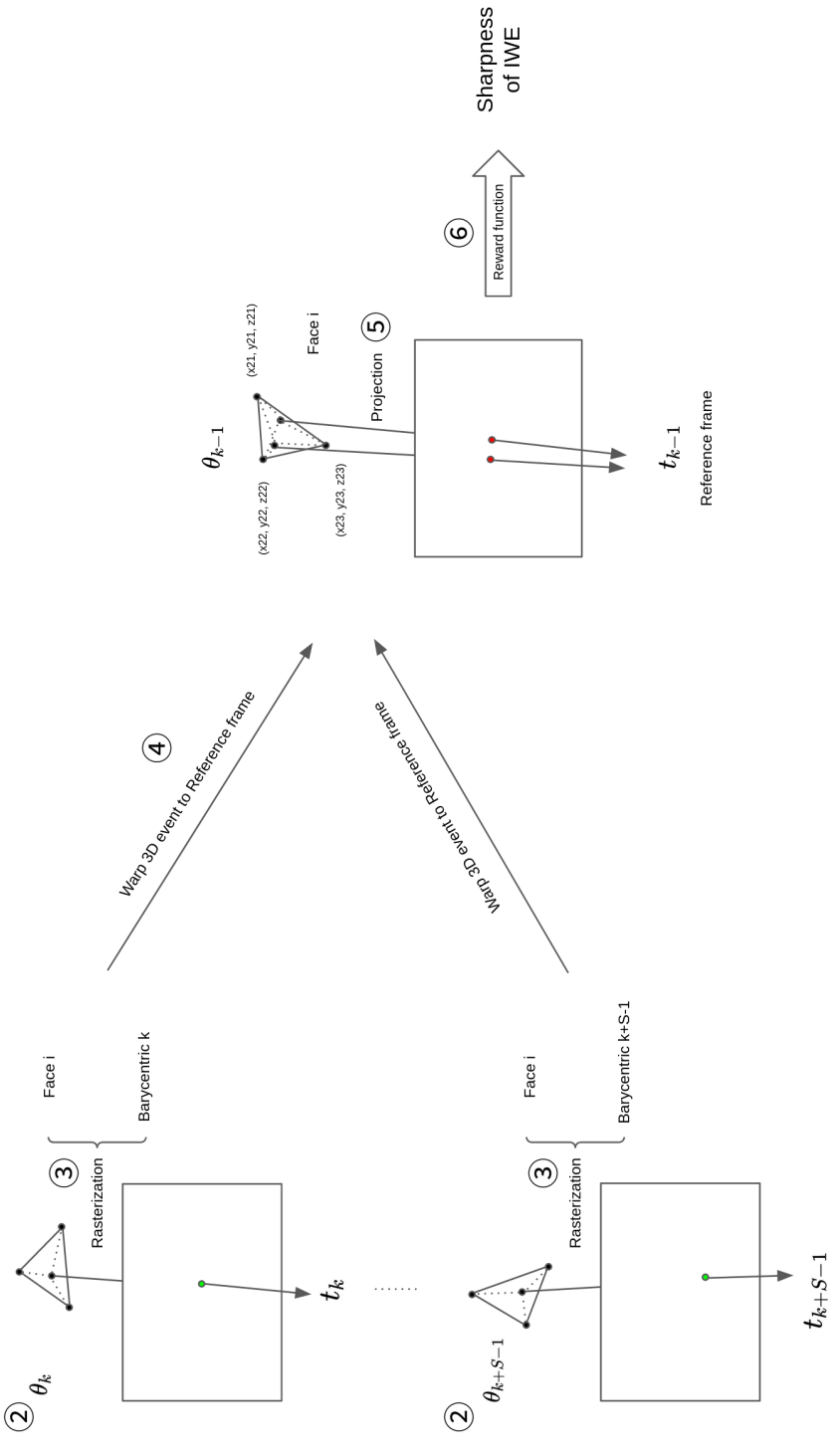


Figure 5.10.: Visualization of detailed steps in contrast maximization. Each step is indexed with the corresponding number. Green dots at frame from t_k to t_{k+S-1} are corresponding events caused by the same mesh face. Red dots at the reference frame t_{k-1} are warped events through reprojection process.

5.5. Full Tracking Framework

As introduced in section 5.3 and section 5.4, the CM-texture-tracker is appropriate to deal with events that are caused by texture mesh faces, and the EM-contour-tracker is suitable on events generated by contour mesh faces, which cannot find corresponding mesh faces via the rasterization process. We distinguish contour events and texture events as introduced in section 5.4. Afterward, we treat these two types of events with EM-contour-tracker and CM-texture-tracker, respectively.

As introduced in section 5.4 and 5.3, the tracking framework using texture-CM and contour-EM principle has their individual reward function. We use weights k_1 and k_2 to compensate for the scale between two reward functions. The final reward function of our tracking framework is

$$obj\_func = k_1 \cdot E\left(LL\left(\theta_k \mid a, e_i\right)\right) + k_2 \cdot \sum_{i,j} e^{(-IWE(i,j)*p)} \tag{5.21}$$

The process of our final tracking framework is visualized in figure 5.11. The tracking framework can be explained as a two-level expectation maximization framework. The first level is to obtain the distribution of contour events among all events in the spatio temporal window. We compute the probability of the face that has the highest likelihood for each event being a contour face, and compare it with a threshold. If the probability is above the threshold, we treat this event as a contour event and perform the second level EM for association likelihood, which is the same framework as introduced in figure 5.6. If the probability is below the threshold, we consider this event as a texture event. We perform contrast maximization in section 5.4 on several texture events, which internally maximize the likelihood that those texture events caused by the same texture mesh face are warped back to the same position on the reference frame.

To increase the smoothness of the tracking process, we also have a constant velocity regularizer term in our reward function. It is weighted by k_3 and turns the final reward function into

$$\begin{aligned} obj\_func =& k_1 \cdot E\left(LL\left(\theta_k \mid a, e_i\right)\right) + k_2 \cdot r_{\mathrm{SoSA}}(IWE(\theta_k)) \\ & - k_3 \cdot \left|\left|\left(\frac{\theta_k - \theta_{k-1}}{\Delta t} - (v_{k-1})\right)\right|\right|^2, \end{aligned} \tag{5.22}$$

where θ_k is the optimized mesh pose of current spatio-temporal window, W_k, θ_{k-1} is the mesh pose of the last window, Δt is the time difference from the last window to the current window. k_1, k_2, k_3 are hyperparameters and are tuned as in 3.3.

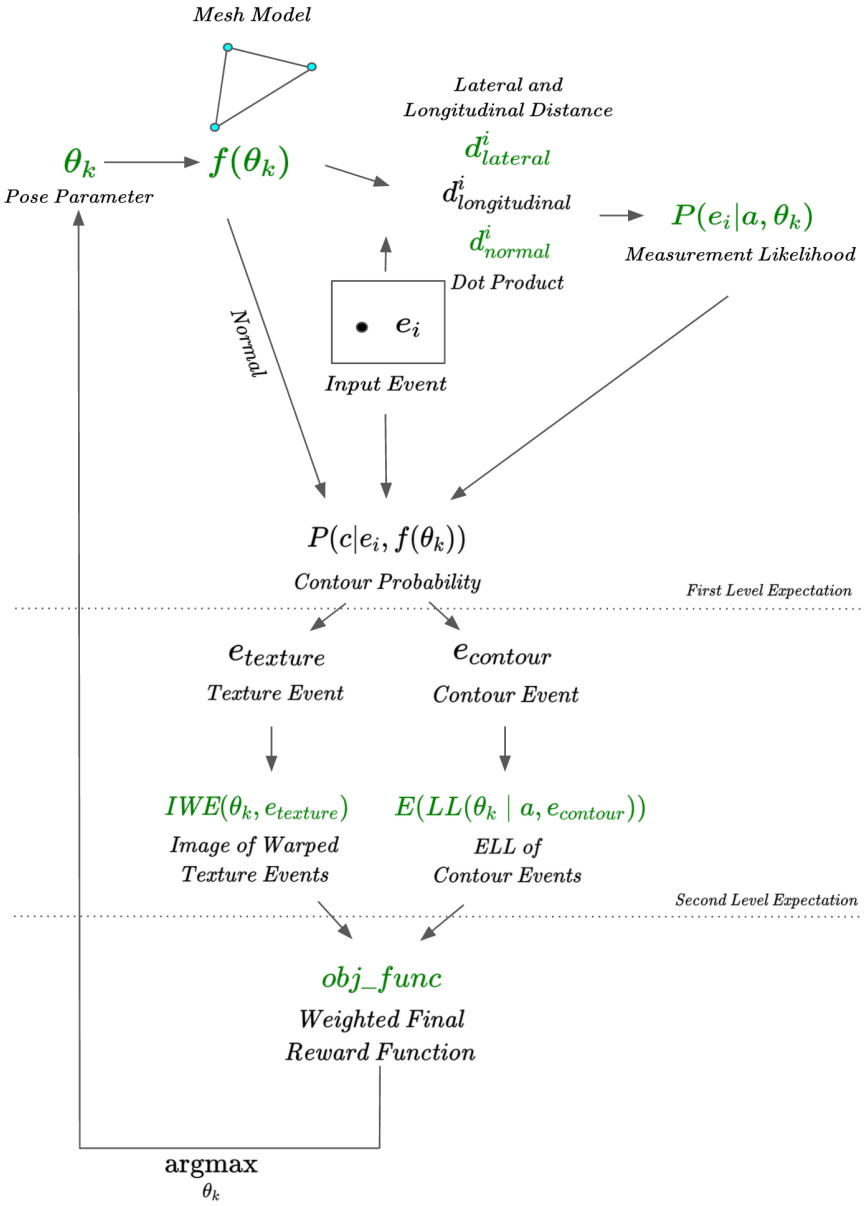


Figure 5.11.: Two-level EM-based tracking framework. At the first level, the distribution of contour situation of events is obtained. At the second level, the distribution of association situation among all faces of events is obtained. Similar to figure 5.6, variables in green contain the gradient flow w.r.t. desired pose θ_k.

5.6. Sliding-Window Optimization

We perform sliding-window optimization in our full tracking framework (Sec. 5.5) to increase the robustness. Figure 5.12 visualizes the combined tracking framework with a sliding window of size $s+1$. As illustrated in the figure, we warp all texture events in spatio-temporal windows $\{W_k, W_{k+1}, ..., W_{k+s}\}$ in the sliding window back to the reference frame to generate the IWE. Then, we use SoSA to measure the sharpness of the IWE. In addition to texture events, we can estimate the expectation of logarithmic association likelihood for contour events in individual spatio-temporal window W_i. It is written as $E(LL(\theta_i|a_i, e_i))$. Thus, the reward function of the whole sliding window is

$$\underset{\theta_k,...,\theta_{k+s}}{\operatorname{argmax}} \left(k_1 \cdot \sum_{i=k}^{k+s} E(LL(\theta_i|a_i, e_i)) + k_2 \cdot r_{SOSA}(IWE(\theta_k, ..., \theta_{k+s})) + k_3 \cdot \sum_{i=k}^{k+s} ||v_i - v_{i-1}||^2) \right), \tag{5.23}$$

where $\{\theta_k, ..., \theta_{k+s}\}$ are desired pose parameters in the sliding window. After the optimization for current sliding window converges, we slide the optimization window forward and initialize the pose parameters with the previous optimization result:

Algorithm 2 Initialization of parameters in sliding window

1: **procedure** INITIALIZATION
2: $\theta_k \leftarrow \theta_{k+1}$
3: $\theta_{k+1} \leftarrow \theta_{k+2}$
4: $\cdots$
5: $\theta_{k+s-1} \leftarrow \theta_{k+s}$
6: $\theta_{k+s} \leftarrow \theta_{k+s-1} + v_{k+s-1} * d_t$

It is worth mentioning that for the last pose parameter θ_{k+s}, there is no previous optimization result available. We use the velocity to initialize that variable as in equation 5.3.

