UPPSALA
UNIVERSITET

UPTEC F 23060

Degree project 30 credits

sept 2023

Event-Based Visual SLAM

An Explorative Approach

Johan Rideg

Master’s Programme in Engineering Physics

UPPSALA UNIVERSITET

Event-Based Visual SLAM: An Explorative Approach

Johan Rideg

Abstract

Simultaneous Localization And Mapping (SLAM) is an important topic within the field of robotics aiming to localize an agent in a unknown or partially known environment while simultaneously mapping the environment. The ability to perform robust SLAM is especially important in hazardous environments such as natural disasters, firefighting and space exploration where human exploration may be too dangerous or impractical. In recent years, neuromorphic cameras have been made commercially available. This new type of sensor does not output conventional frames but instead an asynchronous signal of events at a microsecond resolution and is capable of capturing details in complex lightning scenarios where a standard camera would be either under- or overexposed, making neuromorphic cameras a promising solution in situations where standard cameras struggle. This thesis explores a set of different approaches to virtual frames, a frame-based representation of events, in the context of SLAM. UltimateSLAM, a project fusing events, gray scale and IMU data, is investigated using virtual frames of fixed and varying frame rate both with and without motion compensation. The resulting trajectories are compared to the trajectories produced when using gray scale frames and the number of detected and tracked features are compared. We also use a traditional visual SLAM project, ORB-SLAM, to investigate the Gaussian weighted virtual frames and gray scale frames reconstructed from the event stream using a recurrent network model. While virtual frames can be used for SLAM, the event camera is not a plug and play sensor and requires a good choice of parameters when constructing virtual frames, relying on pre-existing knowledge of the scene.

Faculty of Science and Technology

Uppsala University, Place of publication Uppsala

Supervisor: Oskar Karlsson Subject reader: Ayça Özçelikkale

Examiner: Tomas Nyberg

Populärvetenskaplig Sammanfattning

SLAM (Simultaneous Localization And Mapping) är ett centralt ämne inom robotik som ämnar att lokalisera en agent i en okänd eller delvis okänd miljö samtidigt som denna kartläggs. Förmågan att kunna utföra SLAM är av särskild vikt i farliga miljöer exempelvis vid naturkatastrofer, brandbekämpning och utomjordiska aktiviteter där mänsklig utforskning kan vara för farligt eller opraktiskt. De senaste åren har eventkameror blivit kommersiellt tillgängliga. Denna nya typ av sensor har inte bilder som signal utan istället en asynkron ström av events i mikrosekundsupplösning och kan fånga detaljer trots svåra ljusförhållanden där traditionella kameror skulle vara över- eller underexponerade. Detta gör eventkameran till en lovande kandidat där den traditionella kameran presterar dåligt. Detta examensarbete utforskar olika metoder för att skapa virtuella bilder för SLAM. UltimateSLAM, ett projekt som förenar events, gråskalig bilddata och IMU-data, undersöks där virtuella bilder med fixerad och varierande bildfrekvens samt med och utan rörelsekompensering används. De estimerade banorna jämförs med motsvarande banor när gråskalig bilddata har använts och antalet upptäckta och målföljda landmärken jämförs. Vi använder också ORB-SLAM, ett projekt för traditionell visuell bilddata, för att undersöka virtuella bilder med en Gaussisk viktfunktion och gråskaliga bilder som estimerats av en RNN-modell. Trots att virtuella bilder kan användas för SLAM så är eventkameran inte plug-and-play och kräver väl valda parametrar för att skapa bra virtuella bilder som lutar sig på kunskap om miljön i förtid.

Contents

1 Introduction

1.1 Event cameras

Recently, Dynamic Vision Sensors (DVS), commonly known as event cameras, have been made commercially available. An event camera is composed by an array of event pixels which operate independently from each other. An event pixel fires an event when the intensity change exceeds a threshold and contains information about when and where the event occurred in the form of a timestamp, its pixel position and a polarity, a binary entity indicating whether the intensity increased or decreased. Thus the output signal is not a constant frequency sequence of images as is the case with a standard camera but an asynchronous, spiking signal of individual events. The challenge to accurately represent visual event data in the context of visual odometry (VO) is something that recently have begun to gain some attention [1][2][3][4].

Event cameras typically operate at a high dynamic range (HDR) around 140 dB compared to the dynamic range of standard cameras at around 60 dB [5]. This enables event cameras to capture a wider range of lighting scenarios where a standard camera would be either over- or under exposed in different regions of the scene. The temporal resolution of an event camera is in the order of microseconds, making it possible to capture high speed scenarios such as bullet impacts and complex maneuvers of drones with no motion blur [6]. Because of its neuromorphic design, only the excited event pixels are drawing power, leading to a low power requirement. As a reference, the DAVIS346 has a power requirement in the range of 10 to 170 mW [5]. This is however very dependent of the scene and motion. Ideally, if the camera and scene remain static, no events will fire and the power consumption becomes very low. Because of these reasons, event cameras seem to be a promising sensor for robotic applications where computational hardware and energy storage are often limited. In the context of VO, it may fill some of the gaps where traditional visual sensors struggle, particularly during high speed motion and difficult illumination conditions.

1.2 Feature Detection and Tracking

The field of image feature detection includes a wide range of features or interest points. Ideally, an image feature is a distinguishing element within an image and is invariant to changes in scale, rotation and illumination so that features detected in one image can be reacquired another. This is used in image registration, facial recognition and 3D reconstruction to name some applications. In general, features can be any distinguishing element such as lines and points or more semantic features such as furniture and handwritten text. A commonly used feature in computer vision are corners. Corners are easy to model and are relatively computationally cheap to detect, compared to many other choices of features. Some commonly used corner detector algorithms are:

- FAST
- ORB
- SIFT
- SURF
- Harris Corner

FAST (Features from Accelerated Segment Test) is a method for corner detection primarily characterized by its speed, making it suitable for real-time applications, for example, VO. A basic implementation of FAST is for a pixel coordinate p with and intensity I_p, choose a threshold t and check the intensity of the 16 test pixels S forming a rasterized circle around p. If a set of N contiguous pixels in S have an intensity higher than $I_p + t$ or lower than $I_p - t$, p is considered as a corner. Non-maximum suppression is commonly used to score features within a local region, discarding lower scoring features and retaining only the best performing features. The scoring strategy itself is dependent on the choice of feature. In the case of FAST corners, the cost function could be the sum of absolute differences of the intensities of p and test pixels.

Features can be complemented with a descriptor acting as a ”fingerprint” of a local neighbourhood around the feature. More formally, a descriptor is a vector in a high-dimensional space which ideally is unique for the local image space around a feature. If the descriptors are robust under changes in rotation, scale and illuminance, the descriptor vector should remain in a local neighbourhood within the descriptor space. By comparing descriptors from features found in other images, features can be matched and subsequently tracked. This can also be an important step in determining whether two images are taken from the same perspective if a similar set of descriptors are discovered. Feature tracking can also be done using methods for calculating optical flow. Tomasi built upon the Lucas-Kanade method for calculating dense optical flow (every pixel) by instead restricting the optical flow problem to a sparse set of good features to track. This method is known as the KLT-feature tracker and is a popular method for tracking features between consecutive frames due to its fast performance and is commonly used for feature tracking.

