T.C.

Turkish-German University

Institute of the Graduate Studies in Science and Engineering

Egomotion Estimation by Fusing Events and Depth

Master Thesis

Ebru Subutay

ISTANBUL, 2023

T.C.

Türkish Deutsche Universität

Institute of the Graduate Studies in Science and Engineering

Egomotion Estimation by Fusing Events and Depth

Master Thesis

Ebru Subutay

M.Sc. Robotics and Intelligent Systems, 2023

Advisors

Assist. Prof. Dr. Naime Özben Önhon

Prof. Dr. Guillermo Gallego

Submitted to the Institute of the Graduate Studies in Science and Engineering
in partial fulfilment of the requirements for the Master's degree

ISTANBUL, 14.07.2023

Declaration of Authencity

I hereby declare that I complete the master's thesis independently without any unpermitted external help. All materials used, from published as well as unpublished sources, whether directly quoted or paraphrased, are duly reported. Only the sources and resources listed were used. Furthermore, I declare that the master's thesis, or any abridgment of it, was not used for any other degree-seeking purpose. I give this master's thesis' publication rights to the Institute of the Graduate Studies in Science and Engineering, Turkish-German University.

Ebru Subutay

Istanbul, 14.07.2023

Abstract

For a robot to localize itself starts with perceiving its surroundings. Extracting information from the location of a robot helps to create maps, calculate the robot's overall change in position. Visual sensors provide the most meaningful information among other sensors when perceiving the world However, these sensors struggle to extract meaningful information from challenging scenarios, such as an autonomous robot localizing itself in a storage area with a rapidly changing background or an autonomous vehicle exiting a tunnel. This study endows autonomous vehicles with a perception system that will improve operation in challenging scenarios without the need for alternative sensors,(e.g., LIDAR , GPS), thus decreasing costs. Furthermore, this study aims to reduce the injuries and casualties in human-robot interaction. Event cameras offer crucial advantages compared to traditional cameras in the above-mentioned scenarios. Event cameras are sensors that are inspired by nature, that mimic the human eye. Although these sensors are not unusual to robotic perception, techniques for processing the unique output of event cameras in situations such as localization and egomotion estimation are still non-mature. Specifically, this thesis proposes to fuse data from an event camera and a depth camera to perform pose estimation based on computer vision techniques. The aim is to leverage the complementary characteristics of event and depth cameras to build the perception system of an autonomous agent. Event cameras provide high-speed response while depth cameras simplify the estimation of the 3D structure of the scene. The depth camera assists the event camera in providing the third dimension to gather a 3D representation of a scene thereafter these 3D coordinates can be used for pose estimation. The goal is to provide a robust egomotion system so that motion planning and control operations can be reliably performed afterwards. It is intended to compare the characteristics and performance of the proposed approach with other works to advance the understanding of the problem and its solution. A well-known method Iterative Closest Points, that was originally used to match the Point Clouds, has been used. The results of the point cloud matching are transformations, that will match them to each other, between two consecutive point clouds. The output of this point cloud matching can give the path thus, egomotion estimation of the camera. By identifying the advantages and shortcomings via mentioned method this study will guide future developments in this field.

Keywords: *Event Camera, Sensor Fusion, Visual Odometry, Egomotion Estimation, 3D Computer Vision.*

Özet

Bir robotun kendini konumlandırması, çevresini algılamasıyla başlar. Bir robotun konumundan bilgi çıkarmak, haritalar oluşturmaya, robotun genel konum değişikliğini hesaplamaya yardımcı olur. Görsel sensörler, dünyayı algılarken diğer sensörler içerisinde en anlamlı bilgiyi sağlarlar. Ancak bu sensörler, hızla değişen bir arka plana sahip bir depolama alanında kendini konumlandıran otonom bir robot veya bir tünelden çıkan otonom bir araç gibi zorlu senaryolardan anlamlı bilgiler çıkarmakta zorlanırlar. Bu çalışma, otonom araçlara, alternatif sensörlere (örn., LIDAR , GPS) ihtiyaç duymadan zorlu senaryolarda çalışmayı iyileştirecek ve böylece maliyetleri düşürecek bir algılama sistemi kazandırmaktadır. Ayrıca, bu çalışma insan-robot etkileşimindeki yaralanma ve kayıpları azaltmayı amaçlamaktadır. Olay kameraları, yukarıda belirtilen senaryolarda geleneksel kameralara kıyasla çok önemli avantajlar sunar. Olay kameraları, doğadan ilham alan, insan gözünü taklit eden sensörlerdir. Bu sensörler, robotik algı için olağandışı olmasa da, konumlandırma ve ego-hareket tahmini gibi durumlarda olay kameralarının benzersiz çıktısını işlemeye yönelik teknikler henüz olgunlaşmamıştır. Spesifik olarak, bu tez, bilgisayarla görme tekniklerine dayalı poz tahmini gerçekleştirmek için bir olay kamerasından ve bir derinlik kamerasından alınan verileri birleştirmeyi önermektedir. Amaç, otonom bir aracının algılama sistemini oluşturmak için olay ve derinlik kameralarının tamamlayıcı özelliklerinden yararlanmaktır. Olay kameraları yüksek hızlı tepki sağlarken, derinlik kameraları sahnenin 3B yapısının tahminini basitleştirir. Derinlik kamerası, olay kamerasının bir sahnenin 3B temsilini toplamak için üçüncü boyutu sağlamasına yardımcı olur, ardından bu 3B koordinatlar poz tahmini için kullanılabilir. Amaç, hareket planlama ve kontrol işlemlerinin daha sonra güvenilir bir şekilde gerçekleştirilebilmesi için sağlam bir egomotion sistemi sağlamaktır. Önerilen yaklaşımın özelliklerinin ve performansının, problemin anlaşılmasını ve çözümünü ilerletmek için diğer çalışmalarla karşılaştırılması amaçlanmaktadır. Başlangıçta Nokta Bulutlarını eşleştirmek için kullanılan, iyi bilinen bir yöntem olan Yinelemeli En Yakın Noktalar kullanılmıştır. Nokta bulutu eşleştirmesinin sonuçları, ardışık iki nokta bulutu arasındaki, onları birbirine eşleyebilecek transformasyonlardır. Bu nokta bulutu eşleştirmesinin çıktısı, kameranın egomotion tahmini yolunu verebilir. Bahsedilen yöntemin avantaj ve eksikliklerini belirlemek yoluyla bu çalışma bu alanda gelecekteki gelişmelere yol gösterecektir.