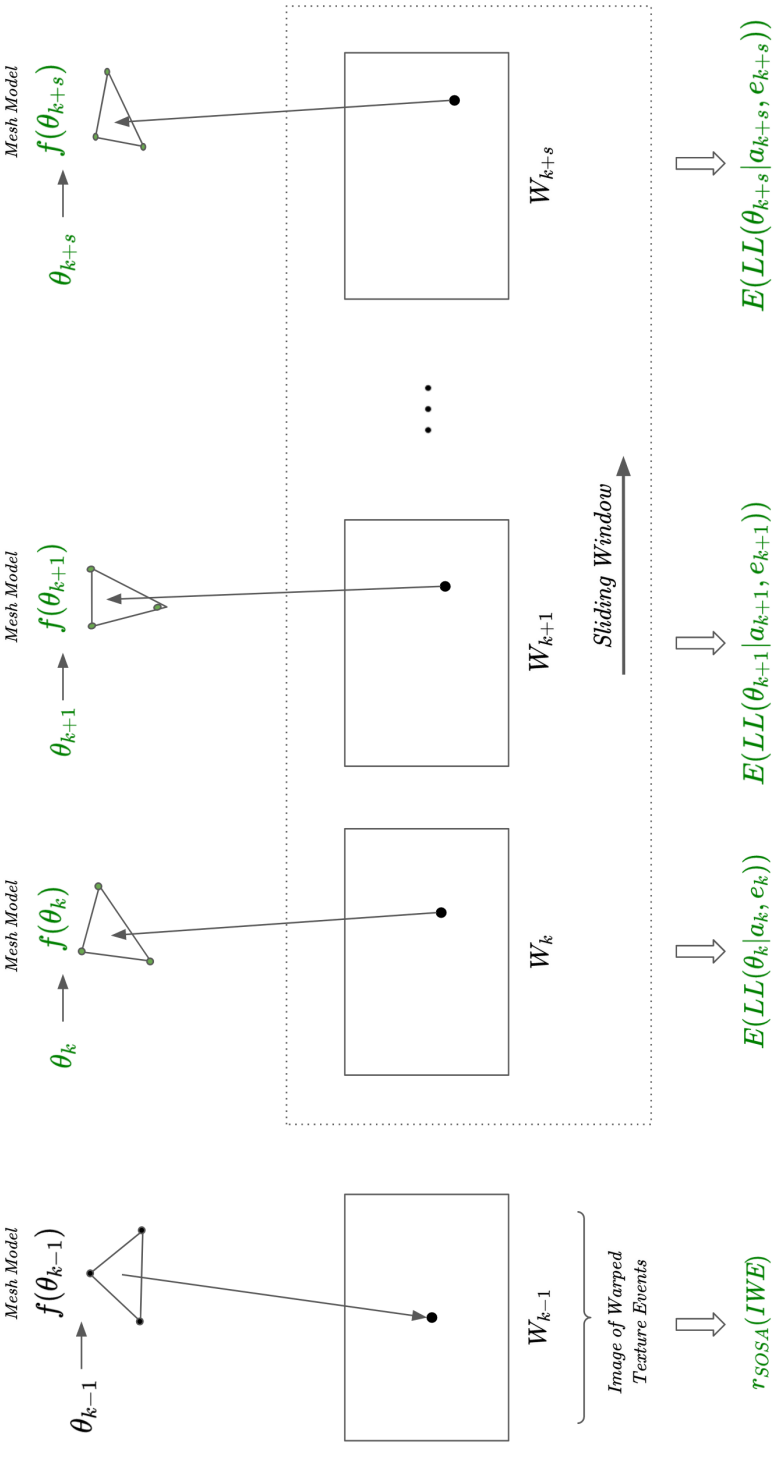


Figure 5.12.: Sliding window optimization in our events-based non-rigid tracking framework. As in figure 5.6, variables containing gradient flow are in green.

6. Experiment and Result

In this section, we report our experiments with the proposed non-rigid tracking methods introduced in chapter 5. Section 6.1 describes details of experiments, including evaluation data, evaluation metrics, and implementation details. Section 6.2 shows the quantitative result on synthetic data while section 6.3 shows the qualitative result on real captured event data. Section 6.4 shows the experiments result of robustness of our approach to different noises and section 6.5 shows the ablation study for the data likelihood. Section 6.6 summarizes the achievements and limitations of our framework.

6.1. Experiment

6.1.1. Evaluation Data

All supported non-rigid object models are introduced in section 3.1. In experiments, we simulated non-rigid motion of the hand (based only on MANO and SMPL-X), non-rigid motion of the human body (based on SMPL-X), and rigid motion of YCB Benchmarks objects, as in figure 6.1.

We simulated non-rigid deformation sequences for the following objects:

- MANO hand model with 45 PCA parameters: as introduced in section 3.1, different poses of MANO hand can be obtained by the full 45-dimensional pose space. Thus, the pose parameter here has the size of $[1, 45]$.
- SMPL-X hand model with 6 PCA parameters: instead of using the full dimension pose space, we can use 6 PCA parameters to control different poses of the hand. The pose parameter here has the size of $[1, 6]$
- SMPL-X body model: similar to MANO hand model, different poses of body model can be obtained by the rotation of 21 body joints. The basic pose parameter for body model has the size of $[21, 3]$. Here, we only use the 3-DoF rotation the left elbow joint to simulate the arm motion.
- Objects YCB Benchmarks: above mentioned models don't contain enough texture. In our experiments, we used a rubik cube which has a simple shape but the rich texture. We simulate and track the rigid motion of the object.

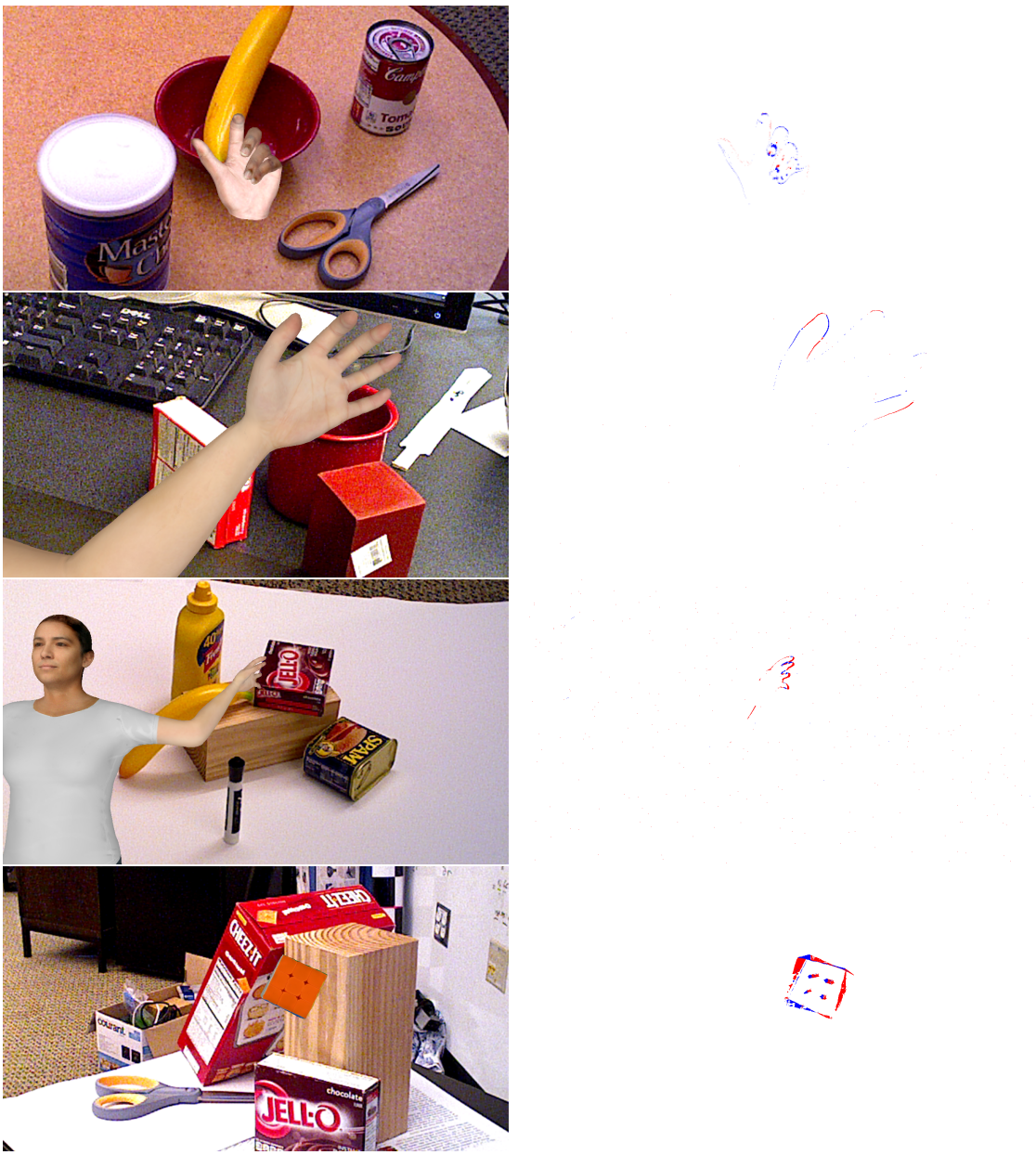


Figure 6.1.: Simulated RGB image (left) and corresponding accumulated events (right). 1^{st} row: MANO hand model. 2^{nd} row: SMPL-X hand model. 3^{rd} row: SMPL-X body model. 4^{th} row: Rubik cube in the YCB Benchmarks.

6.1.2. Sequence description

In addition to different object models, we also simulated with random motion and backgrounds with different textures. For random motion, we simulated sequences with the randomized initial pose and the end pose. Besides, we had random rich-texture images from YCB video dataset [42] as the background. It brings complexity in tracking because the polarity of events is dependent on background color of individual pixels. Simulation with texture background is also more realistic, because a pure color background is extremely hard to get in the real life.

In the data simulation, we randomly draw initial pose, end pose, and background image for each objects to generate the synthetic data. We also add uncertainty to contrast threshold C and salt-and-pepper background noise as described in section 4.4 to close the sim-to-real gap. Here, we draw the contrast threshold from $\mathcal{N}(0.5, 0.0004)$. The thresholds of positive and negative salt-and-pepper noise are set to 1×10^{-5} and 4×10^{-7}, respectively.

6.1.3. Implementation Detail

We implemented our event simulator (Sec. 4) and event-based tracking framework (Sec. 5) in PyTorch and PyTorch3D. Since we used CUDA parallel processing operation (Sec. 3.4.2) to speed up the procedure, we used a NVIDIA 2080Ti GPU in our project.

Besides, it is worth mentioning that the human body and hand are texture-less object and therefore do not generate many texture events in our observation. To increase the computation efficiency, we treat all events in those sequences as contour events and only apply EM-contour-tracker in section 5.3. Thus we only report results for EM-contour-tracking for these sequences. For texture-rich objects, which generate both texture events and contour events, we use the full tracking framework in section 5.5.

As illustrated in figure 5.3, we stacked 300 events into each spatio-temporal window for synthetic event data with resolution of 1280×720. For real event data captured by Davis camera [1], it has a resolution of 240×180 and therefore we stacked 100 events in each window.

In the experiments of objects tracking, we use optuna [1] to tune hyperparameters (Sec. 3.3) for different scenarios. The detail of parameter tuning and optimzied hyperparameters can be found in Appendix A.2. As an optimization-based method, we use Adam [18] as the optimizer.

[1] https://inivation.com/dvp/

6.1.4. Evaluation Metrics

We have different evaluation metrics to perform the quantitative analysis of our tracking framework. For non-rigid objects e.g. human body and hand, there is the joint regressor available. A joint regressor can regress the mesh to skeleton joints. For MANO hand [34] and SMPL body [25], we can get 3D joints from the pose parameters. Therefore, we evaluate performance on the body and hand model with **MPJPE** (**M**ean **P**er **J**oint **P**osition **E**rror)

$$\text{MPJPE} = \frac{1}{N}\sum_{i=1}^{N}\frac{\left\|J_{GT}^{i} - J_{rec}^{i}\right\|_F}{\#Joints}, \tag{6.1}$$

where N is the number of events temporal window; J_{GT} and J_{rec} denote the ground-truth and reconstructed 3D joint locations, respectively. $\#Joints$ is the number of involved joints: in hand tracking experiments, the joints are 15 hand skeleton joints. In the body tracking experiments, we only allow the 3-DoF rotation of the left elbow joint, which brings only motion of the left hand skeleton and the left wrist. Thus, the number of joints $\#J$ here is 16. Finally, $\|\cdot\|_F$ represents for the Frobenius norm, defined as

$$\|A\|_F := \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}\left|a_{ij}\right|^2}. \tag{6.2}$$

In addition to the average 3D joints error, we also introduce the **PCK** (**P**ercentage **C**orrect **K**eypoints) and the **AUC** (**A**rea **u**nder **C**urve) to perform quantitative analysis in more intuitive diagrams as figure 6.2. 3D-PCK describes the percentage of joints which have MPJPE less than the error threshold. AUC calculates the area under the 3D-PCK curve. The larger the AUC, the better the tracking performance.