1.3 Visual SLAM

The ability for an agent to locate itself within an unknown environment enables many useful technologies such as autonomous driving, augmented reality and exploration of hazardous environments. Sometimes, a reliable external positioning system is not available such as in buildings and during extraterrestrial exploration. To overcome these scenarios, the agent has to rely on on-board sensors to perform odometry and map the environment. This is commonly referred to as Simultaneous Localization And Mapping (SLAM) which is a central problem in robotics. SLAM has been solved using several sensor configurations involving both active sensors such as light detection and ranging (LiDAR), sonic navigation and ranging (sonar) and RGB-D cameras and passive sensors such as inertial measurement units (IMU) and traditional RGB- and gray scale cameras. Visual SLAM (VSLAM) and VO are generally categorized as direct and indirect. Direct methods rely directly on pixel intensity values and compares a sequence of frames by a photometric error. This approach enables the use of all pixels within the frame but may become expensive to compute. Indirect visual VSLAM and VO methods operate by having a preprocessing stage or front-end where points of interest are identified. Commonly, these are features that are simple to describe mathematically such as corners or lines but other more semantic approaches exists [7]. Regardless of the choice of features, a sparse representation of the visual data is used to estimate the motion and the location of features. Information about the features, such as position and vector representations of features can be stored and used to facilitate the tracking of these features. Since a monocular setup can only measure a 2D representation of the scene in each individual frame, the third dimension is provided by triangulating the tracked features to relate their relative depth to the sensor and the positional change of the sensor frame to frame. In indirect methods, this often culminates in a optimization problem of minimizing the error between predicted feature point coordinates, given a camera model, and the observed pixel points. This error is called reprojection error and the process of minimizing the reprojection error with respect to a change in pose enables a VSLAM/VO system to track a trajectory over consecutive frames. Each individual estimate is imperfect and introduces some level of uncertainty to the estimated global trajectory. Without a strategy to counter the buildup of errors, the estimated trajectory will drift from the ground truth. One strategy to contain the accumulation of errors is by performing loop closure. The principle behind this is detecting a previously visited location, joining the trajectory estimates at that location and adjusting the intermediary trajectory points accordingly. The design of this back-end stage of the algorithm varies between implementations.

1.3.1 Brief History of Visual SLAM

In the early 2000s, Visual SLAM started gaining real attention. Davidson et. al. presented MonoSLAM [8] in 2007. MonoSLAM formulates the SLAM problem as an Extended Kalman Filter (EKF) framework where the camera trajectory and map are represented as the state vector, and the EKF is used to iteratively update the state estimates based on incoming frames. An issue with this and similar approaches is that as the trajectory and feature map grows, so does the computational requirement. Klein and Murray proposed PTAM [9] in 2007, which set a new standard in SLAM/VO by parallelizing the mapping and tracking and moving away from Bayesian filtering, adopting keyframe-based bundle adjustment instead. The performance of PTAM was limited in the size of the environment it was operating within. As the fields of robotics and computer vision matured, solutions for large scale SLAM became feasible. Engel et. al. published LSD-SLAM in 2015, a direct SLAM system aligning a semi-dense set of pixels of highest intensity gradient and is equipped with loop closure. As tools within the fields of computer vision and optimization became more easily implementable, larger and more complex systems started appearing.

1.3.2 ORB-SLAM

One particularly prominent VSLAM system is ORB-SLAM, utilizing ORB-features which are designed to be robust under orientation and scale changes while being computationally efficient to compute. ORB-SLAM utilizes a binary bag of words that is used to categorize the feature descriptors into a smaller subset which serves to increase the robustness of feature matching and in extension, location recognition. ORB-SLAM is equipped with procedures for both local and global (loop closure) trajectory refinement by employing bundle adjustment, an optimization technique that simultaneously refines a set of estimated trajectory points and observed features in 3D-space. The first version of ORB-SLAM operated using a monocular camera setup but has been further developed, both by the initial authors and other outside parties, to include other setups.

1.3.3 UltimateSLAM

Researchers at the University of Zürich recognized the potential of event cameras and published the paper behind UltimateSLAM [1], an indirect SLAM system fusing gray scale frames, inertial measurements and virtual frames sourced from event cameras, which are discussed in more detail in Section 2.2. The system is equipped with a motion compensation method which relies on an IMU to mitigate accumulation blur that may arise during complex motion. In principle, this aims to project events onto a virtual image plane perpendicular to the event trajectories. FAST corners are used on both gray scale and virtual frames and tracked through KLT tracking. The system does not employ an explicit loop closure procedure but does refine the local trajectory periodically.

1.4 Research Questions

Based on the benefits the event camera may offer and the progress made in the field of VSLAM, it is interesting to investigate the practical usage of event cameras in a SLAM context given methods readily available today.

1. What gains in accuracy can we expect in scenarios with difficult lightning conditions and rapid motion when using event cameras in place of standard cameras?
2. Can existing VSLAM methods, such as ORB-SLAM, be used to perform SLAM using images created artificially from the asynchronous signal of an event camera?

2 Method

2.1 Calibrating an Event Camera

This section describes one method to calibrate an event camera. Some event cameras, such as DAVIS346, are equipped with a gray scale sensor array while others, such as DVXplorer, are not. In order to utilize frame based calibration tools, such as Kalibr [10], we need a method of sourcing frames from events. This was done using e2calib [11], an open source project, which utilize a pre-trained neural network model called E2VID [6] for events-to-image reconstruction and is compatible with The Robot Operating System (ROS), an open source framework for robotic applications. ROS was used to record all data into rosbags, a file format used for logging ROS messages.

We recorded two sequences which lasted about one minute each. The first sequence, **cam_calib.bag**, contains only events which was collected by recording a grid of Apriltags, a less information dense variant of QR codes, while being careful to cover the entire image with the target and doing slow and controlled motions. The second sequence, **imu_cam_calib.bag**, recorded both events and IMU readings and used more rapid motion than the first sequence in order to properly excite the IMU. In theory, one sequence may be enough for both intrinsic and extrinsic calibration but recording one for finding the optical parameters and one to find the extrinsic relationship between the image sensor and the integrated IMU is generally a good idea. This is because a good optical parameter calibration require the target to cover the entire sensor without loosing track while a good IMU-to-camera calibration requires faster motion to properly excite the IMU. The event streams in both datasets needed to be reconstructed to their corresponding image sequences.

Using Kalibr, the intrinsic calibration of the camera could then be performed on the reconstructed image sequence within **cam_recon.bag**. The same events-to-image procedure was performed on the second sequence containing events and IMU readings, merging the reconstructed image sequence with the corresponding IMU readings in **imu_cam_merged.bag**. A third and final recording, **imu_noise.bag**, consisted of five hours of IMU noise data recorded at 800 Hz. This was used to model noise density and random walk of accelerometer and gyroscope through Allan variance analysis. Finally we could model the transformation between camera and IMU, optical properties and IMU noise and biases. The calibration workflow, illustrated in figure 1, results in a calibration file containing calibrated optical parameters, camera-IMU transformations and IMU drift and noise parameters.