Anahtar Kelimeler: *Olay Kamerası, Sensör Füzyonu, Görsel Odometri, Ego-hareket Tahmini, 3B Bilgisayarla Görü*

Acknowledgement

Throughout this extraordinary journey of learning and personal growth, I have been blessed with support from numerous individuals and institutions, to whom I owe my deepest gratitude. To begin, I would like to express my graditude to Türkiye Bilimsel ve Teknolojik Araştırma Kurumu (TÜBİTAK), for their unwavering support of science and research, this master thesis has aided in scope of BİDEB-2210/C. Thanks to the German Academic Exchange Service (DAAD), I had the opportunity to be a part of TU Berlin Faculty IV: Electrical Engineering and Computer Science, Robotic Interactive Perception (R.I.P.) Group, where I found a warm welcome.

My deepest gratitude goes out to all members of the R.I.P. lab for the amazing welcome, friendship, and intellectual stimulation that has greatly enhanced my lifelong curiosity journey's Berlin stop with lifelong friendships.

I owe all I am to my unwaveringly loving, supportive, and patient family members, Çiğdem, Didem, Nesli, Semra, Eda, Fulya, Vanilya... In times of self-doubt, these strong women's confidence in me was a light that led me to stay standing.

Dr. Özben Önhon has been an excellent supervisor and I appreciate all of the help throughout this long process. Her patience and meticulous approach led the way to my accomplishments.

Finally, I would want to express my most heartfelt appreciation to Prof. Dr. Guillermo Gallego, who has been an invaluable advisor, mentor, inspiration, and source of patience during the entirety of my thesis process. Prof. Gallego's guidance in the areas of Computer Vision, Camera Geometry, and Event Cameras has been really helpful and improved the overall quality of the thesis. Without his perceptive criticism and unwavering support, I never would have finished this thesis.

Once again, to everyone I have named and the many more unmentioned, who have touched my journey in profound ways, I am very grateful that our paths have crossed.

As a final word... this work has been done all with love.

Contents

List of Symbols and Abbreviations

3D 3 Dimensional

APE Absolute Pose Error

AoV Angle of view

CCD Charge-Coupled Device

CMOS Complementary Metal-Oxide Semiconductor

DoF Degrees of Freedom

DVS Dynamic Vision Sensor

FoV Field of View

GPS Global Positioning System

Hz Hertz

HDR High Dynamic Range

ICP Iterative Closest Point Algorithm

IR Infra-Red

LIDAR Light Detection and Ranging

µ Micro

O3D Open 3D library

RGB Red,Blue,Green (Color Camera)

RGB-D Red,Blue,Green-Depth (Depth Camera)

RPE Relative Pose Error

ROS Robot Operating System

RMSE Root Mean Square Error

sec Second

SLAM Simultaneous Localization ad Mapping

ToF Time of Flight

VO Visual Odometry

List of Figures

List of Tables

Chapter 1

Introduction

Autonomous vehicles/robots are an important industrial and societal topic, and will continue to be in the future as their usage expands to multiple areas and specialized applications [1],[2]. Rapid advancements in the areas of mobile robotics and industrial automation are required to improve the autonomy of such moving agents (e.g., vehicles, wearables, robots, etc.). Mobile autonomy rests upon technology that enables accurate navigation and localization of the moving agents.

Autonomous robots are typically equipped with multiple sensors, such as cameras, LIDAR and GPS, to perceive and interact with the environment. One of the most crucial tasks for an autonomous agent is to be able to perceive the world (e.g., build a map of the 3D surroundings) and localize itself with respect to it. This task is commonly known as Simultaneous Localization and Mapping(SLAM) or Visual Odometry(VO), a subsection of SLAM, in computer vision and robotics context.

Egomotion Estimation is an essential part of SLAM and VO. Specifically, egomotion estimation consists of determining the 6 degrees-of-freedom motion (6 DoFs: 3 for position, 3 for orientation) of a camera or sensor rig with respect to a 3D environment. While visual SLAM/VO with traditional cameras and LIDAR are mature topics, the resulting perception systems suffer from the inherent shortcomings of the sensors. This produces localization errors in difficult scenarios, such as high-speed, and high dynamic range(HDR). The key challenges identified are; illumination changes, motion blur and computational complexity. This master thesis proposes to tackle these issues with novel bio-inspired sensors: event-based cameras, or simply, *"Event Cameras"* [3]. Event cameras, also called silicon retina or neuromorphic cameras, offer potential advantages over conventional cameras in the above mentioned scenarios. Unlike conventional cameras, event cameras mimic how the transient visual perception system in humans work, by detecting and recording only intensity changes in the scene, without a global exposure or external clock.

The event camera's novel sensor technology provides,

- continuous
- fast
- robust (to changing illumination conditions)

output.

Specifically, this master thesis proposes to fuse data from an event camera and a depth camera to perform pose estimation based on computer vision techniques. The goal is to provide a robust egomotion (i.e., VO) system so that motion planning and control operations can be reliably performed afterwards.

1.1 Sensing the World

Acquiring information from surroundings is an essential task in robotics. Perceiving the world through *sensors* helps us to have a meaningful closed-loop control of every task a robot executes. From the beginning of robotics research, sensors have been researched and developed. Every living being senses the world with biological sensors, likewise, robots are sensing the world with sensors that often mimic biological ones. When focusing on visual information, an eye does so many tasks that today's conventional visual sensor technology cannot achieve.

Visual information is the most informative among other sensor outputs. Today, visual information by itself or a fusion of visual information with depth, LIDAR or event data information is widely used and researched. Visual information can be acquired by different sensors. Cameras are the most common sensors to provide it.

1.1.1 How to Acquire Visual Information?

Cameras that we use today to shoot movies, and broadcast news are capturing visual information using the same principle that E. Muybridge invented at the end of the 19th century [4]. The horse in motion was obtained via a triggering device that took pictures with different cameras at different points in time. The result is a sequence of images. A system that replayed this sequence of images produced a visual illusion of continuous motion, despite there time gaps between the acquired images.