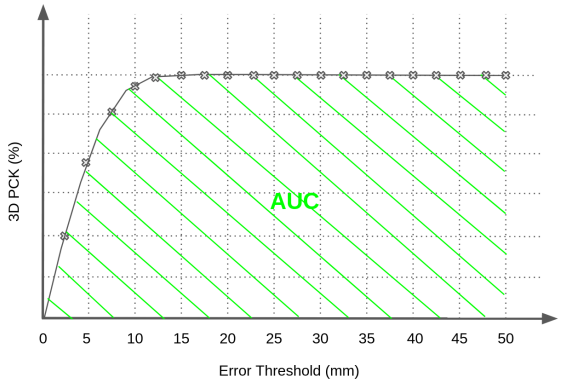


Figure 6.2.: An example of a 3D-PCK curve @50mm and the corresponding AUC.

6.2. Evaluation on Synthetic Data

This section presents the tracking result on various objects introduced in 6.1.1. The tracking result of non-rigid objects, e.g. hand and body, is shown in section 6.2.1. We also evaluated our methods in terms of rigid tracking, including translation and rotation. The result appears in section 6.2.2.

6.2.1. Non-Rigid Tracking

MANO Hand Tracking

We evaluated our non-rigid tracking methods here using 30 synthetic motion sequences of MANO hand model. Each sequence has the random initial pose, random end pose, and random background texture to simulate various hand motions in different scenarios. As introduced in section 6.1.1, the pose of the MANO hand model is parameterized by 45-dimensional parameters in PCA pose space. In the simulation, each pose parameter was randomly sampled from $[-\frac{\pi}{2}, +\frac{\pi}{2}]$. We show a set of samples from this pose range in section B.1. In the tracking process, we optimized 45 PCA pose parameters to perform the tracking of the MANO hand model. We used the evaluation tool to generate the qualitative result (Fig. 6.3) and quantitative result (Tab. 6.1). The quantitative evaluation metrics are explained in section 6.1.4.

The input of our method is an asynchronous events stream split in several spatio-temporal windows. Each window contains a fixed number of 300 events and have therefore dynamic temporal length. We visualize the events in the first and the last spatio-temporal window of a demo sequence in figure 6.3a and 6.3b, respectively. As shown in figures, almost all events during the hand motion are generated by contour mesh faces. As explained in section 6.1.3, we only deployed the EM-contour-tracker (Sec. 5.3) on MANO motion sequences and report the corresponding results.

A qualitative evaluation of the tracking performance on the demo sequence is visualized in figure 6.3. In the demo sequence, the background scene is texture-rich. In addition, all 4 fingers bend together which means the demo sequence contains a large hand motion. The ground-truth hand deformation is presented intuitively in figure 6.3c: the initial pose and the end pose are linearly blended, while the beginning pose is semi-transparent. Figures in the first row (Fig. 6.3a, 6.3b) show the sparse events input of our tracking framework. The initial and end reconstructed hand poses are visualized in figure 6.3d. Again, the initial reconstructed hand is semi-transparent. The qualitative result shows that given the highly sparse events input, our EM-contour-tracker can track the hand motion accurately. More qualitative results can be found in the supplementary material.

(a) Accumulated events in the first spatio-temporal window.

(b) Accumulated events in the last spatio-temporal window.

(c) Ground-truth MANO hand motion of the demo sequence. Hand before the deformation is semi-transparent.

(d) Reconstructed MANO hand motion of the demo sequence. Hand before the deformation is semi-transparent.

Figure 6.3.: Events in individual spatio-temporal windows (above); Ground-truth and reconstructed MANO pose before and after the motion (below), with initial hand pose semi-transparent.

In addition to the qualitative result, we also present the quantitative tracking performance in Table 6.1. The mean MPJPE is 4.52 *mm* and the mean AUC with the maximal threshold at 50 *mm* is 90.67%. As shown in the table, the tracking performance is different among all sequences depending on the complexity of the motions. The best performance has 2.23 *mm* MPJPE and 95.09% AUC under 3D-PCK curve, while The worst performance has 11.13 *mm* MPJPE and 77.64% AUC. The median performance for all sequences is 4.27 *mm* for MPJPE and 91.20% for AUC. Besides, we compare our approach with Nehvi's method [26], which tracks the hand motion with asynchronous event stream. It is also an optimization-based method and has the same object (MANO hand) as our work. The quantitative comparison and 3D-PCK curve appear in Table 6.2 and figure 6.4.

MANO Hand Tracking Experiments		
Sequence Index	MPJPE (mm)	AUC @50mm (%)
1	11.1299	77.63586
2	6.66957	86.46130
3	3.75380	92.16196
4	4.27448	91.17013
5	4.50295	90.71420
6	2.90929	93.76926
7	6.21236	87.24349
8	4.02778	91.65868
9	3.23277	93.24739
10	3.06880	93.50493
11	6.51410	86.73701
12	2.23082	95.09228
13	5.56842	88.59130
14	5.48163	88.88439
15	3.58297	92.43402
16	4.29410	91.03303
17	4.00633	91.71540
18	4.26295	91.19046
19	3.70786	92.29227
20	4.22251	91.27442
21	4.49148	90.78849
22	4.34041	91.04166
23	4.33652	91.29684
24	4.35076	90.89650
25	4.32651	91.04011
26	3.73972	92.12859
27	4.23436	91.21348
28	3.89701	91.86095
29	3.90512	91.99237
30	4.34744	91.04467
Average	4.52076	90.67052101
Median	4.26872	91.20197
Best	2.23082	95.09228
Worst	11.12999	77.63586

Table 6.1.: Quantitative result of EM-contour-tracker on MANO hand motion sequences.

We compare our approach quantitatively on synthetic event data with Nehvi's event-based MANO hand tracking method [26], which is based on the event generation model. The background theory of Nehvi's method is explained in section 2.4.1. To ensure the fairness in the comparison, we tuned the hyperparmeters in Nehvi's framework and in our framework using ground-truth label of synthetic data. The quantitative result in MPJPE and AUC under the 3D-PCK curve show intuitively that our EM-contour-tracker outperforms Nehvi's approach on the MANO hand tracking scenario.

	mean MPJPE	median MPJPE	mean AUC	median AUC
Nehvi et al. [26]	11.61 *mm*	10.85 *mm*	77.59%	78.68%
Ours	**4.52 mm**	**4.27 mm**	**90.67%**	**91.20%**

Table 6.2.: Results on synthetic MANO hand sequences

Given the qualitative result in figure 6.3 and the quantitative result in Table 6.1, we can draw the conclusion on our EM-contour-tracker: although the input of our tracking method is the sparse events stream shown in figure 6.3a and 6.3b, it can still reconstruct the motion of the hand accurately. In other words, the event-based non-rigid tracking method we proposed can extract the deformation information accurately from the events stream, which is the limited input information. Last but not least, a quantitative comparison of the tracking performance between our tracking method and state-of-the-art event-based MANO hand tracking method [23, 35] is presented in section 6.3.

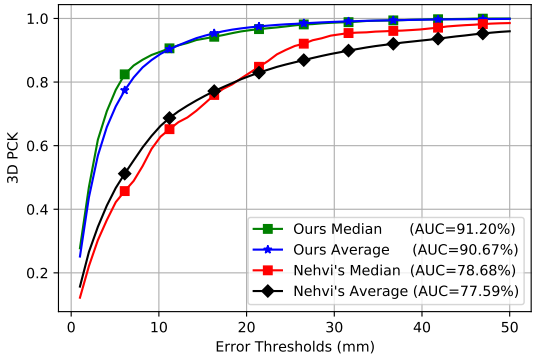


Figure 6.4.: 3D-PCK curve on synthetic hand motion reconstruction

SMPL-X Hand Tracking

For the evaluation of our tracking method on SMPL-X hand tracking, we generated synthetic motion sequences based on SMPL-X hand model. Similar as the MANO sequences mentioned previously, each sequence generated here has the random initial and end pose to simulate different hand motion. As introduced in section 6.1.1, the pose of a SMPL-X hand model is parametrized by 6 parameters in the PCA pose space. In the simulation, all 6 parameters were sampled from a uniform distribution with the range of $[-\frac{\pi}{2}, +\frac{\pi}{2}]$, similar to MANO evaluation sequences. In the tracking process, we optimized 6 pose parameters to perform the tracking of the SMPL-X hand model. We present the qualitative and the quantitative results in this section.

Our tracking method takes only events as input, as shown in figures 6.5a and 6.5b. Similar as the simulated MANO sequences, the events during the motion of SMPL-X are also generated mainly by contour mesh faces. Thus, we only deployed EM-contour-tracker (Sec. 5.3) to perform the hand tracking, as explained in section 6.1.3.

A qualitative tracking result of the SMPL-X hand motion demo sequence is presented in figure 6.5. The input events of the demo sequence are shown in the first row. The ground-truth hand motion is illustrated in figure 6.5c, and the reconstructed hand motion using our tracking method is in 6.5d. In both figures, the initial hand poses are semi-transparent. The qualitative result shows that the reconstructed end hand pose is exactly similar as the ground-truth end pose. According to that, we can say that our tracking method works well on the tracking task of the SMPL-X hand motion.

In addition to the qualitative result of the demo sequence, we show the quantitative performance of our approach for all sequences in Table 6.3. We also visualize the 3D-PCK curve in figure 6.6a. Our approach on the SMPL-X hand motion sequences has the average MPJPE of 1.11 *mm* and has 96.38% AUC under 3D-PCK curve. The median performance among all sequences shows that our approach has 0.76 *mm* MPJPE and 97.27% AUC. The sequence 7 has the best performance with MPJPE of 0.12 *mm*, which means each joint has only 0.12 *mm* error w.r.t. ground-truth position on average and it is a tiny reconstruction error. The sequence 5 has the worst performance. However, it has the MPJPE of 4.37 *mm*, which is still a small reconstruction error. Thus, we conclude that our approach is appropriate for the tracking of SMPL-X hand motion.

We can compare the performance of our approach on MANO hand motion reconstruction (Tab. 6.1) and SMPL-X hand motion reconstruction (Tab. 6.3).

Motion Sequences	Mean MPJPE (mm)	Median MPJPE (mm)
MANO Hand	4.52076	4.26872
SMPL-X Hand	**1.10835**	**0.76377**

SMPL-X Hand Tracking Experiments		
Sequence Index	MPJPE (mm)	AUC @50mm (%)
1	0.64099	97.61057
2	1.33952	96.29861
3	1.83420	95.06832
4	0.54154	97.67787
5	4.37342	90.53448
6	1.95539	95.08425
7	0.12334	97.64257
8	0.68217	97.49362
9	0.61206	97.04586
10	1.40122	96.24539
11	0.37895	97.86953
12	1.09043	96.86055
13	1.07047	96.88518
14	0.80958	90.26266
15	0.86362	97.21003
16	0.32528	97.92473
17	2.30138	94.56938
18	1.29191	96.49254
19	0.80604	97.32147
20	1.05453	96.80513
21	0.33178	97.74533
22	0.60639	97.59841
23	0.28463	97.95401
24	0.72150	97.41465
25	4.21576	90.85210
26	0.56281	97.67582
27	0.51449	97.77622
28	0.52726	97.73778
29	1.51441	96.08000
30	0.47584	97.70191
Average	1.10835	96.38130
Median	0.76377	97.26575
Best	0.12334	97.95401
Worst	4.3734	90.26266

Table 6.3.: Qualitative result of EM-contour-tracker on SMPL-X hand motion sequences.

(a) Events in the first spatio-temporal window of SMPL-X hand motion.

(b) Events in the last spatio-temporal window of SMPL-X hand motion.

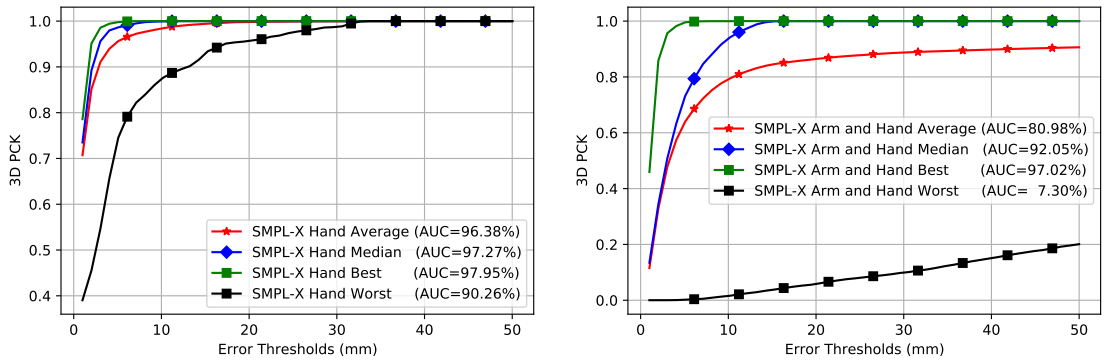
(c) Ground-truth SMPL-X hand poses of the demo sequence.