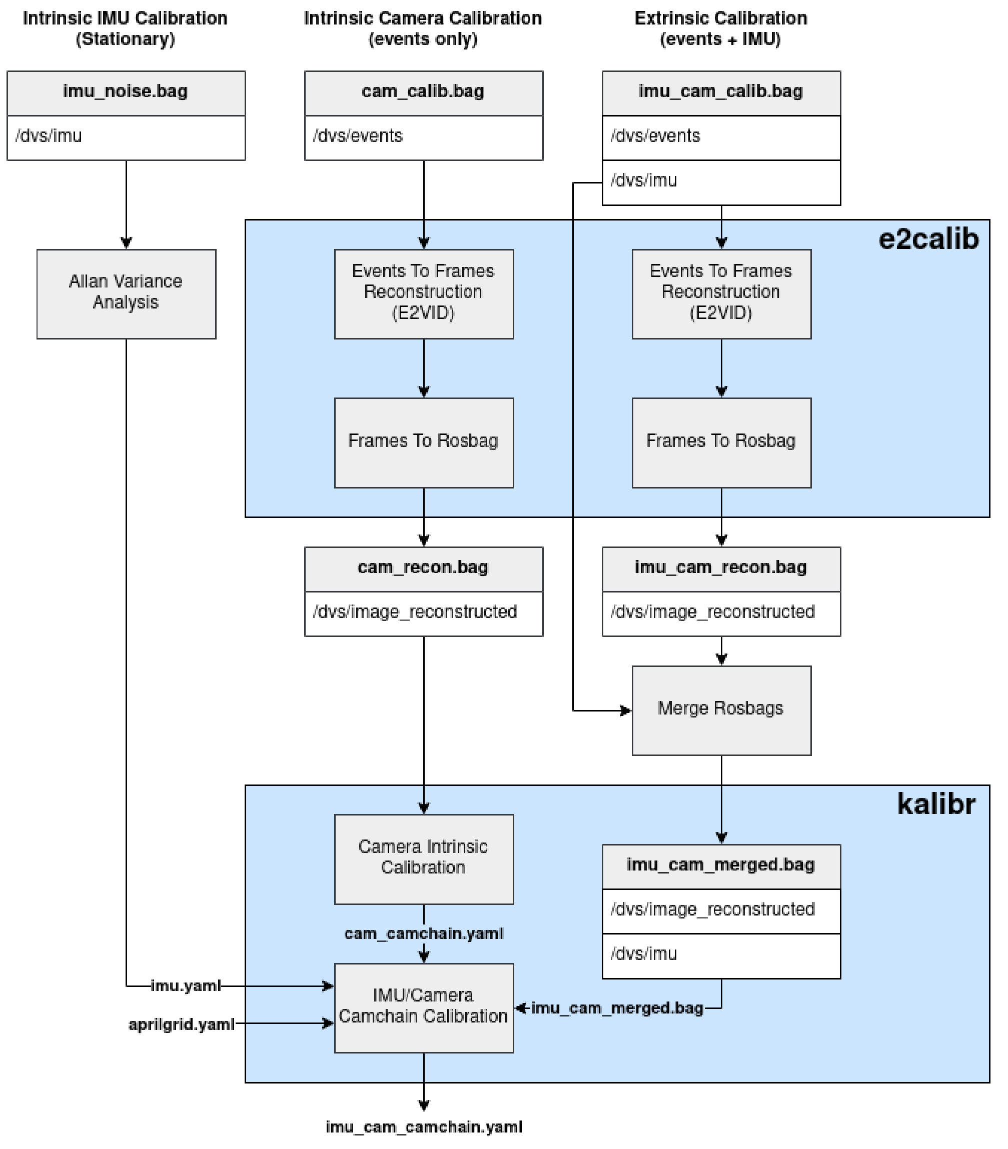

Figure 1: Flowchart describing the calibration procedure using events streams and an IMU-readings.

2.2 Virtual Frames

With standard cameras, a sequence of images is readily available and we may apply techniques withing the field of computer vision immediately. With event cameras, we need a scheme to to construct the corresponding sequence of images. There are multiple approaches to this problem. Using a fixed number of events per frame should hypothetically, given a homogeneously textured environment and low-noise conditions, self-compensate for blur during translatory motion. This would be performed by accumulating a set number of events per frame (EPF) events into each virtual frame. This will mitigate translatory blur artifacts but introduces problems of its own. The frequency of virtual frames will be varying with the event rate, producing high volumes of virtual frames during fast motions and low volumes when relative scene is close to static illustrated in figure 2. This is unsuitable for a VSLAM implementation for several reasons. One reason is that most SLAM implementations rely on fixed frequency sensors. Another important reason to this is that even if frames can be produced at a high frame rate during high speed motion, the requirement posed onto the computational hardware may render the setup infeasible.

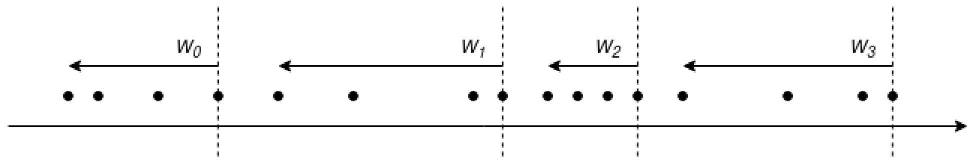

Figure 2: Event accumulation using a fixed number of events. The accumulation windows W_i differ in both length are generated at a varying frame rate. EPF = 4

A second method would be to split the event stream into regular spatio-temporal slices and project these onto the image plane. Doing this would yield a sequence of virtual frames at a fixed frequency. However, if the motion, or the scene's texture level is varying, some virtual frames would contain a significantly higher number of events and appear either blurred or not contain enough detail to perform any meaningful analysis. One approach to mitigate this is by combining the first and second approaches, projecting a fixed number of events onto the image plane but doing so in regular intervals, illustrated in Figure 3.

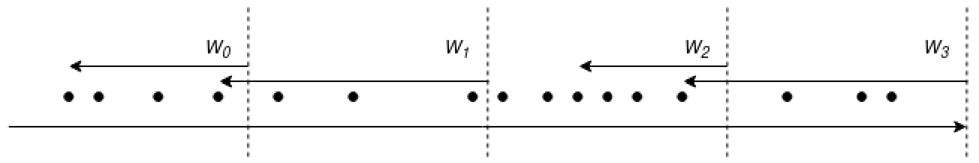

Figure 3: Event accumulation using both fixed frame rate and a fixed EPF. The accumulation windows W_i are evenly distributed. Overlap of event may occur when the event rate is low and events may be skipped when event rate is high. EPF = 4

This approach is still not without its issues in the sense that it still assumes a relatively similar level of texture of the scene throughout the motion. A low EPF may be suitable for simple environments, such as indoors while not offering enough detail in outside environments where the texture level is generally larger. Similarly, a high EPF may be suitable for scenes with a high level of texture but will introduce unnecessary blur should the camera transition into a simpler environment.

2.3 UltimateSLAM

2.3.1 Dataset

The dataset used for UltimateSLAM was comprised of six sequences from The Event Camera Dataset [12] containing one minute of gray scale images, events and IMU-readings recorded using a DAVIS240. All sequences are divided into two sections. The first part of a sequence uses slow and controlled motion while the second part uses rapid, shake-like motion. The sequences capture, with varying levels of texture, both normal and HDR environments.