At the end of the 20th Century, another sensor was designed to acquire visual information. Misha Mahowald & Carver Mead gathered the disciplines of electronics engineering and biology together and this led to a novel, neuromorphic sensor design, called silicon retina, which is inspired by the human retina, today it is commonly known as Dynamic Vision Sensor (DVS) [5] or *event camera* [3]. The event camera design emerged from the knowledge that the above-mentioned photographic cameras do not work in the same way as the human eye or the brain work. Mahowald wanted to model the neural structure of the human eye

on a silicon substrate. The spark of *Neuromorphic Engineering* is the work of Mahowald and Mead [6], bringing biology into chip design on silicon.

Let us take a brief look at today's commercially available cameras. *Standard cameras*, also known as *frame-based cameras*, acquire images of a scene at a fixed time interval when the shutter is lifted in front of the camera sensor and light rays hit the sensor. They are the advanced version of E. Muybridge's acquisition device. Due to sensor structure (CCD or CMOS), standard cameras produce a grid of pixels as output. Each pixel contains the amount of light (e.g., intensity) of the light rays reaching the pixel during the time interval given by the shutter speed of the camera. Standard cameras may have more than one channel, such as color cameras or multi-spectral cameras.

RGB-D (depth) cameras provide color (RGB) and depth (D) information, which are crucial for many tasks in computer vision, such as localization and mapping, object detection, recognition and segmentation. In RGB-D cameras, each pixel includes information on color and distances of the features from the camera plane which helps to understand the 3D structure of a scene. Depth can be obtained through a stereo method with an infrared projector or with a time-of-flight(ToF) sensor. Besides their outstanding performance, standard cameras suffer from data redundancy, latency and limited dynamic range due to their principle of operation (i.e., fixed imaging interval or shutter speed). This is where the event camera's novel approach comes to the rescue.

An event camera detects the light intensity changes for each pixel independently. The signals produced by each pixel are nearly continuously monitored (i.e., with µsecond resolution). The main difference between standard cameras and event cameras is that in event cameras each pixel operates asynchronously and only senses intensity changes in logarithmic scale [7]. An event camera does not need to wait for all pixels to produce an output. On the contrary, it aims to transfer the output of an individual pixel when a change of light intensity, i.e., when a change in the scene is detected by the sensor. In this way, it is very efficient and functional to be used in computer vision applications that require high speed and/or robustness to changing light conditions.

This master thesis combines the benefits of event cameras with RGB-D in order to tackle the camera motion estimation problem.

Pinhole Camera

A camera is a device that captures light by projecting a 3D scene onto a 2D plane, which is called the image plane. This projection process is described by a series of transformations. To tackle problems related to SLAM and VO it is required to know the *Intrinsic and Extrinsic parameters* or *Camera Matrix*, which contain both intrinsic and extrinsic parameters of the camera.

A camera can be mathematically modelled for a better understanding of the conversions among different dimensions. There are various mathematical camera models that describe this projection process of 3D world coordinates to 2D image plane coordinates. [8, 9, 10]. Yet, most conventional cameras can be described by the pinhole camera model [11, 12]. This model is widely used in

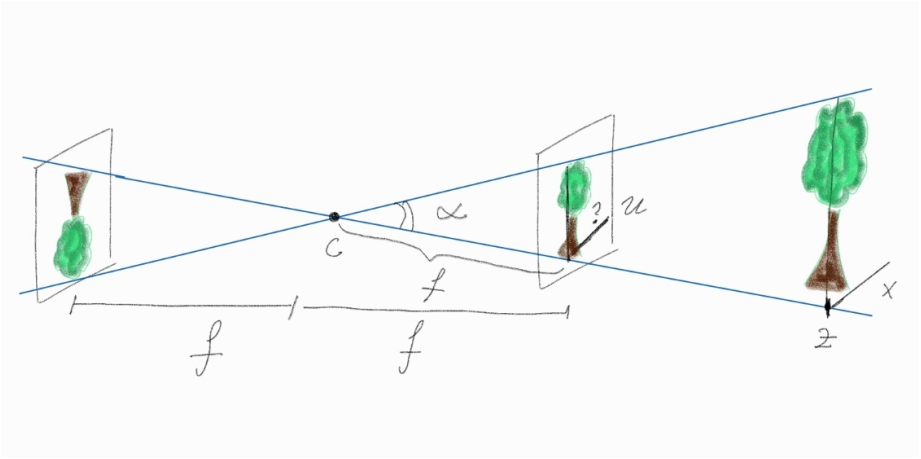

Figure 1.1: Schematic of the Pinhole Camera Model.

many applications in the field of computer vision due to its easiness of use and its reasonable outcomes.

The event camera and conventional camera that are used within this master thesis can be modeled with the *Pinhole Camera Model* shown in Fig. 1.1.

The key aspects of the Pinhole Camera Model are:

Extrinsic Parameters define the camera's position and orientation in the world coordinate system. This is also called a camera pose. When the position of the camera changes according to a reference frame, the extrinsic parameters will change. This 3D change of coordinates is often represented by a *Transformation matrix*.

Intrinsic Parameters refer to the internal characteristics of a camera. We assume these parameters remain constant regardless of the camera's position and orientation in the world. The parameters are:

- Focal Length f: it is the distance between the camera's optical center (center of projection) and the image plane.
- Principal point coordinates cx and cy: they are the location where the optical axis of the camera intersects the image plane.
- Skew angle: it is the angle between the x and y pixel axes on the image plane (it is typically 90 degrees with modern pixel manufacturing systems).
- Pixel size.