(d) Reconstructed SMPL-X hand poses of the demo sequence.

Figure 6.5.: Events in individual spatio-temporal windows (above); Ground-truth and reconstructed SMPL-X hand pose before and after the motion (below), with the initial hand pose semi-transparent.

The result in the table above shows that our approach achieves better performance on SMPL-X hand model than single MANO hand model. It is worth mentioning that the results are not from the same motion sequences, but the PCA pose parameters are randomly drawn from the uniform distribution with the same range. Our analysis is that the difference can be caused by the dimension of pose parameters. As introduced in section 6.1.1, MANO model is controlled by 45-dimensional pose parameters which span the whole hand pose space, while SMPL-X hand model only uses the first 6 PCA parameters. Essentially, the SMPL-X hand motion sequences have 6 pose parameters to optimize, while the MANO hand motion sequences have 45 pose parameters to optimize, which is more complex for the optimizer. Thus, the comparison above is unfair and does not contain much information. According to Romero, Tzionas, and Black [34], the 6 first pose parameters span about 81% hand pose space. Therefore, SMPL-X suffice to model most common hand poses. In practice, we can use SMPL-X

hand to model and reconstruct the hand motion.



(a) SMPL-X hand.

(b) SMPL-X arm and hand.

Figure 6.6.: 3D-PCK curve @50*mm* of SMPL-X hand (a) as well as SMPL-X body and hand (b) reconstruction.

SMPL-X Arm and Hand Tracking

Finally, we show the tracking performance on combined motion of SMPL-X body and hand model. The hand pose is parametrized by 6 PCA parameters, while the body pose is parametrized by the 3 rotation parameters of the left elbow joint. The rotation around the elbow joint brings motion of the left arm and the left hand. Thus, the motion here can be seen as the reconstruction of deformation, rotation, and translation of the hand. In this experiment, the hand and the elbow joint have their motion simultaneously. In the simulation, we sampled the hand pose parameters from a uniform distribution with range of $[-\frac{\pi}{2}, +\frac{\pi}{2}]$, and sampled the elbow joint rotation parameters from a uniform distribution with the range of $[0, 1]$. We show a set of samples from this pose range in section B.3. In the tracking experiment, we jointly optimized for 3 rotation parameters of the left elbow and 6 PCA parameters of the left hand to perform the tracking of the SMPL-X arm and hand motion.

Our approach takes asynchronous event stream as input and stacks 300 events into each spatio-temporal window to reduce the computation time. The accumulated events in the first and the last spatio-temporal window (Sec. 5.2) are visualized in figures 6.7a and 6.7b, respectively. Similar to previous experiments, we only deploy EM-contour-tracker (Sec. 5.3) because almost all events are generated by contour mesh faces in our observation.

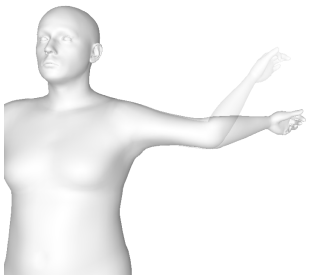
We visualize the qualitative tracking result on figure 6.7. The input event stream is illustrated in the first row, while the ground-truth and the reconstructed arm and hand

motion are shown in figure 6.7c and 6.7d, respectively. The initial pose of the arm and hand are in transparent. The qualitative result shows that our tracking method can reconstruct the motion of arm and hand in the demo sequence accurately.

(a) Events in the first spatio-temporal window of SMPL-X body and hand motion.

(b) Events in the last spatio-temporal window of SMPL-X body and hand motion.

(c) Ground-truth SMPL-X body and hand poses of the demo sequence.



(d) Reconstructed SMPL-X body and hand poses of the demo sequence.

Figure 6.7.: Events in individual spatio-temporal windows (above); Ground-truth and reconstructed SMPL-X body and hand pose before and after the motion (below), with the initial pose semi-transparent.

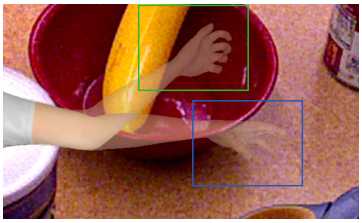
In addition to the qualitative result of the demo sequence, we show the quantitative evaluation result of all sequences of the SMPL-X arm and hand motion in Table 6.4. The corresponding 3D-PCK curve is shown in figure 6.6b. Here, our tracker achieves the average MPJPE of 15.39 mm and 80.98% AUC under the 3D-PCK curve. Besides, we also show the median reconstruction performance of our method among all test sequences: it achieves 3.92 mm as MPJPE and 92% as AUC. The median result shows that our approach can accurately reconstruct the combined arm motion and the hand motion.

Apparently, our approach fails in sequence 1 and sequence 9. We investigated into why our approach fails in these two sequences. We visualize the groud-truth images,

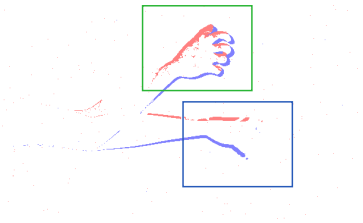
SMPL-X Arm and Hand Tracking Experiments		
Sequence Index	MPJPE (mm)	AUC @50mm $(\%)$
1	114.93061	8.44076
2	4.13422	91.64820
3	2.02904	95.64299
4	6.02724	87.93815
5	1.24610	97.01994
6	6.52830	86.88121
7	3.06240	93.68731
8	8.78230	82.41097
9	83.72781	7.29643
10	12.59070	74.82034
11	12.24150	76.21525
12	1.96041	95.82178
13	3.50925	92.93285
14	1.89513	96.10460
15	3.71677	92.54633
16	3.69315	92.44335
17	2.18549	95.42289
18	4.72697	90.45637
Average	15.38819	80.98498
Median	3.92549	92.04578
Best	1.24610	7.29643
Worst	114.93061	97.01994

Table 6.4.: Qualitative result of EM-contour-tracker on SMPL-X arm and hand motion sequences.

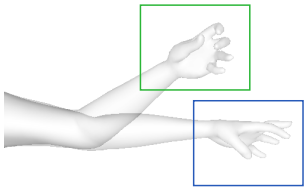
input event stream, and reconstructed arm and hand in figure 6.8. The initial pose is in the blue bounding box, and the final pose is in the green bounding box. Figure 6.8b shows that the hand at the initial pose does not generate valid events. The reason can be inferred from figure 6.8a: the fingers at the initial pose has the similar color as the background. According to the event generation model (Eq. 4.1), no events are generated by the motion of fingers. The lack of events leads to the failure case of our approach on sequence 1. However, as illustrated in figure 6.8c, our EM-contour-tracker can still reconstruct the arm motion, because the events of arm motion are generated as usual.



(a) Ground-truth motion in sequence 1.



(b) Events in sequence 1.



(c) Reconstructed arm and hand pose in sequence 1.

Figure 6.8.: Analysis of the failure case of our EM-contour-tracker on sequence 1 in SMPL-X arm and hard tracking

In section 6.2.1, we show the qualitative result of our tracking method on MANO hand motion, SMPL-X hand motion, and SMPL-X body motion sequences. As illustrated in figures of input events (Fig. 6.3a, 6.3b, 6.5a, 6.5b, 6.7a, 6.7b), we can observe that those events are mainly generated by the motion of contour of mesh faces. Thus, we deployed the EM-contour-tracker (Sec. 5.3) to perform the event-based tracking of above mentioned non-rigid objects. Generally, our method has a solid performance on non-rigid objects, e.g. arm and body, only based on sparse events input.

6.2.2. Rigid Tracking

Because the body-related non-rigid objects don't contain much texture, the events are not likely to be generated by texture mesh faces. Thus, we also perform the rigid tracking of the objects in YCB Benchmarks. As illustrated in the last row of figure 6.1, the rubik cube can generate texture events by the hole on the surface. Besides, events generated because of the motion of boundary edge are still considered as contour events. Thus, we deploy our combined framework (Sec. 5.5) to perform the event-based tracking of rigid-motion sequences of the rubik cube.

During our experiments, we found that the our tracking method is biased towards

the translation along z-axis in 6-DoF rigid objects tracking. We show the analysis of this phenomenon at the end of the section. Thus, we simplify the 6-DoF tracking to 3-DoF planar motion tracking, namely fix the depth value of the object. We use the **ATE** (**A**bsolute **T**rajectory **E**rror) as the evaluation metric:

$$\mathrm{ATE}_{\mathrm{trans}} = \frac{1}{m}\left(\sum_{i=1}^{m}\left\|\mathbf{p}_{\mathbf{rec}}^{\mathbf{i}} - \mathbf{p}_{\mathbf{gt}}^{\mathbf{i}}\right\|^2\right)^{\frac{1}{2}}, \tag{6.3}$$

$$\mathrm{ATE}_{\mathrm{rot}} = \frac{1}{m}\sum_{i=1}^{m}\left(\angle\left(R_{rec}^{i}\cdot {R_{gt}^{i}}^{-1}\right)\right) \tag{6.4}$$

We evaluate our tracking framework with 3-DoF motion sequences, combining the 1-DoF planar rotation and the 2-DoF planar translation. We show the root mean squared translation error and the mean rotation error in Table 6.5. The results shows that our proposed tracking framework (Sec. 5.5) performs well in the planar rotation tracking only using event stream.

	$\mathrm{ATE}_{\mathrm{trans}}$ (mm)	$\mathrm{ATE}_{\mathrm{rot}}$ $(^\circ)$
3-DoF Planar Motion	1.9667	5.9530

Table 6.5.: Results on synthetic 3-DoF planar motion sequences of rigid objects.

As mentioned before, our proposed tracking framework (Sec. 5.5) is biased towards the translation along z-axis. Our analysis is that the contrast maximization (CM) term (Sec. 5.4) brings this effect. It can be explained by the area of projected events as shown in figure 5.9: the larger z-value for the object is optimized, the smaller area all events are warped to. When the number of warped events does not change, the smaller projection area means more concentrated events and therefore a higher sharpness of the IWE.

Theoretically, our EM-contour-tracking term can help in avoiding the bias in the z-direction. As introduced in section 5.5, we use the dot product between the mesh face and the event bearing vector to decide whether the event is classified as contour events or texture events. Basically, EM term (Eq. 5.17) has smaller reward when object depth is smaller, because less events will be classified as contour events. In the loss landscape (Fig. 6.9), the EM term can partly avoid the bias in translation along the z-direction. However, the ground-truth pose is still not at the local minimum of our objective function. We believe that the current combined framework (Sec. 5.5) needs more regulation to perform the 6-DoF tracking.

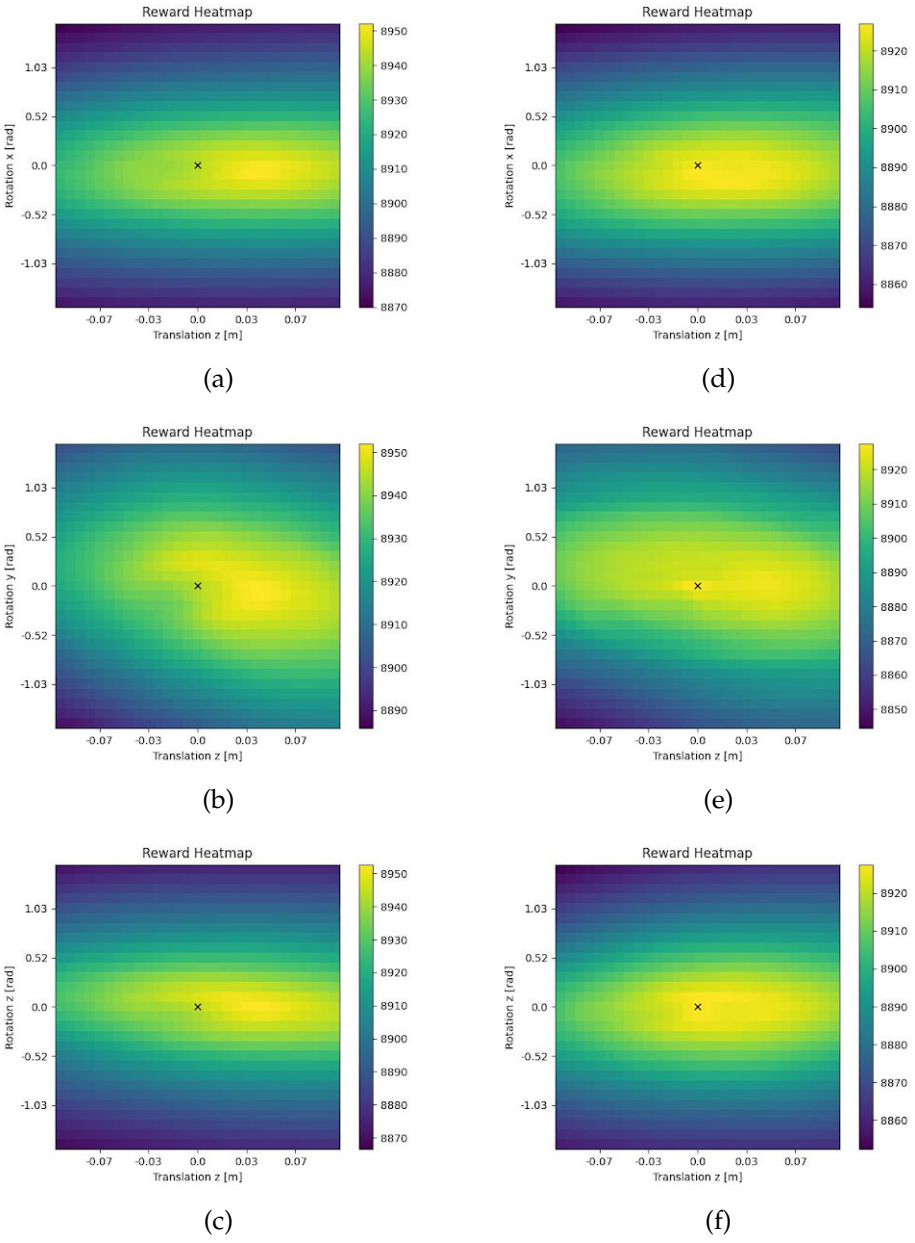


Figure 6.9.: Loss landscape of CM (Left) and combined CM + EM (right). It shows that our proposed full tracking method, which combines EM-based term with CM-based term, can partly overcome the bias towards translation along positive-Z direction. For all plots, X-axis represents the translation along Z. Y-axis represents rotation in x, rotation in y, and rotation in z for (a) and (d), (b) and (e), (c) and (f), respectively. Cross represents the ground-truth pose.