Sequence	Description
boxes_6dof	A collection of boxes with varying level of texture.
dynamic_6dof	Office environment with a person walking around.
hdr_boxes	6DOF motion with limited illumination.
hdr_poster	6DOF motion with limited illumination.
poster_6dof	A poster with varying level of texture.
shapes_6dof	Low level of texture scene with simple shapes.

Table 1: Short descriptions of sequences used from The Event Camera Dataset.

2.3.2 Experiments with Different Virtual Frame Schemes

Five different cases are explored with a base case being only gray scale frames. The four virtual frame schemes that are used are constant frame rate with motion compensation (CC), constant frame rate without motion compensation (CNC) where the frame rate is the same as the frame rate of the corresponding gray scale sequence of 24 Hz. The other two cases are varying frame rate with motion compensation (VC) and varying frame rate without motion compensation (VNC), producing virtual frames when enough events are collected, illustrated in Figure 2. In all experiments, a FAST threshold of 50 was used for corner acquisition. The KLT tracker was configured using 2 pyramid levels and a patch size of 24x24 pixels. For virtual frame accumulation, boxes_6dof used 25,000 EPF, hdr_boxes, hdr_poster and poster_6dof used 20,000 EPF, dynamic_6dof used 15,000 EPF and shapes_6dof used 4000 EPF. The choices of EPF are primarily based on the level of texture of the scene. For example, a higher EPF is suitable in scenes with a high level of texture while a lower EPF is chosen when the scene is simple.

2.3.3 Evaluation

The goal is to investigate the quantity and quality of features and the quality of trajectories in 3D-space. Because it is not unreasonable that the features found through each sensor are different in both quantity and quality, we want to minimize the effect of the back-end responsible for refining the trajectory when new frames are produced. To do this, a short, five second segment, is examined for both the slow and rapid motion sections in each sequence. To perform a fair comparison between different sequences, we run UltimateS-LAM five times on each sequence. Each individual run produces an estimated trajectory and feature logs on a frame to frame basis. The quality of each run is measured by finding the mean position error (MPE) and normalizing with respect to the length of the run to get a mean position percentage error (MPPE) so we have a quantity invariant to the varying lengths of trajectories between different sequences. Since the ground truth is sampled at a higher frequency than the estimations are made, the ground truth is linearly interpolated to match each estimate in time. Let $\boldsymbol{p}_{gt}^i$ and $\boldsymbol{p}_{es}^i$ be the ground truth and estimated position at frame index i of N frames. To determine the length of the trajectory, we aggregate the non-interpolated ground truth translations $\boldsymbol{p}_{gt}^j$.

$$
\begin{aligned}
L &= \sum_{j=1}^{N_{gt}} \left\|\boldsymbol{p}_{gt}^j - \boldsymbol{p}_{gt}^{j-1}\right\|_2 \\
\text{MPE} &= \frac{\sum^N \left\|\boldsymbol{p}_{es}^i - \boldsymbol{p}_{gt}^i\right\|}{N} \\
\text{MPPE} &= \frac{MPE}{L}
\end{aligned}
$$

Rotation is omitted as an IMU, providing translational and angular acceleration, is used and the system always have a sense of rotational direction due to gravity. The IMU can not be disconnected as it is involved in virtual frame motion compensation. Features are logged during the use of gray scale frames and each virtual frame method. Features are categorized whether a feature has been successfully tracked across a set number of frames or not. For each sequence, out of the five runs collected, the median run based on the MPPE is chosen as the sequence's representative run.

UltimateSLAM trajectory estimates use the first frame as a point of reference while ground truth is provided in another. Therefore, the estimated and ground truth trajectories must first be aligned. The rotational part of the trajectory is expressed through quaternions. A quaternion is a four element complex quantity commonly used when expressing rotation in 3D-space. The main advantage of using quaternions is that gimbal lock is avoided at the singularities where using euclidean angles have singularities. Taking the Hamilton product of two quaternions corresponds to first rotating by the first quaternion followed by a rotation by the second. The conjugate of a quaternion corresponds to a inverse rotation. As an example, rotating by $\boldsymbol{q}$ and then taking the Hamilton product of $\boldsymbol{q}$ and $\boldsymbol{q}^*$ results in no rotation. Let $\boldsymbol{R}$ be the rotation matrix describing the rotation from unity quaternion, $\boldsymbol{p}^0$ be the position, and $\boldsymbol{q}^0$ be the rotation of the first trajectory entry. Ground truth and estimated trajectories are aligned by:

$$\boldsymbol{p}^{\boldsymbol{i}}_{\boldsymbol{aligned}} = \boldsymbol{R}(\boldsymbol{p}^i - \boldsymbol{p}^0)$$
$$\boldsymbol{q}^{\boldsymbol{i}}_{\boldsymbol{aligned}} = q^i_x\boldsymbol{i} + q^i_y\boldsymbol{j} + q^i_z\boldsymbol{k} + q^i_w = \boldsymbol{q}^i \overset{Hamilton}{\times} (\boldsymbol{q}^0)^*$$

2.4 ORB-SLAM

The UZH-FPV Drone Racing Dataset [13] was recorded using a quadcopter equipped with a DAVIS346 event camera. Sequences vary from outdoors to indoor drone racing tracks and is characterized by high speed and aggressive maneuvers. The dataset contains gray scale frames, events and IMU-readings among other sensory data not used. The virtual frames are constructed as described in Section 2.2 at a constant frame rate matching the DAVIS346 gray scale sensor at around 24 Hz. To combat rotational blur, we apply a Gaussian map function to map timestamps to pixel intensity values in the virtual frames. Consider a slice of events with corresponding normalized timestamps $t_i(u, v)$ in an interval $[0, 1]$ for events at pixel coordinate (u, v). We can control the attenuation of events further from the middle of the slice by adjusting k to get more well defined lines in the virtual frames.

$$I(u, v) = e^{-k(t_i(u,v)-0.5)^2}$$

Since the timestamps are normalized between 0, 1 the events with timestamps close to 0.5 should have a high intensity value while the latest and oldest events are discarded. Intuitively this is best explained using a rotating line. The image region close to the center of rotation the image may have very few events generated while events produced further will cover a larger section of the image. The strategy is to prioritize the events happening in the middle of the spatio-temporal slice which lay on a straight line on the image plane. Attenuating the early and later events aims to mitigate the accumulation blur due to the varying level of event density during rotation.

2.4.1 Reconstructed Gray Scale Images

E2VID allows us to reconstruct a gray scale image sequence from events with the goal of keeping some of the benefits an event camera holds over a standard camera, primarily the event camera's high dynamic range and temporal resolution. We reconstructed gray scale frame sequences at the same frequency of 24 Hz that the gray scale sensor was operating at. We also used the same number of events for the reconstructed gray scale frames that was used for the virtual frames.

3 Results

3.1 UltimateSLAM

Sequence	FO	CC	CNC	VC	VNC
boxes_6dof	2.63	**2.07**	165.17	17.30	17.03
dynamic_6dof	2.65	**1.41**	1.89	8.63	9.39
hdr_boxes	3.64	**1.20**	1.24	12.01	13.08
hdr_poster	**1.64**	3.21	3.31	8.36	7.59
poster_6dof	**1.03**	1.38	1.60	10.55	10.44
shapes_6dof	**1.76**	6.69	5.82	8.56	7.91

Table 2: Percentage errors during slow sections of sequences using gray scale frames only (FO), constant frame rate with motion compensation (CC), constant frame rate without motion compensation (CNC), varying frame rate with motion compensation (VC) and varying frame rate without motion compensation (VNC).