Lens Distortion Parameters, which are introduced into the image projection process, due to the imperfections in camera lenses that can lead to distortions in the captured image. Typically consisting of radial and tangential distortions,

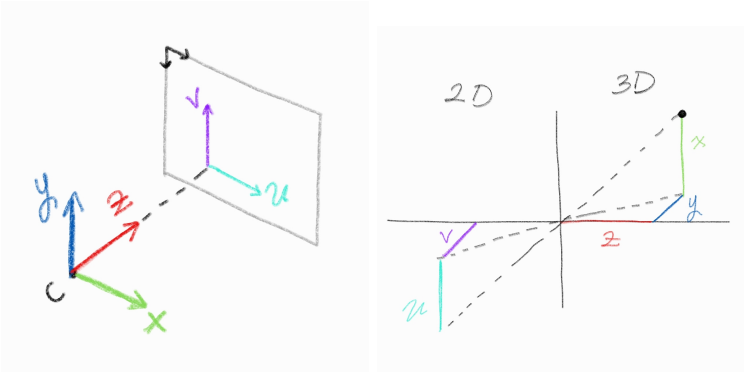

Figure 1.2: Perspective projection of a point, from 3D (world) to 2D (image plane).

this parameter must be accurately adjusted to ensure precision in computer vision tasks.

Mathematical Model of a Pinhole Camera

In the pinhole camera model represented in Fig. 1.1, rays of light from points in the 3D world are projected through a small aperture, called the centre of projection (pinhole), and onto the 2D image plane.

In order to project the 3D world into the 2D plane with a pinhole camera, there are three frames of reference to consider, as shown in Fig. 1.2: the world coordinate system, the camera coordinate system, and the image coordinate system.

- The world coordinates describe the physical 3D world independently from the camera's perspective and intrinsic parameters. The world coordinates to determine a global frame of reference.
- The camera coordinates, in which the camera is stationary and the centre of projection is at the origin, is used to simplify the representation of objects (e.g., points) in the scene, as seen from the camera's perspective.
- The image coordinates are two-dimensional, describing the location of points projected on the image plane of the camera.

The image plane is where the actual pixels of the image reside. The origin is typically represented at the upper left corner of the image, as shown in Fig. 1.2 left side, with the x-coordinate increasing towards the right and the y-coordinate increasing downwards.

The representation of a 3D point in world coordinates is: $Q = (x, y, z)^\top$ (Euclidean coordinates). When this point Q is projected onto the 2D plane, its image plane coordinates are: $q = (u, v, 1)^\top$ in homogeneous coordinates.

The relationship between the world coordinates and the image plane coordinates is described by a transformation that includes a rotation matrix and a translation vector (e.g., the extrinsic camera parameters). The transformation combines extrinsic and intrinsic parameters.

The **extrinsic parameter matrix** describes the location and orientation of the camera in the world coordinate system, while the **intrinsic parameter matrix** describes the properties of the camera's image-capturing process, such as the focal length and the Field of View(FoV). By using the extrinsic parameter matrix, points in world coordinates can be converted to the camera coordinate system, while the intrinsic parameter matrix converts points from the camera coordinate system to the image plane coordinate system. These two matrices are used together, also known as the camera projection matrix, to transform points in the world coordinate system into the image coordinate system. Understanding these transformations is important in computer vision and motion estimation applications, as this projection process allows us to correctly interpret and analyze the images captured by a camera.

This projection process converts the coordinates of a 3D point into the coordinates of a 2D image. It does this by scaling the coordinates by focal length, and shifting the coordinates so that the origin is located at the top left corner of the image, rather than at the centre of the camera.

The extrinsic parameter matrix is the 4×4 matrix.

$$T = \begin{pmatrix} R_{3\times3} & t_{3\times1} \\ 0_{1\times3} & 1_{1\times1} \end{pmatrix} \tag{1.1}$$

and the intrinsic parameter matrix K is the 3×3 upper triangular matrix.

$$K = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}, \tag{1.2}$$

where the parameters c_x and c_y are known as the principal point coordinates and represent the position where the optical axis of the camera intersects the image plane.

In most cases, the principal point is not located at the exact center of the image. The focal length, represented by the parameter f (f_x and f_y since there are two dimensions), determines the amount of zoom and is based on the specific camera lens and sensor being used. It is important to note that the units of f must be compatible with the units of the other parameters and that, in practice, the 3D point coordinates are often expressed in meters or millimetres and the image coordinates are expressed in pixels.

The mapping from 3D points in camera coordinates to 2D points on the image plane, as a non-linear transformation involving division by the depth (i.e., Z) coordinate, which makes the equation non-linear can be written as a simple linear transformation using homogeneous coordinates. This transformation assumes that the intersection of the image plane with the optical axis (i.e., principal point) coincides with the origin of the image plane coordinates. In practice,

this may not be the case, so one needs to apply a shift to the transformation to account for the offset of the principal point. The resulting transformation, which expresses the mapping in homogeneous coordinates, is called the perspective projection equation of the pinhole camera model, shown in homogeneous coordinates below.

$$q = K \underbrace{(I_{3\times3}, 0_{3\times1})}_{\text{projection}} T \begin{pmatrix} Q \\ 1 \end{pmatrix}, \tag{1.3}$$

where Q is a point in the 3D world coordinate system. $Q = (X, Y, Z)$.

T is the transformation matrix that translates and rotates the world coordinates into the camera coordinate system. It is a 4x4 matrix, consisting of a 3x3 rotation matrix and a 3x1 translation vector.
$(I_{3\times3}, 0_{3\times1})$ is a projection matrix that drops the Z-coordinate after transformation, leaving only the X and Y coordinates.
K is the intrinsic camera matrix from Eqn. 1.2.
The final result is q, a point in the image plane of the camera. This model is an approximation that works well for pinhole cameras, where the light from a point in the scene passes through the pinhole and projects onto the image plane.

To convert a point in the world coordinate system to the image coordinate system, we would first use the extrinsic parameters to transform the point to the camera coordinate system, then project, and then use the intrinsic parameters to transform the point from the camera coordinate system to the image coordinate system. Understanding these coordinate systems and transformations is important in computer vision and robotics applications, as it allows us to correctly interpret and analyze the images captured by a camera.

Event Camera

Event cameras have unique sensing properties. Optically (e.g., geometrically), their mathematical model is the same as the above-mentioned frame-based cameras: the pinhole camera model. There are three main types of event camera devices: the Dynamic Vision Sensor(DVS) [13], the Asynchronous Time-Based Image Sensor(ATIS) [14] and the Dynamic and Active Pixel Vision Sensor(DAVIS) [15]. Event cameras are a type of visual sensor that differs from traditional frame-based cameras in several ways. Compared to traditional frame-based cameras, that include CCD or CMOS sensors, event cameras operate differently by capturing the changes in the scene with high temporal resolution and low latency, rather than capturing full frames at a fixed rate. Event cameras generate a stream of asynchronous "events" in response to changes in the scene. These events are generated only when a change in the scene brightness is detected, resulting in low latency and high temporal resolution.