6.3. Real Data Experiment

In addition to experiments on synthetic sequences, we did hand tracking experiments on real data sequences. We used Davis 240C camera [2] for recording events at 1 MHz as the input of our tracking framework and recording intensity frames at 24 FPS as the visual reference.

Our method assumes that the initial parameters are known. For the initialization of the MANO model, we deployed the pre-trained model of MeshGraphormer [23] to infer the mesh model from the grayscale image. Then, we minimized the Chamfer distance between the inferred mesh and the predicted mesh to optimize for shape and pose parameters. We manually adjusted the global rotation and translation of the hand model to align the rendered hand image with the captured hand image.

(a) Captured hand image (b) Rendered hand image (c) Aligned hand images with adjusted R and T

Figure 6.10.: Initial MANO hand model parameters estimation of real demo sequence.

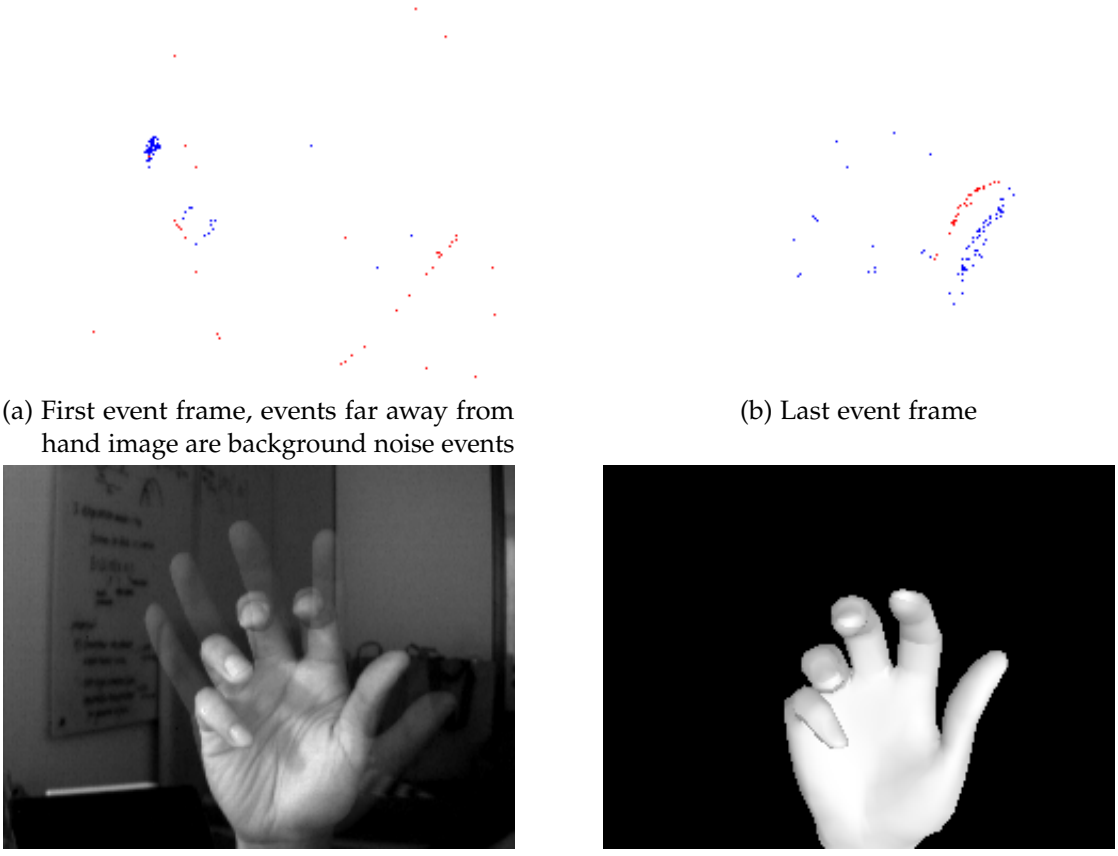
The noise is an unavoidable issue for event cameras. In our experiments, we observed a huge amount of noisy events which are not generated by the hand motion. Thus, we activated the Davis noise filters while capturing the event data. However, there are still background noise events captured as shown in figure 6.11a. To deal with these events, we have a filtering mechanism in our tracking framework: the event whose unprojecting ray having the lateral distance to the closest mesh face larger than a threshold is marked as a noisy event. In other words, the noisy event is far away from the mesh and is unlikely to be caused by the mesh model. This method effectively blocks noisy events in the final tracking framework.

Similar as hand tracking experiments on synthetic sequences, we deployed EM-contour-tracker here. Each spatio-temporal window contains 100 events. We show the first and the last event frame on the first row, and show the qualitative tracking result

[2] https://inivation.com/wp-content/uploads/2019/08/DAVIS240.pdf

on the second row in figure 6.11. The qualitative result shows that our tracking method performs well on captured real event data.

Compared to tracking results on synthetic sequences in figure 6.3, the tracking performance on real event data is worse. Our analysis is that the estimated initial MANO parameters is not 100% accurate as in synthetic sequences, and the noise filter cannot filter all noisy events which are not generated by the hand motion. Besides, we don't have ground-truth pose parameters in real data sequences to tune the hyperparameters of our tracking framework. These limitations can introduce the additional error into the final tracking performance.



(a) First event frame, events far away from hand image are background noise events

(b) Last event frame

(c) Captured hand motion

(d) Reconstructed hand motion

Figure 6.11.: Input event buffers (first row); Captured hand motion and reconstructed hand motion (second row) of real data demo sequence.

We show the qualitative comparison between our method and Nehvi's appraoch [26], EventHands [35], and MeshGraphormer [23] on real captured data in figure 6.12.

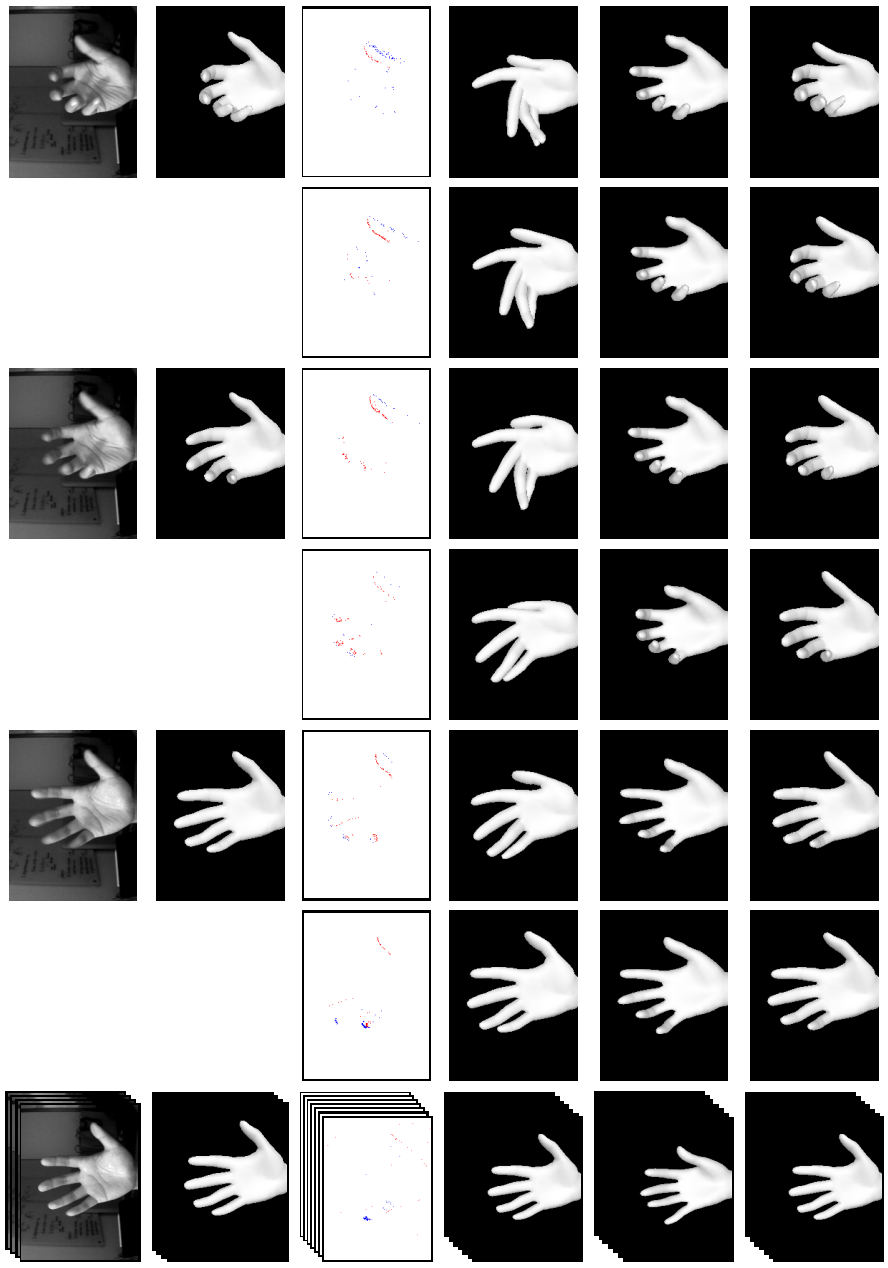


Figure 6.12.: Qualitative comparison with state-of-the-art image-based and event-based hand reconstruction methods. Note that Lin et al. [23] infers MANO mesh models from intensity images; Rudnev et al. [35] infers 6 main PCA pose parameters of MANO model from event stream; Nehvi et al. [26] and ours approach optimize for full dimension (45) PCA pose parameters of MANO model from event stream.

The results show that our approach qualitatively has the best reconstruction result compared to event-based method [26, 35]. Besides, our method doesn't suffer from the motion blur and has the reconstruction with a very high temporal resolution (125 event frames v.s. 3 intensity frames). It shows that our approach outperforms Meshgraphoermer [23] in these two metrics.

6.4. Robustness to Noise

We show the robustness of our proposed EM-contour-tracker (Sec. 5.3) to different classes of noise. We demonstrate the investigation in the SMPL-X hand motion reconstruction.

Robustness to initial template noise

As a template-based approach, the initial template is essential for our method. We first demonstrate that our approach is robust to the noise of the initial template. Here, we sample 6-dimensional initial pose parameters of the SMPL-X hand model from a Gaussian distribution with the mean of ground-truth values and different level of standard deviations.

The 3D-PCK curve and AUC value of each standard deviation are in Fig. 6.13. The result shows that our approach still has the AUC of 86% when the standard deviation is already 0.8, which illustrates that the proposed method is robust to the noise of initial template.