Sequence	FO	CC	CNC	VC	VNC
boxes_6dof	2.68	**1.22**	2.80	7.01	6.78
dynamic_6dof	2.03	**1.27**	2.47	6.67	6.61
hdr_boxes	2.93	**2.65**	2.81	10.71	11.57
hdr_poster	1.38	**1.24**	1.27	11.30	11.20
poster_6dof	3.46	2.16	**1.42**	5.51	5.31
shapes_6dof	**1.91**	3.74	3.58	5.98	6.41

Table 3: Percentage errors during rapid sections of sequences using gray scale frames only (FO), constant frame rate with motion compensation (CC), constant frame rate without motion compensation (CNC), varying frame rate with motion compensation (VC) and varying frame rate without motion compensation (VNC).

Operating on gray scale data produced a more consistent error when the motion was slow and outperformed the virtual frame schemes in half of the sequences as shown in Table 2. In these sequences, using only event data resulted in a significantly higher positional error. Virtual frames using constant rate and motion compensation performed with a positional error of nearly half of the corresponding error in two of the sequences. Fixed frame rate without motion compensation resulted in the lowest positional error during the sequence hdr_poster but is comparable with the gray scale positional error. In contrast, during rapid motion, virtual frames using constant rate and motion compensation outperformed all other attempts in four out of six sequences. Using gray scale data resulted in a lower positional error in shapes_6dof shown in Table 3.

Looking at the number of tracked features, shown in Appendix B, both compensated and uncompensated virtual frames maintained the most consistency between slow and rapid motion when the frame rate was adaptive. Disregarding these, gray scale frames tend to allow for more consistently tracked features than the constant frame rate virtual frames. The number of tracked features was similar between gray scale frames and constant frame rate virtual frames in the HDR sequences during slow motion. In all cases, the total number of features identified using virtual frames was higher than the number of features using gray scale images using this particular FAST-threshold. Reacquiring these in following frames was done more consistently using gray scale frames as out of the total features, a larger portion was persistent compared to the experiments using virtual frames.

3.2 Virtual and Reconstucted Gray Scale Frames

Using gray scale images ORB-SLAM successfully maintained track during the entire flight sequence. Because the dataset only contain monocular visual data and we do not rely on any other sensors or known visual targets, the scale of the trajectory is ambiguous. Figure 4 shows gray scale, reconstructed frames from E2VID and virtual frames using the accumulation scheme discussed in Section 2.2. from different part of the flight sequence. The degrading behaviour of the reconstructed gray scale images can be seen when the event rate is low, visualized in the second column of Figure 4. In the corresponding virtual frame, the effect of having a fixed events-per-frame is apparent as the only event generating source is a person walking away from the camera and taking up a very limited part of the image. This leads to an accumulation blur and this happen whenever the level of texture changes, albeit this is an extreme example. The high dynamic range of the event camera seem to have been caught in both the reconstructed and virtual frames. The open door and windows exposing the outside, brighter environment is saturated in the gray scale images while the window scaffolding and terrain silhouette can be seen in the reconstructed and virtual frames. The virtual frames suffers from hyperactive pixels that fire events on and off at a microsecond resolution and significant noise around event generating regions where the intensity gradient is high. Since the same data is used for image reconstruction, these issues affect the quality of reconstructed frames also. In the case of virtual frames, the use of a Gaussian weight function did mitigate the accumulation blur to some extent shown in Appendix A. Using a Gaussian intensity map instead of not applying any weighting seem to result in less impact on the quality of the virtual frame when varying the EPF. In the extreme case of large EPF or rapid motions, unrealistic artifacts could arise when two different event-generating regions cross the same section of the virtual frame.

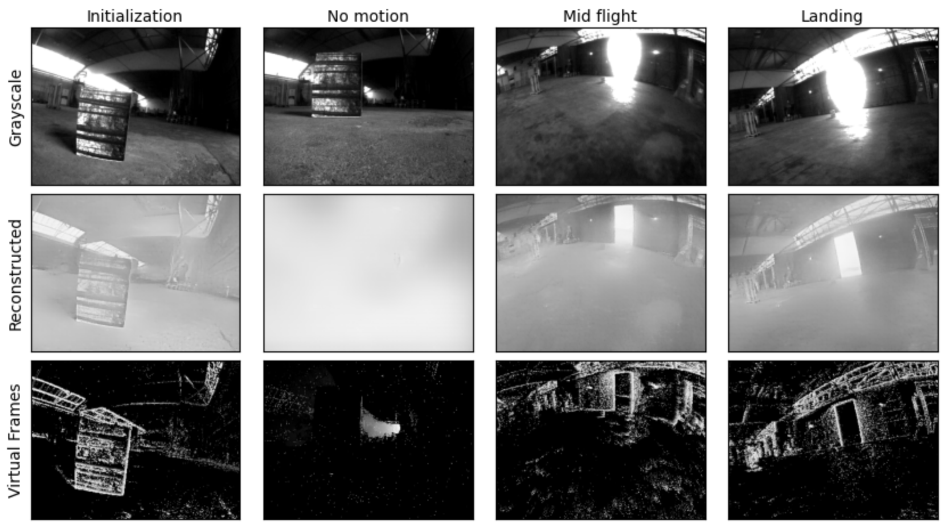

Figure 4: Gray scale, reconstructed and virtual frames from an indoor drone flight sequence.

3.2.1 ORB-SLAM - Virtual Frames

Attempts when feeding ORB-SLAM with virtual frames sourced from events using the method described in section 2.2 did not result in any meaningful trajectory data. We could not maintain a feature track throughout the flight sequence.

3.2.2 ORB-SLAM - Reconstructed Gray Scale Images

Using reconstructed gray scale images, ORB-SLAM was able to track during limited segments of the sequence but lost track during the flight sequence several times, resetting the map and trajectory.

4 Discussions

4.1 UltimateSLAM

During rapid scenarios, using virtual frames outperformed gray scale frames in all scenarios but one, being shapes_6dof. This is a scene with a low level of texture which, combined with the noisy characteristic of the event camera, proves to be a challenge as the scene's intensity gradient is flat almost everywhere, lowering the average rate at which events are generated. During slow scenarios, it is difficult to say that the event camera leads to better performance over the gray scale camera or vice versa and should be dependent on the specific scene in which the sensor resides within. Unintuitively, it is also difficult to point out a clear winner of the sequences containing HDR-environments when, in the case of gray scale frames, only a limited part of the FOV is detailed.

Settings for virtual frames, feature detection and tracking, among many other system settings can be varied and it is fully possible that the set used in these experiments was not optimal. Another source of error may be suboptimal calibration. Since the IMU is used for motion compensation for the virtual frames, any errors in extrinsic calibration parameters will result in erroneous motion compensation attempts which can artificially blur some depths in the scene or in the worst case, blur the image to the point of being unusable if ill-calibrated. Since the motion compensation method used in UltimateSLAM is optimal at the scenes' median depths, tracking may also be difficult where the depths deviates greatly. Because of these reasons, it may be possible to achieve more accurate results with a different configuration of UltimateSLAM.