Event cameras have gained significant attention in recent years due to their unique advantages over traditional cameras, such as their high dynamic range,

low power consumption, and low latency. These advantages make them particularly suitable for applications that require fast and accurate sensing, such as robotics, autonomous vehicles, and augmented reality.

To explain the above terminology; low latency is an important characteristic of event cameras that enables them to provide fast and accurate sensing of changes in the scene brightness. Latency refers to the delay between the occurrence of an event in the scene and the detection of that event by the camera. In event cameras, the latency is typically very low, on the order of µseconds, because events are generated only when a change in the scene brightness is detected, rather than discretely capturing frames at a fixed rate.

High temporal resolution is another important characteristic of event cameras and it refers to the ability of a camera to detect changes in the scene over time, and in event cameras, the temporal resolution is typically very high, on the order of µseconds. HDR allows to capture scenes with large variations in brightness without saturating the photoreceptors.

The working principle of event cameras is based on *temporal contrast*, which refers to the ability to capture the differences of a scene in time (temporal). It is a method for detecting changes in the intensity of light in a scene by comparing the signals that are stored as log intensities in a pixel with constant observation. In more detail, each pixel of an event camera has a *photodiode*, which is a light sensor. The photodiode generates an electrical current when light hits its surface. The current generated by the photodiode is then passed through a *logarithmic response circuit* that allows the sensor to handle a wide range of light intensities. The output of the logarithmic circuit is then passed through *differentiator* that detects the change of the light intensity, rather than the absolute level of light. The signal from the differentiator is then fed into a *comparator* and a *flip-flop*. When the integrated change is more than a predetermined threshold (called contrast sensitivity) it means that an "event" will be triggered. The comparator resets the state of the flip-flop that results in a polarity value: ON (increase) or OFF (decrease). The timestamp t, pixel coordinates (x, y) (also called "address") and polarity p constitute an *event*, and the *Address-Event Representation (AER)* circuit reads out the output. The events can be transmitted with a minimal delay with 1 µsec temporal resolution. As a result, the rate often reaches kHz resolution. Event cameras employ a type of local gain adjustment through the encoding of log intensity of light changes, enabling their functionality across a wide spectrum of light changing, spanning from 2 lux over to 100k lux. In brief, when there is a change in the scene, such as an object moving or a light turning on or off, the signals from the photoreceptors change, causing a spike in the output signal of the pixel. This spike is detected as an event, which represents a change in the scene.

An event is represented by the tuple,

$$\dot{e} = (x, y, t, p), \tag{1.4}$$

where x, y are pixel coordinates, t is the timestamp and p is the polarity (e.g., sign) of the brightness change: -1 (OFF) or +1 (ON).

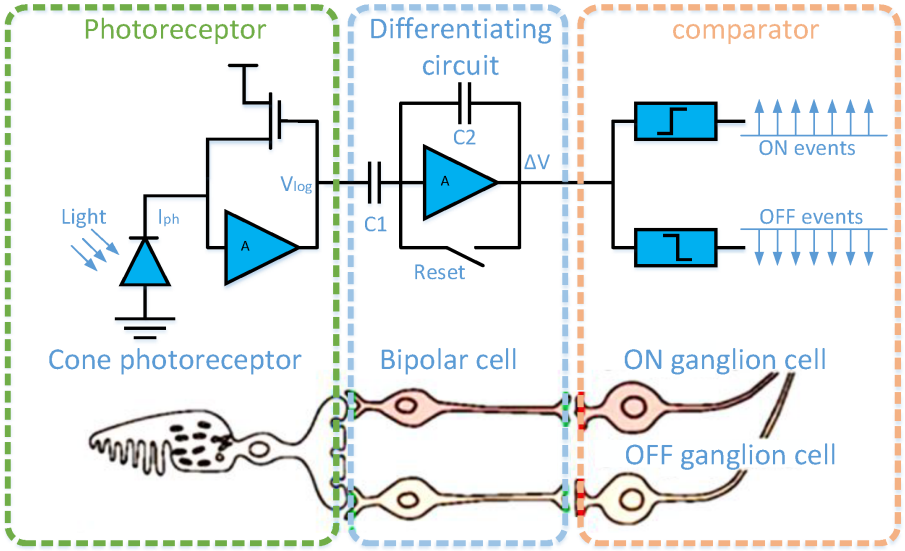

Figure 1.3: The event camera sensor DVS, inspired by human eye cells.

The temporal contrast principle takes advantage of the fact that changes in the scene occur at different rates in different parts of the image. This allows the sensor to focus on regions of the image where changes are occurring while ignoring regions where there are no changes, like the transient visual pathway of the human visual system.

Fig. 1.3, which is in courtesy of [5], shows the three-layer model of the DVS event camera pixel, comparing the bio-inspired model (bottom) with the actual circuitry (top). In comparison to the human eye, the DVS has some similarities in its sensing mechanism. Each pixel of a DVS has a photodiode, which is similar to the photoreceptor cells, rods and cones, in the human retina, which transduces light into a neural signal. The current generated by the photodiode is passed through a logarithmic response circuit, this allows the DVS to handle a wide range of light intensities similar to how the human eye can adjust to see in diverse lighting conditions, from dim environments to bright sunlight. The output of the logarithmic circuit is passed through a differentiator, that is, close aspects of the human visual system, where motion detection is prioritized over static images. Then the comparator compares the intensity change to a reference level and if it is higher than the reference, it will set or reset the flip-flop, similar to the way neurons fire: they fire that means send, a signal when the input stimulation exceeds a certain threshold. The pixel coordinates, the timestamp, and the polarity are packaged and read out comparable to how neurons transmit signals into the brain, conveying information about where and when stimuli occurred. Both rely on photodiodes/photoreceptors to detect changes in light intensity. However, the human eye has millions of photoreceptors that capture images

Figure 1.4: DAVIS346 from iniVation AG, Switzerland.

continuously, while event cameras only generate events when there is a change in the scene with photodiodes. This makes event cameras more similar to the human eye's ability to detect motion and changes in the scene. Additionally, the temporal contrast principle used in event cameras takes advantage of the fact that changes in the scene occur at different rates in different parts of the image, similar to how the human eye processes visual information with varying sensitivity across the retina.