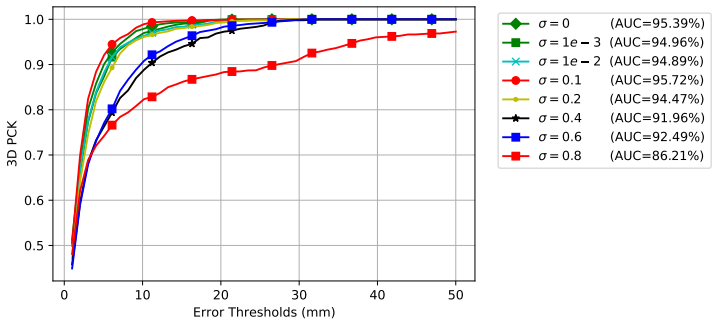
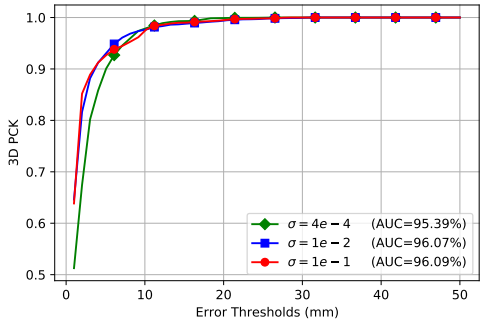


Figure 6.13.: Robustness to different level of initial template noise

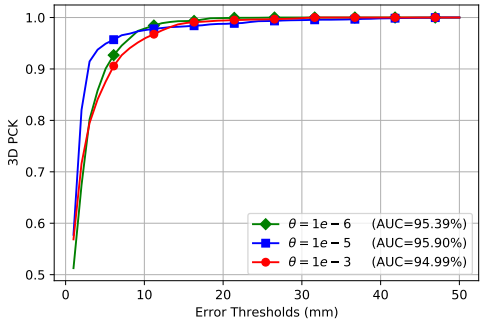
Robustness to uncertainty of contrast threshold

A real event-based camera has uncertainty in the contrast threshold [12], which brings noise in the captured event stream. In our simulator, we already model the uncertainty by sample the contrast threshold from $\mathcal{N}(0.5, 0.0004)$ (Sec. 4.4). To demonstrate that our approach is robust to different level of uncertainty in the contrast threshold, we use different levels of standard deviation for contrast threshold sampling.

We show the 3D-PCK curves and AUC values of different noise levels in Fig. 6.14a. The result shows that our approach is not much affected by the uncertainty of the contrast threshold.



(a) Robustness to different level of contrast threshold uncertainty.



(b) Robustness to different level of salt-and-pepper noise.

Figure 6.14.: Robustness to different level of uncertainty in contrast threshold and of salt-and-pepper noise.

Robustness to salt-and-pepper noise

An event-based camera suffers from the salt-and-pepper noise [12]. We model the salt-and-pepper noise by sampling the probability at each pixel from a uniform distribution and compare with a threshold (Sec. 4.4).To model the different level of the salt-and-pepper noise, we select different threshold θ.

We show the 3D-PCK curves and AUC values of different thresholds in figure 6.14b. It shows that our EM-contour-tracker has solid performance to different amount of salt-and-pepper noise. Our method still achieves considerable performance by the threshold of 10^{-3}, which corresponds to a high amounts of salt-and-pepper noise and it is unlikely happen in the real-world scenario.

Given the quantitative evaluation in figure 6.13 and 6.14, we can draw the conclusion that our approach is robust to initial template noise, uncertainty on contrast threshold,

and salt-and-pepper noise. Our analysis is that, we have advantages in robustness to noise as a 3D-geometric based approach. For example, we only consider event who have the minimal lateral distance (Fig. 5.4) smaller than a threshold. Otherwise it is considered as an outlier event. It effectively reject the salt-and-pepper noise on the background.

6.5. Ablation Study

In the ablation study, we investigate variants of the data likelihood term formulated for E-step (Eq. 5.8) and M-step (Eq. 5.9). The data likelihood of E-step is formulated by the lateral probability, the longitudinal probability, and the contour probability. In the ablation study, we formulate the data likelihood in the E-step by either lateral probability and longitudinal probability:

$$P\left(e_i|f_j,\theta_k\right) \propto P_{lateral} \cdot P_{longitudinal}, \tag{6.5}$$

or the lateral probability and the contour probability:

$$P\left(e_i|f_j,\theta_k\right) \propto P_{lateral} \cdot P_{contour}. \tag{6.6}$$

The proposed data likelihood in the M-step is formulated by the lateral probability and the longitudinal probability. In the ablation study, we formulate the data likelihood only with the lateral probability:

$$P\left(e_i|f_j,\theta_k\right) \propto P_{lateral}. \tag{6.7}$$

We demonstrate the ablation study in the SMPL-X hand motion reconstruction. The quantitative results of above mentioned variants are shown in Table 6.6.

	MPJPE (mm)	AUC $(\%)$
E3M2 (Eq. 5.8, 5.9)	1.5289	95.9308
$\text{E2}_{\text{normal}}$M2 (Eq. 6.6, 5.9)	1.6523	93.5601
$\text{E2}_{\text{longitudinal}}$M2 (Eq. 6.5, 5.9)	2.2500	92.7352
$\text{E3M1}_{\text{lateral}}$ (Eq. 5.8, 6.7)	1.9891	92.8573

Table 6.6.: Ablation Study on probability terms of the data likelihood in the E-step and the M-step

The quantitative results in table above show that the contour probability is essential for the formulation of the data likelihood term both in the E-step and the M-step. It also shows that introducing longitudinal probability in the E-step can slightly improve the performance. Thus, our current data likelihood formulation (Eq. 5.8, 5.9) is the best variant on the SMPL-X hand sequences.

6.6. Conclusion

Here, we summarize the achievement and the limitation of our proposed approach (Sec. 5.4, 5.3, 5.5) given the experiment results in previous sections. Section 6.2.1 presents the performance of our EM-contour-tracker on the human arm and hand tracking. Intuitively, arm and hand do not contain much texture and therefore the most events are contour events generated by the motion of boundary edges. The quantitative and qualitative results show that our EM-contour-tracker accurately reconstructs the hand and arm motion. We also show the qualitative comparison with state-of-the-art event-based [35] and RGB-based [23] hand tracking and reconstruction methods in section 6.3. It shows that our approach outperforms the state-of-the-art event-based hand tracking methods in accuracy. Besides, our approach does not suffer from motion blur in high speed motion, and can have much higher temporal resolution reconstruction result than the RGB-based method. In section 6.4, we show that our approach is robust to the initial template noise, the uncertainty of the event generation threshold, and the salt-and-pepper noise. The robustness is essential for our framework to perform well on real event data. Thus, we draw the conclusion that our EM-based contour tracking framework works well and is suitable in human hand tracking.

We show the performance of the combined contrast maximization and EM-contour-tracking framework on rigid object tracking. Although results in Table 6.5 show that our full tracking framework performs well in 3-DoF motion tracking, the CM term has bias towards the translation in z-axis and is limited to perform the full 6-DoF motion tracking. We analyze the bias in section 6.2.2 and figure 6.9 shows that our proposed EM term can partly overcome the bias. The tracking of the full 6-DoF motion is one of the future work of our approach.

7. Overview and Outlook

In this chapter, a summary and some continuing concepts and work regarding the presented event-based non-rigid tracking methods are given. In section 7.1, a conclusion of the whole project is drawn. It includes the summary of our proposed approach and the experiments. We also present our analysis including contributions and limitations of our approach based on experiment results. Section 7.2 introduces several modern frameworks which are due to limited time and lack of prior knowledge have not been explored and implemented in the project.

7.1. Conclusion

7.1.1. Summary

Event-based cameras have received much attention in recent years due to their bio-inspired properties. Prior works show that event-based cameras have the great potential in the compute vision tasks with high-speed motion and high-dynamic-range scenarios [21, 40]. Recently, event-based cameras are deployed in the capture of non-rigid object, e.g. human bodies and hands. Nehvi, for example, designed a template-based hand tracking framework using event generation model [26]. Rudnev trained a fully-supervised neural network using synthetic event data to perform the hand tracking task [35]. However, these studies either have poor tracking performance or have not focused on the self-supervisory from event stream. In this study we proposed a event-based non-rigid tracking framework using the contour expectation maximization tracking algorithm and the contrast maximization framework, which can perform tracking under the self-supervisory from the event stream.

To generate the synthetic event data of non-rigid objects, we developed an event stream simulator (Cha. 4) extending from Nehvi's simulator [26] and ESIM [31]. Our simulator increases efficiency in the event generation process by the adaptive sampling and the parallel programming. We show the comparison with state-of-the-art event simulators [26, 31, 35] in section 4.6.

We demonstrated that events can be classified into texture events and contour events. Texture events do not suffer from self-occlusion (Fig. 5.8), they are feasible to be dealt with the contrast maximization algorithm [14]. For contour events, we proposed a novel

approach which maximizes the expectation of the data likelihood given the association between events and corresponding mesh faces (Sec. 5.3). In the full tracking framework (Sec. 5.5), we distinguish contour events and texture events, and process them with appropriate tracking framework.

We evaluate our proposed event-based tracking approach (Sec. 5.3) with experiments (Cha. 6). It shows that our EM-contour-tracker performs well in the tracking of the texture-less objects motion, because most events are generated by contour mesh faces. For texture-rich objects, we show that our full tracking framework (Sec. 5.5) can work in the 3-DoF rigid motion tracking, but is not feasible in the full 6-DoF motion tracking.

7.1.2. Analysis

Here, we analyse our approach based on experiment results appearing in Chapter 6. In our project, we proposed the contour expectation maximization tracking approach for contour events and proposed the contract maximization tracking approach for texture events. We show experiments and results in Chapter 6.

We used EM-contour-tracker to arm and hand motion sequences, since human arm and hand are texture-less and most events are then generated by the motion of boundary edges (Fig. 6.1). We show the results in section 6.2.1 that the proposed method is ideally capable for scenarios where most events are caused by the contour of objects. Our analysis is that for contour events, finding the corresponding mesh face and aligning them by maximizing the expectation of the measurement likelihood of association is an effective approach to reconstruct the non-rigid motion. Most notably, we compare our approach with state-of-the-art hand tracking and reconstruction works (Sec. 6.3) and demonstrate that our approach outperforms them in different metrics. Since there is no previous work addressing the event-based non-rigid tracking problem using contour events, this study therefore indicate that our proposed EM-contour-tracking approach can bring a contribution to the scientific community.

However, there is a limitation of our EM-contour-tracking approach. Assuming an non-contour event whose unprojection ray intersects with a mesh face, it already has high expectation over the measurement likelihood of the association. It means that our contour-EM-tracker is not feasible to reconstruct the motion from texture events. Thus, the contour-EM-tracking approach fails if texture events are wrongly classified as contour events.

We proposed a tracking method based on the contrast maximization algorithm (Sec. 3.6) to process texture events. Since motion of texture-rich objects generate both texture events and contour events, we use the dot product between the event bearing vectors and mesh face normals to formulate the contour probability and distinguish contour events and texture events. We show the rigid tracking experiment in section 6.2.2. The

result shows that our proposed combined tracking framework (Sec. 5.5) can perform the tracking of the 3-DoF planar rigid motion. Because there is no previous work using contrast maximization framework [14] to perform the object tracking, we think our approach could be a contribution to the scientific community.

However, the combined tracking framework still has limitations. We found that the contrast maximization framework has a bias towards the translation along the positive z-direction in the full 6-DoF rigid motion tracking. The reason for the bias is that the larger z value for objects corresponds to a smaller projection area and therefore a higher contrast of the final IWE (Fig. 5.9). We showed in figure 6.9 that adding EM-contour-tracking term to the contrast maximization term can avoid partly the above mentioned bias. However, the ground-truth value is still not at the local minimum of the loss landscape (Fig. 6.9) and the gradient is unclear. Thus the 6-DoF rigid motion tracking is a limitation of our proposed combined tracking framework. This limitation could be addressed in future works (Sec. 7.2).

Finally, we analyse the run-time of our EM-contour-tracking approach. For hand sequences, the tracking method takes averagely 8.76 seconds for 100 iteration steps for a spatio-temporal window (Sec. 5.2) of events. For human body sequences like arm and hand motion, the tracking method takes averagely 50.72 seconds for 100 iteration steps for each window. Note that the hand mesh model has 1538 mesh faces while the body mesh model has 20908 mesh faces. In our EM-contour-tracking approach, we calculate the data likelihood of an event and each mesh face of the model, which is computationally expensive. Besides, it is also redundant because an event can only be caused by several closest mesh faces. Thus, our approach has limitation to perform the real-time tracking.

7.2. Future Works

As explained in section 6.2.2, our proposed framework has bias towards translation in the z-direction in the 6-DoF motion tracking. Thus, one of the future work is the avoidance of the bias. The expectation maximization contour tracking term can partly avoid the bias (Fig. 6.9). Apparently, it needs more regulation on existing tracking framework to perform the full 6-DoF tracking.