4.2 ORB-SLAM

Repurposing existing VSLAM methods on event sourced visual data was more difficult than anticipated and can probably be explained by a myriad of factors. The first and foremost being the noisy nature of the sensor. A vertical line may captured by a traditional camera is captured in its entirety and rasterized onto the sensor array while in the case of an event camera, an intensity change need to cross a threshold or to produce an event at any individual pixel. If the scene contains intensity gradients close to reaching this threshold, the resulting virtual frame can be very noisy from frame to frame and a corner may not appear the same in following frames, compromising the tracking. This is especially apparent around the door in the virtual frames from the drone flight sequence in the Appendix A. There exists other sources of noise such as thermal noise and hyperactive pixels however the latter can be calibrated and compensated for to some extent.

The gray scale sensor also has the advantage of capturing the entire FOV while the event camera captures the highest gradient changes. This produces a significantly more sparse representation of the scene and tends to restrict the amount of visual features that can be located and combined with the noisy nature of virtual frames, the tracking of individual features is more difficult compared to when using a gray scale sensor. This does not necessarily mean that an event camera is unsuited for traditional VSLAM systems. The indirect method of identifying and tracking corners might not be the best approach for event-based VSLAM. Direct methods may be a better approach, solving the pose change by minimizing the photometric error instead. This also has the advantage of utilizing the entire visual data instead of creating a sparse feature representation. Direct methods employed on gray scale image sequences has previously been made real-time performing by only operating on the highest gradients of the image. Since this is given for free in virtual frames, it may be a more appropriate choice. This has been performed in EDS [14], a project fusing gray scale and event data which delivers promising results.

It is possible that ORB-SLAM may operate on event-sourced image sequences. When we tested the sequence of reconstructed gray scale images using E2VID on the event stream from the UHZ-FPV dataset, ORB-SLAM operated as intended during limited sections of the flight sequence. ORB-SLAM was used in a similar way in the article behind EDS. However, while many advantages can be gained, such as HDR and temporal control during event accumulation, it is not feasibly performed in real-time as the reconstruction of a sequence took the better part of an hour to complete.

5 Conclusions

The first aim of this thesis was to investigate the usage of frame sequences sourced from event data. The image sequences tested were virtual frames using a Gaussian temporal map and reconstructed gray scale frames using the network model E2VID. The second aim was to evaluate the positional accuracy of an event-based SLAM algorithm using four different accumulators using fixed and varying frame rates with or without motion compensation.

The event camera is not a plug-and-play visual sensor and will probably require methods tailored specifically for event-based SLAM to fully utilize the potential the sensor offers. The attempts to use ORB-SLAM on images sourced from events did not yield meaningful trajectories and probably requires further processing to work robustly. ORB-SLAM could operate in short segments on reconstructed images by E2VID and further processing may provide more desireable results. Doing this in real-time and still have computational resources left, specifically time, to perform the other stages involved in many SLAM systems remain a challenge on limited hardware.

The potential the event camera offers in terms of temporal resolution was observed in experiments using UltimateSLAM where all but one experiments resulted in a lower positional error using virtual frames during rapid motion rather than using gray scale frames. The experiment where using gray scale images resulted in a lower positional error than virtual frames was during the sequence shapes_6dof. This is most likely primarily due to the particularly low level of texture during this sequence. During slow motion scenarios, the usage of gray scale and virtual frames both resulted in a similar level of positional error. Unexpectedly, no apparent difference was recorded during slow segments in HDR-scenarios. It is possible that both calibration and system parameters could be improved further. Using virtual frames requires some level of knowledge of the scene beforehand in order to avoid over- or under-accumulation of events.

Direct approaches for VSLAM systems operating on virtual frames may be a more appropriate approach for event cameras. Because events are generated at pixels where the intensity changes rapidly, the information of where the intensity gradient is large is already known.

5.1 Future Work

A Spiking Neural Network (SNN) is a type of neuromorphic artificial neural network that aim to mimic the behavior of biological neurons by either transmitting fully or not at all determined by a threshold, similar to how biological neurons interact. Unlike traditional artificial neural networks, SNNs operate on a more event-driven basis, with information encoded in the timing and frequency of input signals. Viale et al. [15] successfully demonstrated car detection using an event camera and a Loihi chip, a neuromorphic chip, using 310 mW. Kim [2] also discussed the potential of neuromorphic processors as a front-end stage in a SLAM system and the difficulty of efficient event-based SLAM on traditional von Neumann computing architecture.

Therefore, it would be interesting to investigate the use of neuromorphic processing to associate depths to a virtual image based on the spatio-temporal information the events contain. This could produce a semi-dense depth map and effectively make the event camera into a pseudo 3D-sensor and would be appropriate for direct methods of VO/VSLAM. On traditional computational hardware, it is unlikely that this can feasibly run in real time on offline computational hardware but with the coming of neuromorphic processing units, this may be a viable method to gain depth data in real time and save a significant portion of the costs involved in traditional depth estimation techniques. To the best of this thesis's author's knowledge, only one paper presents a method to produce semi-dense depth estimation in real-time using event cameras [16]. This is accomplished by utilizing an electronically controlled liquid lens and a spiking neural network to quickly sweep the depth of focus in the scene. The events produced during the sweep is fed into a spiking neural neural network to produce semi-dense depth maps at the reported rate of 100 hz using 200 mW of power for lens control, camera and computation. Hopefully, neuromorphic hardware will see more attention in the coming years and be further studied in the context of event cameras and visual odometry.

References

[1] 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 and Automation Letters (RA-L)*, 2018.

[2] H. Kim, "Real-Time Visual SLAM with an Event Camera," Ph.D. dissertation, Imperial College London, 2017.

[3] K. Xiao, G. Wang, Y. Chen, Y. Xie, H. Li, and S. Li, "Research on Event Accumulator Settings for Event-Based SLAM," in *2022 6th International Conference on Robotics, Control and Automation (ICRCA)*, 2022, pp. 50–56.

[4] A. Z. Zhu, N. Atanasov, and K. Daniilidis, "Event-Based Visual Inertial Odometry," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5816–5824.

[5] G. Gallego *et al.*, "Event-Based Vision: A Survey," vol. 44, no. 1. Institute of Electrical and Electronics Engineers (IEEE), 2022, pp. 154–180.

[6] H. Rebecq, R. Ranftl, V. Koltun, and D. Scaramuzza, "High Speed and High Dynamic Range Video with an Event Camera," *IEEE Trans. Pattern Anal. Mach. Intell. (T-PAMI)*, 2019.

[7] B. Li, D. Zou, D. Sartori, L. Pei, and W. Yu, "TextSLAM: Visual SLAM with Planar Text Features," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[8] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-Time Single Camera SLAM," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.

[9] G. Klein and D. Murray, "Parallel Tracking and Mapping for Small AR Workspaces," in *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, 2007, pp. 225–234.

[10] Furgale, Paul and Rehder, Joern and Siegwart, Roland, "Unified temporal and spatial calibration for multi-sensor systems," in *2013 IEEE/RSJ*

International Conference on Intelligent Robots and Systems, 2013, pp. 1280–1286.

[11] M. Muglikar, M. Gehrig, D. Gehrig, and D. Scaramuzza, "How to Calibrate Your Event Camera," in *IEEE Conf. Comput. Vis. Pattern Recog. Workshops (CVPRW)*, June 2021.