The DAVIS contains a global shutter APS in addition to the DVS, both sharing the same photodiode array. Thus, it has a unique ability to produce conventional image frame data alongside event data, which is useful for integrating event-based vision with traditional methods.

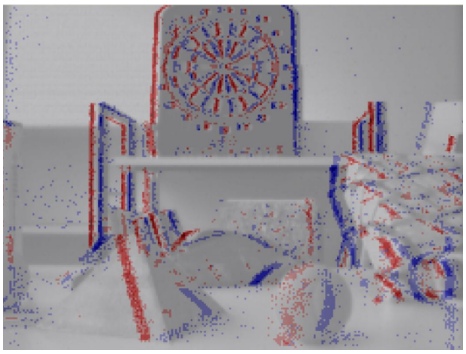

Figure 1.5: An example of an event camera output

The DAVIS346 shown in Fig. 1.4, taken from [16], is the event camera that is used in this thesis. The DAVIS346 was developed by researchers at the Institute of Neuroinformatics(INI) in Zurich, Switzerland. It is named after its resolution of 346×260 pixels. Compared to the first commercially available event camera, the DVS128 (128×128 pixels), the DAVIS346 has a relatively high spatial resolution, which can be advantageous for applications such as object recognition and tracking. However, this spatial resolution is still small compared to that of conventional photography cameras (several megapixels). The DAVIS346 is good for prototyping, as in the case of this thesis, due to its ability to capture spatiotemporally aligned events and frames (e.g., on the same pixel array and synchronized, sharing the same clock). By capturing conventional frames, geometric camera calibration with standard methods is possible.

In Fig. 1.5 An example of event camera output (in red/blue) overlaid over a grayscale frame from the DAVIS has shown. The pixels that are blue and red mean the log intensity at those pixels has increased beyond a certain threshold. Red pixels indicate negative events, in brief, illumination from bright to dark light. Blue pixels indicate positive events, a change from dark to bright.

Standard Cameras vs. Event Cameras vs. Human Eye

Event cameras and conventional cameras differ in several key aspects; event cameras operate asynchronously, capturing brightness changes at the pixel level independently when a significant change occurs, while conventional cameras work synchronously, capturing full frames at fixed intervals. This results in event cameras having a faster response time, higher dynamic range, and lower power consumption. Additionally, event cameras produce sparse data output with lower bandwidth requirements, as they only transmit relevant changes in the scene. Conversely, conventional cameras generate dense output and require higher bandwidth due to their frame-based capture. The human eye, interestingly, functions in a manner that shares characteristics with both types of cameras. Like event cameras, the human eye is highly responsive to changes in the environment and has an impressive dynamic range. The retina can detect changes in brightness and send signals to the brain, much like how an event camera detects brightness changes. This makes event cameras well-suited for applications like robotics, autonomous vehicles, and real-time tracking, where low-latency and high dynamic range are crucial, while conventional cameras are more suitable for photography and cinematographic applications. Event sensing is beneficial as it is data-driven rather than clock-driven, like standard cameras, which allow high temporal resolution and low latency. Event cameras are robust to various lighting conditions due to the fact that they rely on intensity changes rather than absolute intensity values. Tab. 1.1 shows the detailed specifications of each camera and human eye for comparison.

Parameter	Event Camera	Human Eye	Standard Cam.
Principle of Operation	Asynchronous,based on changes in brightness (events)	Asynchronous, based on changes in the visual scene	Synchronous, frame-based capture
Response time	µseconds or less	13 ms for saccades, $<$ 1 ms for µsaccades	Milliseconds (depends on frame rate)
Dynamic range	High (up to 120 dB or more)	High (up to 120 dB)	Lower (typically 60-80 dB)
Power consumption	Low, as they only react to changes in the scene	Low, as photoreceptor cells adapt to light levels	Higher, as they capture entire frames continuously
Data output	Sparse, only triggered by changes in brightness (events)	Continuous, but with higher acuity in fovea	Dense, full frames at a fixed rate
Data rate / Bandwidth	Low, only relevant data is transmitted	Moderate, brain processes selective visual info	High, full frames are transmitted
Sensitivity to motion blur	Low, reduced motion blur due to fast response time	Low, due to saccadic eye movements	Higher, due to slower response time
Low-light performance	Good, as they are sensitive to small changes in brightness	Good, due to rod cells in retina	Varies, can be poor without adequate lighting
Complexity	Typically more complex to process and interpret events	Complex, brain processes visual info	Easier to process and interpret
Applications	Robotics, autonomous vehicles, AR/VR, real-time tracking	General vision, object recognition, navigation	Photography, video recording, surveillance

Table 1.1: Comparison of cameras and human eye

1.1.2 How to Acquire Depth Information?

Depth perception is a critical aspect of visual perception that allows a human or robot to perceive the 3D structure of the world around it. Acquiring depth enables us humans and robots to estimate the distance of objects in the environment, which is important for various tasks, such as navigation and manipulation. Depth information is typically acquired using depth sensors, such as

- stereo cameras,
- ToF cameras,
- structured light sensor,
- LIDAR, etc.

those will be explained in more detail in Sec. 3.2.2.

These sensors generate point clouds, which are sets of data points in a 3D coordinate system representing the surfaces of the objects that are in the scene, or depth maps, which are 2D images where each pixel value corresponds to the distance between the camera and the object at a specific location, that provide information about the distance of objects from the camera. To acquire depth, there are different sensors listed above. In the case of visual sensing of depth with cameras, the number of cameras is important to determine the depth acquisition method.