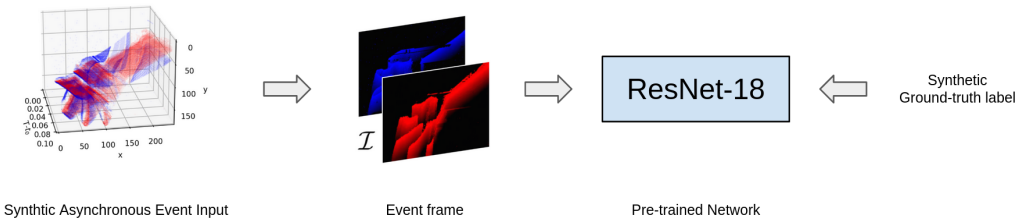
Another aspect to improve is the computational time. As an optimization-based method, our approach needs multiple steps until loss function converges. Although we parallelized the event processing by using PyTorch and CUDA, it still does not achieve the real-time performance. Therefore, one of the future work could be the optimizing of the code structure and implement it efficiently in `C++`. Besides, the data likelihood computation in the EM-contour-tracker is computationally expensive and redundant.

This can be addressed in future works by associating events only to several closest mesh faces to compute the data likelihood. In addition, we can extend our approach to a learning-based method. It will achieve the real-time performance, because the inference is extremely fast for trained a neural network.

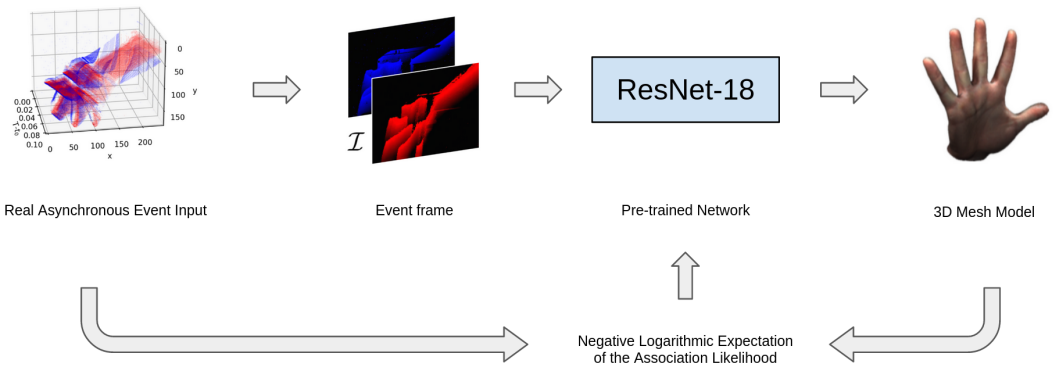
Recently, several learning-based works for event-based hand tracking and reconstruction are published. Essentially, it is difficult to obtain the ground-truth label for real data. Rudnev [35] proposed that it is possible to train a neural network on synthetic event data and infer the hand pose on real event stream. As an optimization-based approach, results in section 6.2.1 show that our objective function performs well in arm and hand tracking. Thus, we can extend our method to a self-supervised hand tracking framework, which can be trained with real event data without ground-truth labels. It can be divided into the following steps:

- Pre-training: we can pre-train the neural network using synthetic event data with ground-truth labels. The synthetic data can be used to initialize the weights of neural network, which could reduce the time of the training stage.
- Training: Given the real event data input, the pre-trained network can predict hand pose parameter θ. We generate the 3D hand model given the pose parameter θ. Then, we calculate the negative logarithmic expectation of the data likelihood given associations (Sec. 5.3), which can be used as the loss function of the neural network and should be minimized.
- Inference: Once the training with real event data is done, we can perform the inference with the trained network and predict the hand pose parameter directly from the real event data. The inference should achieve the real-time performance.

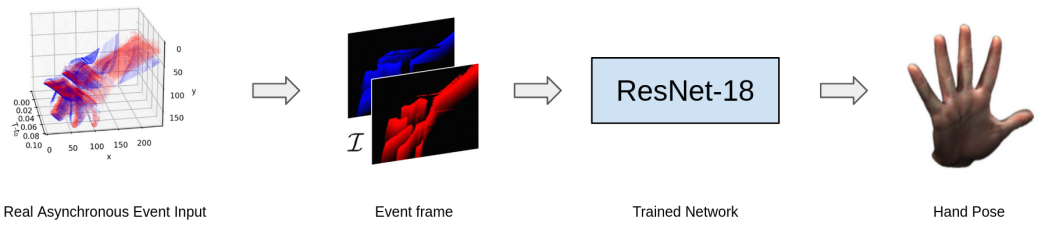
We also visualize each individual stage intuitively in figure 7.1.



(a) Initialize weights of neural network with synthetic event data.



(b) Train neural network using proposed self-supervisory with real event data.



(c) Infer hand pose from real event data using trained network.

Figure 7.1.: Pre-training(a), training(b), and inference(c) of the self-supervised method using our proposed objective function. Event stream and Event frame are from [35].

A. Hyperparameter

A.1. Hyperparameter Tuning

As introduced in section 3.3, we use optuna [1] to tune all hyperparameters in our framework. Essentially, we have following hyperparameters in our approach:

- α: sharpness control variable for lateral distance (Eq. 5.5)
- β: sharpness control variable for longitudinal distance (Eq. 5.6)
- γ: sharpness control variable for normal dot product (Eq. 5.1)
- k_1: weights of expectation maximization term (Eq. 5.22)
- k_2: weights of contrast maximization term (Eq. 5.22)
- k_3: weights of constant velocity term for hands (Eq. 5.22)
- c: Huber scale (Eq. 5.5)
- lr: step size of Adam optimizer
- $\theta_{lateral\_distance}$: lateral distance threshold for inlier events
- $\theta_{expectation\_update}$: gradient threshold for updating hidden variable distribution of EM-method
- θ_{early_stop}: gradient threshold for early stopping after the convergence of the optimization
- outlier_likelihood: likelihood to downweight outlier events. Appears on the denominator of the hidden variable distribution term (Eq. 5.13)

For each scenario, we have 10 training sequences to tune the hyperparameters. We use the MPJPE (Sec. 6.1.4) as the metric of the loss function. Optuna will find the smallest MPJPE error for the hyperparameters.

A.2. Hyperparameter Applied

We show the hyperparameters we used in the experiments (Cha. 6). Depending on the scenarios, we have different settings of hyperparameters for the motion reconstruction based on the MANO model and the SMPL-X model. The loss of the hyperparameter tuning is visualized in figure A.1.

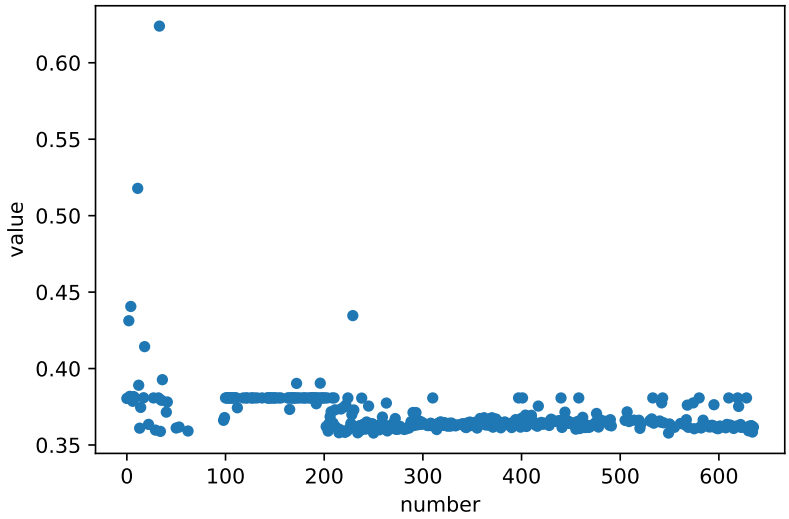
MANO

- $\alpha = 3.104415774078913 \times 10^{-6}$
- $\beta = 0.2766735382305266$
- $\gamma = 11.398513831616794$
- $k_1 = 0.000101846010110427$
- $k_2 = 0.0$, only EM-tracker (Sec. 5.3) deployed
- $k_3 = 7.030594621430028 \times 10^{-5}$
- c $= 0.005964889664885093$
- lr $= 0.00065505427907943$
- $\theta_{lateral\_distance} = 0.01124935159840215$
- $\theta_{expectation\_update} = 0.23403676564874817$
- $\theta_{early\_stop} = 2.6222626765129198 \times 10^{-8}$
- outlier_likelihood $= 5.099061715891875 \times 10^{-10}$

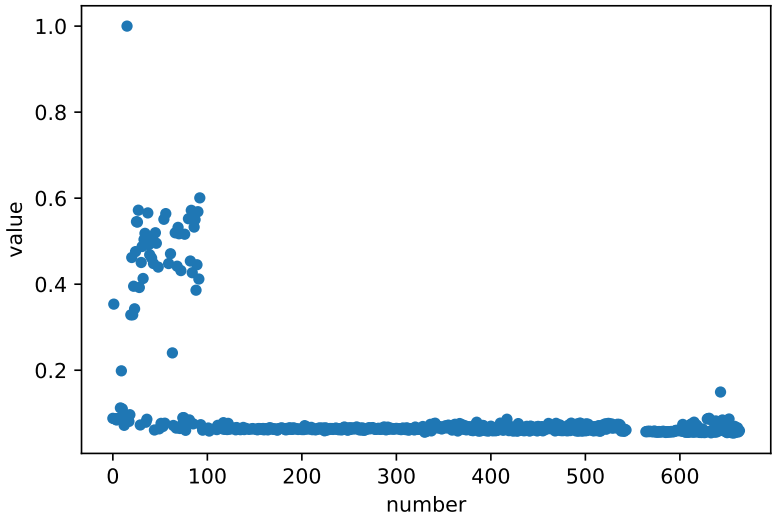
SMPL-X

- $\alpha = 1.0304767369667457 \times 10^{-7}$
- $\beta = 0.10679963391728674$
- $\gamma = 0.10986587419562713$
- $k_1 = 1.0$
- $k_2 = 0.0$, only EM-tracker (Sec. 5.3) deployed
- $k_3 = 0.00789683086$

- c = 1.0
- lr = 0.003150912997278243
- $\theta_{lateral\_distance} = 0.0022776835235587$
- $\theta_{expectation\_update} = 13.811671131591776$
- $\theta_{early\_stop} = 1.2568652735513764 \times 10^{-8}$
- outlier_likelihood $= 1.0 \times 10^{-10}$



(a) Optuna loss for MANO scenario



(b) Optuna loss for SMPL-X scenario.

Figure A.1.: Optuna loss of the hyperparamters tuning for different scenarios.

B. Sampled Data

We visualize here the some randomized pose of different objects.

B.1. MANO

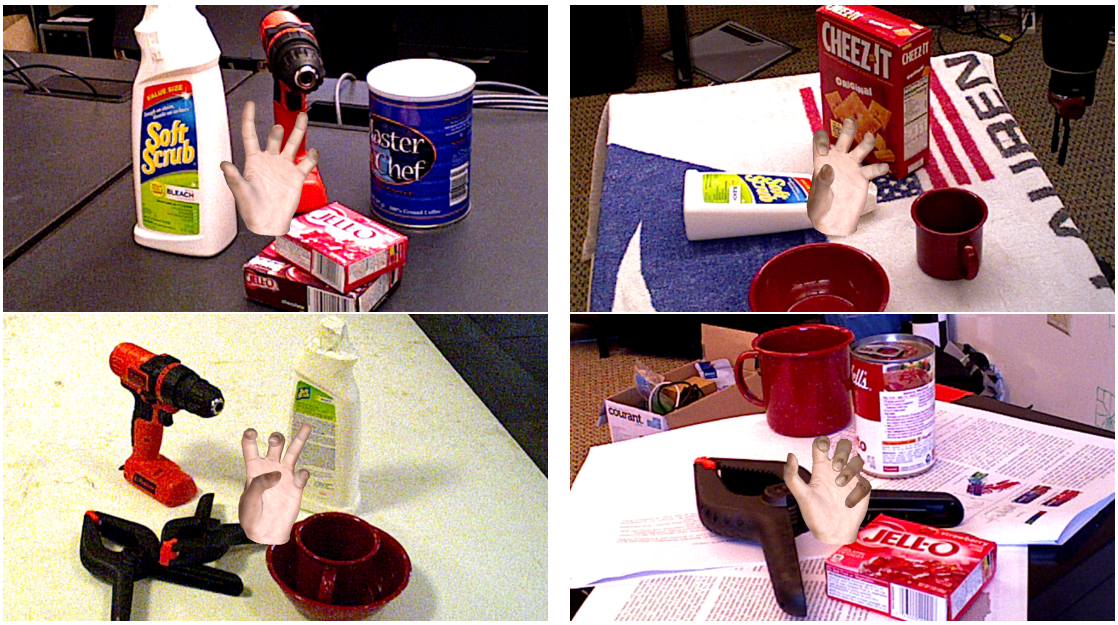


Figure B.1.: Some sampled poses in synthetic MANO hand sequences.