[12] E. Mueggler, H. Rebecq, G. Gallego, T. Delbruck, and D. Scaramuzza, "The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and SLAM," *The International Journal of Robotics Research*, vol. 36, no. 2, pp. 142–149, feb 2017.

[13] J. Delmerico, T. Cieslewski, H. Rebecq, M. Faessler, and D. Scaramuzza, "Are We Ready for Autonomous Drone Racing? The UZH-FPV Drone Racing Dataset," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2019.

[14] J. Hidalgo-Carrio, G. Gallego, and D. Scaramuzza, "Event-aided Direct Sparse Odometry," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* IEEE Computer Society, jun 2022, pp. 5771–5780.

[15] A. Viale, A. Marchisio, M. Martina, G. Masera, and M. Shafique, "CarSNN: An Efficient Spiking Neural Network for Event-Based Autonomous Cars on the Loihi Neuromorphic Research Processor," in *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021.

[16] G. Haessig, X. Berthelon, S.-H. Ieng, and R. Benosman, "A Spiking Neural Network Model of Depth from Defocus for Event-based Neuromorphic Vision," *Scientific Reports*, vol. 9, 2019.

A Virtual Frames With and Without Weighting

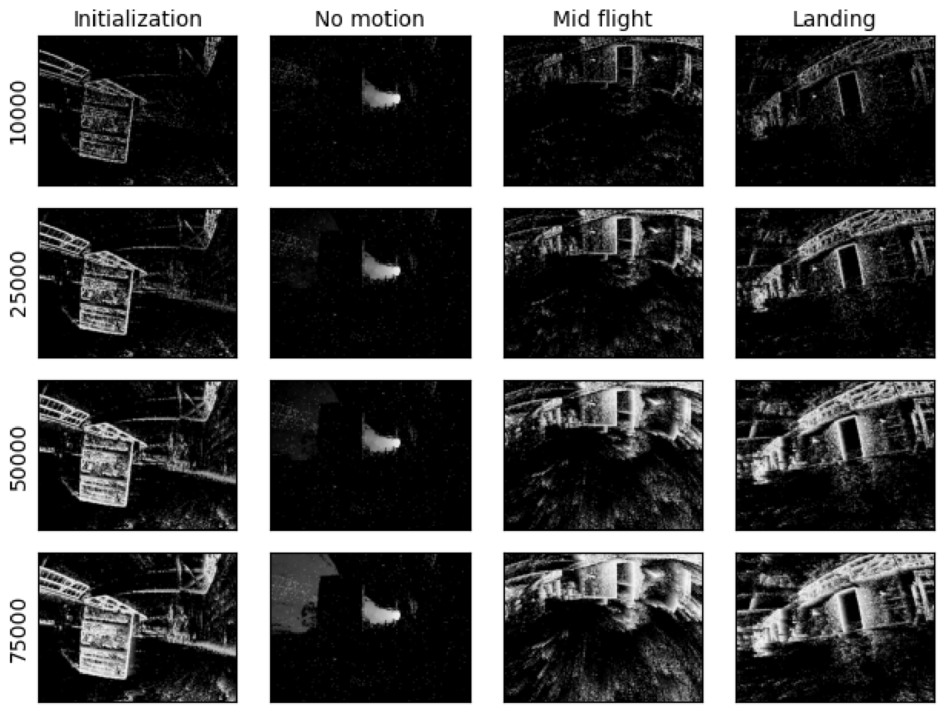

Figure 5: Virtual frames during an indoor flight sequence using a linear intensity map. Rows are sorted with respect to increasing EPF.

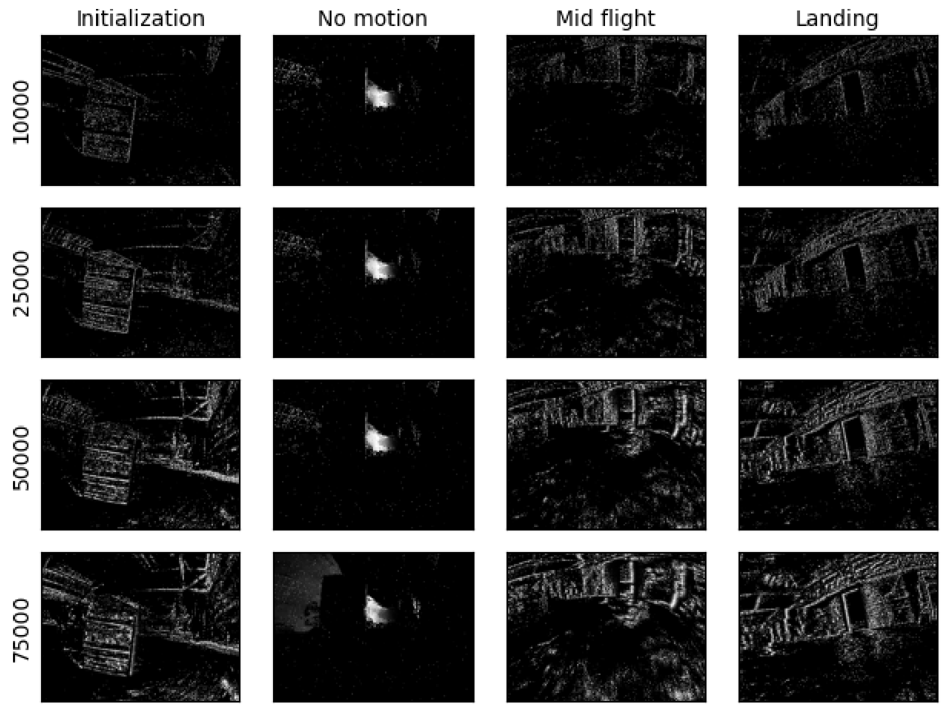

Figure 6: Virtual frames during an indoor flight sequence using a Gaussian intensity map. Rows are sorted with respect to increasing EPF.

B Tracked and Detected of Features

This section contains statistics of detected and tracked features during both slow and rapid motion across all used sequences. The figures show mean, min and max (gray) and standard deviation (black) during a five second flight sequence using a set of five different frame sequences. The frame sequences are composed by standard gray scale frames (FO) and four different virtual frame sequences. The virtual frame schemes are fixed frame rate with motion compensation (CC), fixed frame rate without motion compensation (CNC), varying frame rate with motion compensation (VC) and varying frame rate without motion compensation (VNC). This holds for all figures.

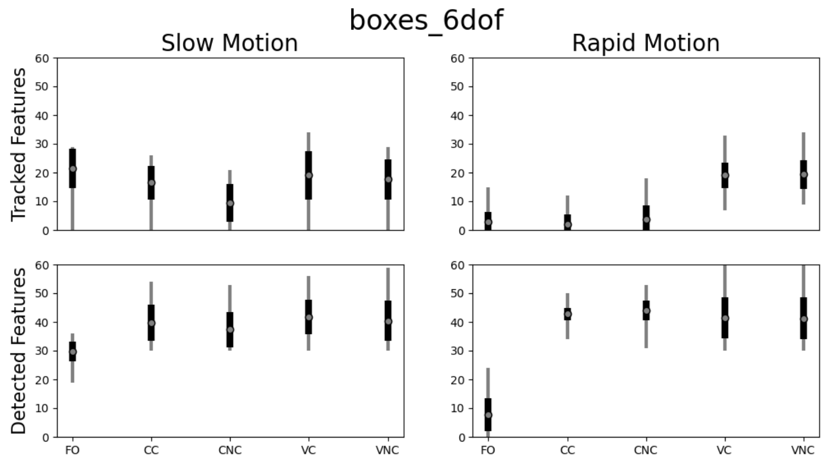

Figure 7: Feature statistics during the sequence boxes_6dof.