Stereo (two or more cameras) vision is the mimicry of the human eye to acquire depth. Typically, two cameras are placed side-by-side at a predetermined distance. This distance is called baseline d. The cameras placed within a baseline, capture similar scenes. Those scenes are from slightly different perspectives. The difference in perspective yields the features in these two images to have disparity in position and orientation. This difference is larger for features that are close to the camera plane and less for features that are farther away. To find correspondences between those two images there are various methods. After correspondences are found, the difference in the coordinates of the left and right images can be used to compute the depth D of the object/feature with respect to the camera rig. In canonical stereo configuration, the formula for the depth is given by the Eqn. 1.5.

$$D = \frac{Bf}{d}, \tag{1.5}$$

where B is the baseline (the distance between two cameras attached). f is the focal length (the distance from the camera lens to the image sensor). d is the disparity (the difference between an object seen by the left and right cameras).

In the **monocular** (single-camera) case, one aspect of the scene can be acquired. Yet, it is possible to recover 3D information from 2D data over time. Structure from motion(SfM) [17] is a method that recovers depth information using the change in position of points across multiple images that are acquired at multiple viewpoints (e.g., by a moving camera). The logic behind inferring depth from these sequential images is leveraging parallax: points further away from the camera move less across consecutive images than points closer to the camera. Furthermore, from a single image, there are cues that can give information about depth.

1.2 Egomotion Estimation

Egomotion Estimation is the process of estimating an autonomous agent's motion in 3D using the sensor information it receives. The term is often used in the context of computer vision [18], yet it is not exclusively tied to image information [19]. An important aspect for a robot to localize itself is to know its own pose (position and orientation) and accumulate the actual position information over time, by its movement and thereafter actuate/control operations accordingly. Egomotion estimation with standard cameras has been widely studied in the literature. However, due to their working principle standard cameras are dependent on some aspects, such as shutter speed, and may suffer from motion blur, limited dynamic range and relatively low temporal resolution [7] On the contrary, event cameras with their novel sensor technology, demonstrated significant advantages over standard cameras. Event cameras output pixel-wise intensity changes asynchronously, as mentioned earlier in Sec. 1.1.1 in more detail. Event cameras offer a high temporal resolution, high dynamic range and virtually no motion blur [7]. Although event cameras record data at a speed that standard cameras can not record, and therefore there are applications where it can be very advantageous, event cameras have their own challenges. Handling the high volume data streams at high speed might be challenging with autonomous hardware [20].

In this master thesis, the aim is to leverage the complementary strengths of the event camera's high temporal resolution and dynamic range and an RGB-D camera's spatial and depth information to tackle the task of egomotion estimation. This combination of information aims to provide a robust and accurate egomotion estimation under challenging conditions.

1.3 Aim, Challenges and Contributions

In the literature, there are many computationally complex frameworks that perform egomotion estimation with a standard camera input. However, autonomous agents require algorithms and sensors with low computational complexity and low power consumption. Event cameras, which capture changes asynchronously in contrast to standard cameras which capture a complete image, is advantageous over standard camera also with their ability to operate in low-light conditions and high temporal resolution. By utilizing the distinct, yet, complementary data provided by an event camera and a depth sensor, this research aims to generate a robust and comprehensive representation of a given scene. In order to solve the challenging light conditions and rapid motion detection problem, the main purpose is to make the *Iterative Closest Point*(ICP), usable with the event and depth information to achieve egomotion estimation. ICP [21], which is one of the simplest point cloud registration methods, has also been used as an egomotion estimation method in some studies [22].
However, there are several challenges that must be addressed while estimating egomotion with two kinds of sensors, a synchronous one and an asynchronous

one. The asynchronous, event-driven nature of event cameras and the frame-based nature of RGB-D cameras direct researchers to develop novel algorithms for sensor fusion, synchronization and data process. The task of egomotion estimation with an event camera is challenging due to sparsity and noise in the event data, as well as lack of depth information. On the other hand, RGB-D cameras cannot handle is fast motion, extreme light conditions such as direct sunlight. When there is a need for power efficiency RGB-D cameras are not the best sensor to use. Moreover, both sensors, event cameras and RGB-D cameras struggle with featureless surfaces. Since the challenging task, fast motion in indoor environments with reflective surfaces is the concern of this study, both sensor characteristics yield a complete system and both sensors are perfectly suited choices.

The novelty of this study is to create a perception system of an autonomous agent in accordance with novel autonomous driving studies. The sensors required to create this system are different from the existing studies in the literature since there are still ongoing problems with existing structures such as costly sensor infrastructure, delays in control of the systems, instabilities, etc. [23] With an event camera and a depth camera, a more robust system is assured due to their complementary characteristics.

To shorten what this section mentions, this study adopts an innovative method of merging events and depth to enhance the understanding of egomotion estimation. The goal of the study is to improve the precision of the estimating process and to investigate the computational consequences of this approach. This technique has applications beyond the area of autonomous vehicles and might be used in augmented and virtual reality settings. Moreover, the information to be obtained via the designed platform will be able to use for positioning, mapping and autonomous driving applications. The findings of this study should help inform future studies in the field and prepare the way for the creation of more effective, real-time applications. More information about the study's methodology, conclusions, and potential applications will be presented in subsequent chapters.

Chapter 2

State of the Art

Egomotion estimation or Visual Odometry means understanding how a camera, robot, etc. moves in an environment, is a crucial problem in robotics and computer vision. Traditionally, standard cameras are used for this task which also comes with some problems. Conventional cameras struggle where the lighting conditions or object movements change rapidly. This is where dynamic vision sensors, and event cameras, emerge among other sensors with an unusual approach of capturing changes in the scene, which outputs a series of *events* instead of capturing every pixel in each periodic time stamp which also includes redundant features. Event cameras are faster and more precise than conventional cameras thus they are well suited for egomotion estimation in environments that has insufficient lighting conditions or objects moves rapidly. However, working with event data comes with its own set of challenges. Series of events are different from usual, visual inputs, such as point clouds coming from LIDARs or images coming from standard cameras. Depth information tells how far things are from the camera, adding a new layer of information. When combined with event data, it can help understand the environment better and improve egomotion estimation. In this chapter, it is planned to explore how researchers are using event cameras, combining event and depth data to tackle the problem of egomotion estimation. By the end of this chapter, it is planned to give a clear picture of where this study stands today in the exciting world of egomotion estimation using event and depth data.