B.2. SMPL-X Hand

B.3. SMPL-X Arm and Hand

Figure B.2.: Some sampled poses in synthetic SMPL-X hand sequences.

Figure B.3.: Some sampled poses in synthetic SMPL-X arm and hand sequences.

List of Figures

List of Tables

Bibliography

[1] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. "Optuna: A Next-generation Hyperparameter Optimization Framework." In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*. Ed. by A. Teredesai, V. Kumar, Y. Li, R. Rosales, E. Terzi, and G. Karypis. ACM, 2019, pp. 2623–2631. DOI: 10.1145/3292500.3330701.

[2] P. Bardow, A. J. Davison, and S. Leutenegger. "Simultaneous Optical Flow and Intensity Estimation from an Event Camera." In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pp. 884–892. DOI: 10.1109/CVPR.2016.102.

[3] J. T. Barron. "A General and Adaptive Robust Loss Function." In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, 2019, pp. 4331–4339. DOI: 10.1109/CVPR.2019.00446.

[4] A. Bartoli, Y. Gérard, F. Chadebecq, T. Collins, and D. Pizarro. "Shape-from-Template." In: *IEEE Trans. Pattern Anal. Mach. Intell.* 37.10 (2015), pp. 2099–2118. DOI: 10.1109/TPAMI.2015.2392759.

[5] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. "Algorithms for Hyper-Parameter Optimization." In: *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain*. Ed. by J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. C. N. Pereira, and K. Q. Weinberger. 2011, pp. 2546–2554.

[6] C. M. Bishop. *Pattern recognition and machine learning, 5th Edition*. Information science and statistics. Springer, 2007. ISBN: 9780387310732.

[7] A. Bozic, P. R. Palafox, M. Zollhöfer, A. Dai, J. Thies, and M. Nießner. "Neural Non-Rigid Tracking." In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin. 2020.

[8] S. Bryner, G. Gallego, H. Rebecq, and D. Scaramuzza. "Event-based, Direct Camera Tracking from a Photometric 3D Map using Nonlinear Optimization." In: *International Conference on Robotics and Automation, ICRA 2019, Montreal, QC, Canada, May 20-24, 2019*. IEEE, 2019, pp. 325–331. DOI: 10.1109/ICRA.2019.8794255.

[9] B. Çalli, A. Singh, J. Bruce, A. Walsman, K. Konolige, S. S. Srinivasa, P. Abbeel, and A. M. Dollar. "Yale-CMU-Berkeley dataset for robotic manipulation research." In: *Int. J. Robotics Res.* 36.3 (2017), pp. 261–268. DOI: 10.1177/0278364917700714.

[10] P. Davies. *Kendall's Advanced Theory of Statistics. Volume 1. Distribution Theory*. 1988.

[11] Y. Feng, V. Choutas, T. Bolkart, D. Tzionas, and M. J. Black. "Collaborative Regression of Expressive Bodies using Moderation." In: *International Conference on 3D Vision (3DV)*. 2021.

[12] G. Gallego, T. Delbrück, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. J. Davison, J. Conradt, K. Daniilidis, and D. Scaramuzza. "Event-Based Vision: A Survey." In: *IEEE Trans. Pattern Anal. Mach. Intell.* 44.1 (2022), pp. 154–180. DOI: 10.1109/TPAMI.2020.3008413.

[13] G. Gallego, M. Gehrig, and D. Scaramuzza. "Focus Is All You Need: Loss Functions for Event-Based Vision." In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, 2019, pp. 12280–12289. DOI: 10.1109/CVPR.2019.01256.

[14] G. Gallego, H. Rebecq, and D. Scaramuzza. "A Unifying Contrast Maximization Framework for Event Cameras, With Applications to Motion, Depth, and Optical Flow Estimation." In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. Computer Vision Foundation / IEEE Computer Society, 2018, pp. 3867–3876. DOI: 10.1109/CVPR.2018.00407.

[15] M. Gehrig, M. Millhäusler, D. Gehrig, and D. Scaramuzza. "E-RAFT: Dense Optical Flow from Event Cameras." In: *International Conference on 3D Vision, 3DV 2021, London, United Kingdom, December 1-3, 2021*. IEEE, 2021, pp. 197–206. DOI: 10.1109/3DV53792.2021.00030.

[16] J. A. Holdener. "Review: Roots to Research: A Vertical Development of Mathematical Problems by Judith D. Sally; Paul J. Sally," in: *Am. Math. Mon.* 116.8 (2009), pp. 754–758.

[17] H. Kim, S. Leutenegger, and A. J. Davison. "Real-Time 3D Reconstruction and 6-DoF Tracking with an Event Camera." In: *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VI*. Ed. by B. Leibe, J. Matas, N. Sebe, and M. Welling. Vol. 9910. Lecture Notes in Computer Science. Springer, 2016, pp. 349–364. DOI: 10.1007/978-3-319-46466-4_21.

[18] D. P. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization." In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Y. Bengio and Y. LeCun. 2015.

[19] S. K. Lam, A. Pitrou, and S. Seibert. "Numba: a LLVM-based Python JIT compiler." In: *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC, LLVM 2015, Austin, Texas, USA, November 15, 2015*. Ed. by H. Finkel. ACM, 2015, 7:1–7:6. DOI: 10.1145/2833157.2833162.

[20] J. Lamarca, S. Parashar, A. Bartoli, and J. M. M. Montiel. "DefSLAM: Tracking and Mapping of Deforming Scenes From Monocular Sequences." In: *IEEE Trans. Robotics* 37.1 (2021), pp. 291–303. DOI: 10.1109/TRO.2020.3020739.

[21] H. Li and J. Stueckler. "Tracking 6-DoF Object Motion from Events and Frames." In: *IEEE International Conference on Robotics and Automation, ICRA 2021, Xi'an, China, May 30 - June 5, 2021*. IEEE, 2021, pp. 14171–14177. DOI: 10.1109/ICRA48506.2021.9561760.

[22] T. Li, T. Bolkart, M. J. Black, H. Li, and J. Romero. "Learning a model of facial shape and expression from 4D scans." In: *ACM Trans. Graph.* 36.6 (2017), 194:1–194:17. DOI: 10.1145/3130800.3130813.

[23] K. Lin, L. Wang, and Z. Liu. "Mesh Graphormer." In: *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*. IEEE, 2021, pp. 12919–12928. DOI: 10.1109/ICCV48922.2021.01270.

[24] S. Liu, W. Chen, T. Li, and H. Li. "Soft Rasterizer: A Differentiable Renderer for Image-Based 3D Reasoning." In: *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. IEEE, 2019, pp. 7707–7716. DOI: 10.1109/ICCV.2019.00780.

[25] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black. "SMPL: a skinned multi-person linear model." In: *ACM Trans. Graph.* 34.6 (2015), 248:1–248:16. DOI: 10.1145/2816795.2818013.

[26] J. Nehvi, V. Golyanik, F. Mueller, H. Seidel, M. Elgharib, and C. Theobalt. "Differentiable Event Stream Simulator for Non-Rigid 3D Tracking." In: *IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2021, virtual, June 19-25, 2021*. Computer Vision Foundation / IEEE, 2021, pp. 1302–1311. DOI: 10.1109/CVPRW53098.2021.00143.

[27] J. Östlund, A. Varol, D. T. Ngo, and P. Fua. "Laplacian Meshes for Monocular 3D Shape Recovery." In: *Computer Vision - ECCV 2012 - 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part III*. Ed. by A. W. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid. Vol. 7574. Lecture Notes in Computer Science. Springer, 2012, pp. 412–425. DOI: 10.1007/978-3-642-33712-3_30.

[28] S. Parashar, D. Pizarro, and A. Bartoli. "Isometric Non-Rigid Shape-from-Motion with Riemannian Geometry Solved in Linear Time." In: *IEEE Trans. Pattern Anal. Mach. Intell.* 40.10 (2018), pp. 2442–2454. DOI: 10.1109/TPAMI.2017.2760301.

[29] G. Pavlakos, V. Choutas, N. Ghorbani, T. Bolkart, A. A. A. Osman, D. Tzionas, and M. J. Black. "Expressive Body Capture: 3D Hands, Face, and Body From a Single Image." In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, 2019, pp. 10975–10985. DOI: 10.1109/CVPR.2019.01123.

[30] N. Ravi, J. Reizenstein, D. Novotný, T. Gordon, W. Lo, J. Johnson, and G. Gkioxari. "Accelerating 3D Deep Learning with PyTorch3D." In: *CoRR* abs/2007.08501 (2020). arXiv: 2007.08501.

[31] H. Rebecq, D. Gehrig, and D. Scaramuzza. "ESIM: an Open Event Camera Simulator." In: *2nd Annual Conference on Robot Learning, CoRL 2018, Zürich, Switzerland, 29-31 October 2018, Proceedings*. Vol. 87. Proceedings of Machine Learning Research. PMLR, 2018, pp. 969–982.

[32] H. Rebecq, T. Horstschaefer, and D. Scaramuzza. "Real-time Visual-Inertial Odometry for Event Cameras using Keyframe-based Nonlinear Optimization." In: *British Machine Vision Conference 2017, BMVC 2017, London, UK, September 4-7, 2017*. BMVA Press, 2017.

[33] H. Rebecq, R. Ranftl, V. Koltun, and D. Scaramuzza. "High Speed and High Dynamic Range Video with an Event Camera." In: *IEEE Trans. Pattern Anal. Mach. Intell.* 43.6 (2021), pp. 1964–1980. DOI: 10.1109/TPAMI.2019.2963386.

[34] J. Romero, D. Tzionas, and M. J. Black. "Embodied hands: modeling and capturing hands and bodies together." In: *ACM Trans. Graph.* 36.6 (2017), 245:1–245:17. DOI: 10.1145/3130800.3130883.

[35] V. Rudnev, V. Golyanik, J. Wang, H. Seidel, F. Mueller, M. Elgharib, and C. Theobalt. "EventHands: Real-Time Neural 3D Hand Reconstruction from an Event Stream." In: *CoRR* abs/2012.06475 (2020). arXiv: 2012.06475.

[36] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach (4th Edition)*. Pearson, 2020. ISBN: 9780134610993.

[37] V. Sidhu, E. Tretschk, V. Golyanik, A. Agudo, and C. Theobalt. "Neural Dense Non-Rigid Structure from Motion with Latent Space Constraints." In: *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XVI*. Ed. by A. Vedaldi, H. Bischof, T. Brox, and J. Frahm. Vol. 12361. Lecture Notes in Computer Science. Springer, 2020, pp. 204–222. DOI: 10.1007/978-3-030-58517-4_13.

[38] T. Stoffregen and L. Kleeman. "Event Cameras, Contrast Maximization and Reward Functions: An Analysis." In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, 2019, pp. 12300–12308. DOI: 10.1109/CVPR.2019.01258.

[39] R. Szeliski. *Computer Vision - Algorithms and Applications*. Texts in Computer Science. Springer, 2011. ISBN: 978-1-84882-934-3. DOI: 10.1007/978-1-84882-935-0.

[40] A. R. Vidal, H. Rebecq, T. Horstschaefer, and D. Scaramuzza. "Ultimate SLAM? Combining Events, Images, and IMU for Robust Visual SLAM in HDR and High-Speed Scenarios." In: *IEEE Robotics Autom. Lett.* 3.2 (2018), pp. 994–1001. DOI: 10.1109/LRA.2018.2793357.

[41] A. Watt and M. Watt. *Advanced animation and rendering techniques - theory and practice*. Addison-Wesley, 1992. ISBN: 978-0-201-54412-1.

[42] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. "PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes." In: *Robotics: Science and Systems XIV, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, June 26-30, 2018*. Ed. by H. Kress-Gazit, S. S. Srinivasa, T. Howard, and N. Atanasov. 2018. DOI: 10.15607/RSS.2018.XIV.019.

[43] A. Z. Zhu, L. Yuan, K. Chaney, and K. Daniilidis. "Unsupervised Event-Based Optical Flow Using Motion Compensation." In: *Computer Vision - ECCV 2018 Workshops - Munich, Germany, September 8-14, 2018, Proceedings, Part VI*. Ed. by L. Leal-Taixé and S. Roth. Vol. 11134. Lecture Notes in Computer Science. Springer, 2018, pp. 711–714. DOI: 10.1007/978-3-030-11024-6_54.