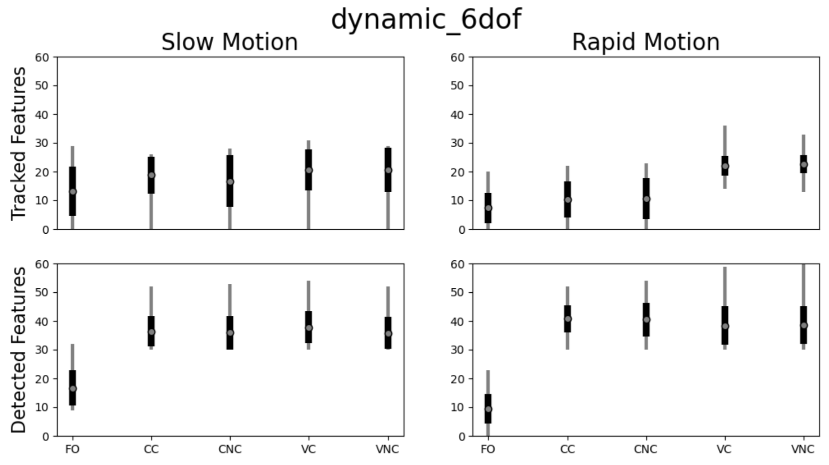

Figure 8: Feature statistics during the sequence dynamic_6dof.

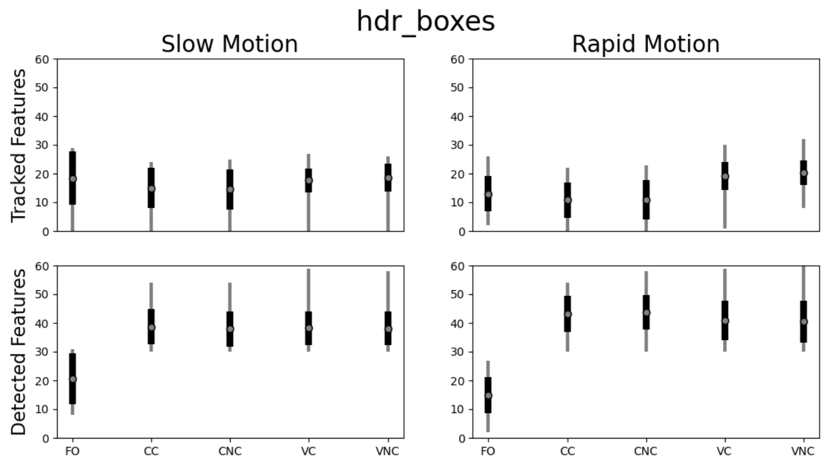

Figure 9: Feature statistics during the sequence hdr_boxes.

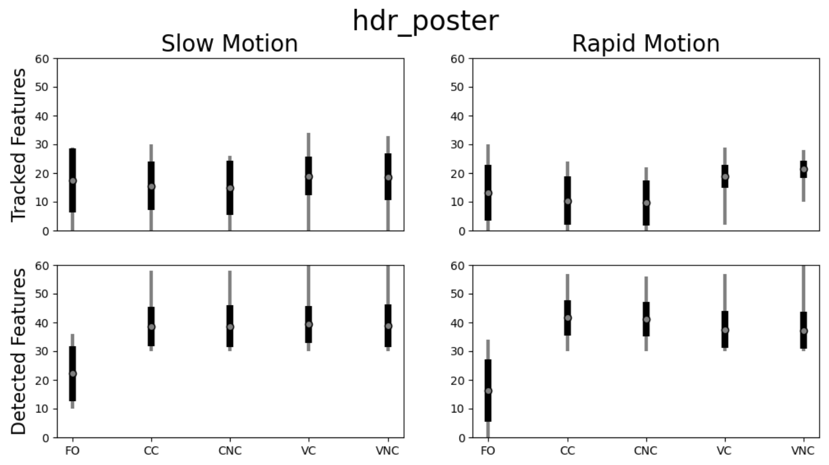

Figure 10: Feature statistics during the sequence hdr_poster.

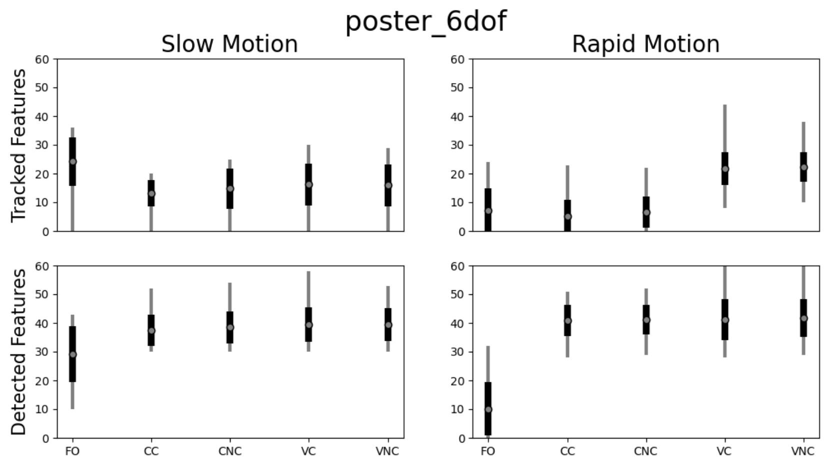

Figure 11: Feature statistics during the sequence poster_6dof.

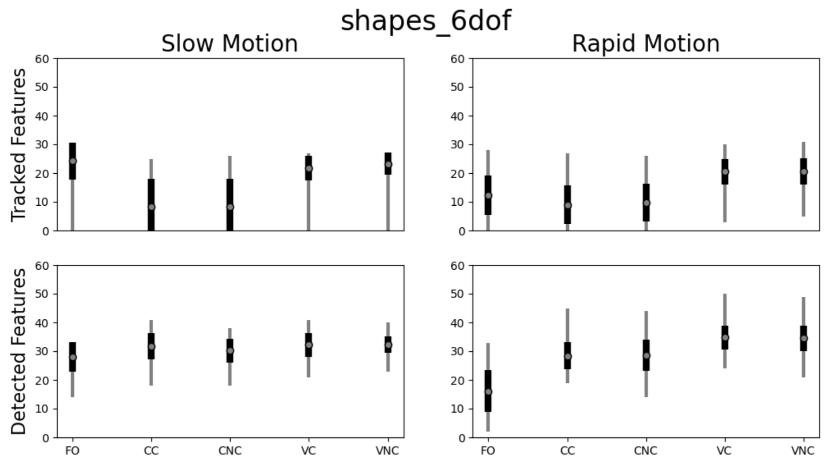

Figure 12: Feature statistics during the sequence shapes_6dof.