2.1 Odometry

Odometry is the use of data aggregated from sensors to estimate change in position over time, as well as overall motion. The goal is to measure the displacement and orientation to some origin. To achieve accurate navigation and localization of a moving agent afterwards, perform motion planning, mapping and control. Numerous studies which used different sensor combinations could not yield results with are less prone to inaccuracies. By utilizing encoders from

the wheels to measure the wheel rotation, wheel odometry calculates the position of a robot. Wheel slip is a common problem with wheel odometry when pose estimation calculations gradually become inaccurate due to the wheels' irregular loss of tracking. [24]
An internal navigation system (INS) is highly susceptible to accumulating drift and expensive solution that combines a motion sensor (accelerometer) and a rotation sensor (gyroscope) with a relative positioning technique to provide the position and orientation of an object relative to a known starting point, orientation, and velocity. Its use alone has been attempted to replace the monocular visual odometry technique, however, it has failed and caused a significant drift error in the system.[25]
The Global Positioning System(GPS) also known as the Global Navigation Satellite System(GNSS) is a satellite navigation system that employs satellites to provide position information. Electronic receivers can correctly identify their longitude, latitude, and altitude/elevation with an accuracy ranging from a few centimetres to meters by using time signals transmitted from these satellites. Satellites continually broadcast signals; it is crucial to remember that obstacles which decrease satellite signals might interfere with GPS performance, thus it functions best where there is a clear view of the sky. Despite its widespread use, GPS systems may not always be accessible or entirely reliable in certain environments such as; tunnels, caves, city canyons, etc. as they can introduce errors in position calculations within meters which is not acceptable for safe autonomous navigation. [26]
Laser sensors are used in remote sensing technology to measure distances by transmitting a laser beam toward a target and examining the reflected light when the transmitted signal returns to the receiver. Time of Flight(ToF) or phase-shift methods are used in laser-based distance measurements. In a ToF system, a short laser pulse is sent out, and the time this laser takes to return is measured. This type of sensor is frequently referred to as laser radar or Laser Imaging Detection and Ranging(LIDAR) sensor. A drawback of LIDAR is that compared to other sensors, it offers a highly expensive solution. In addition, the analysis of LIDAR data has a high computational cost which may limit the ability to run in real-time applications. Furthermore, scanning can fail when the material appears as transparent for the laser, such as glass, because the reflections on these surfaces might generate suspicious output[27]. Besides, LIDAR has advantages such as high speed and accuracy if proper conditions are satisfied. [28] LIDAR is extensively used in obstacle detection and avoidance, mapping, as well as 3D motion capture. LIDAR can be combined with GPS and INS similar to other techniques to improve the accuracy and precision of outdoor positioning applications. [1]

2.1.1 Visual Odometry, Egomotion Estimation with Standard Cameras

Visual Odometry(VO) is the process of estimating your position and orientation of the ego-motion of an agent (e.g. vehicle, human, robot) with respect to an

initial reference frame by tracking visual features via sensors such as an optical camera itself [18] or optical camera with some variety of different odometry techniques' support that are mentioned in the Sec. 2.1. VO operates by incrementally estimating the position of the vehicle by examining the changes via images taken by onboard cameras. In order for VO to work effectively, there are a few requirements. Firstly, the environment should have adequate illumination to provide clear images. Additionally, the scene should be relatively static and contain enough texture to enable the extraction of apparent motion. Lastly, capturing consecutive frames is crucial to achieving an accurate estimation of the pose of the vehicle. *Egomotion Estimation* is the process of estimating the motion of an agent relative to the world. Egomotion estimation is a significant focus with many computationally complex frameworks that require expensive sensors. The available frameworks differentiate in terms of precision and time consumption with a standard camera input. These frameworks typically use a variety of techniques such as Lucas-Kanade optical flow [29], Horn-Schunck optical flow [30], The Structure from Motion algorithm [12].

2.1.2 Egomotion Estimation with Event Cameras

Event Cameras have been used for robotics and computer vision applications since their first days commercially available. One of the first works in egomotion estimation proposed an Event Particle Filter algorithm for egomotion estimation using a DVS [31]. Weikersdorfer et al. present an algorithm that requires a static and manually created map for navigation. However, this is a significant restriction for real-world applications since the purpose of SLAM/VO algorithms is to create maps simultaneous to motion.

In [32], an improved version of [31] Modified Particle Filter(MPF) special for events used for estimating the position of the camera over time when there are noisy frames or events in this case. MPF attunes to event camera sensor properties (characteristics), and continuous event inputs. This Simultaneous Localization and Mapping(SLAM) approach dynamically, successively generates and updates the map during self-localization.

Ni et al. addressed the obstacle in the operation of robot manipulators on a micro-scale based on sensor feedback that the controllers get. This study proposed the method of an event-based computer vision system with a standard camera's static object detection and event camera's high-speed tracking abilities[33]. Ni et al. developed an Event-based Iterative Closest Point algorithm that operates directly on event camera output that results the tracking of a robot manipulator in 250µsec resolution via computer vision. Overall, this work made a significant advancement in the field of microrobotics and micro-scale manipulation. It successfully demonstrates the use of event cameras to overcome the major challenges associated with high-speed tracking.

Valeiras et al. proposed a novel 3D pose estimation method using an event camera [34]. In this study, Asynchronous Time-based Image Sensor(ATIS) was used, which has a 304x240 array of asynchronously operating pixels. The pose estimation algorithm assumes the model of the object and its initial pose is

known and the events are matched to these features. The motion is estimated by direct transformation and velocity estimation. The direct transformation method is conducting translation and rotations according to reference frame. Afterwards pose is updated and the transformation matrix is updated. However, this process requires high computational complexity. Therefore, this approach can not be applied to every event that occurs.

Kueng et al. utilize a DAVIS [15], introduced in 2014, which is capable of capturing events and frames [35]. The novel low-latency visual odometry algorithm uses edge-like features that are extracted using frames. Then employs continuously updated, asynchronous events for tracking these features. By computing a 3D map of the scene and tracking the 6D pose of the DAVIS in µsecs, this approach provides advancements in applications that require high speed such as autonomous navigation.

[36] propose a 6D pose tracking algorithm that uses an event camera and an internal measurement unit(IMU) without any prior knowledge. The algorithm operates asynchronously and offers pose updates at a speed proportional to camera velocity. For egomotion estimation, different sensor combinations have been used, yet RGB-D cameras and event cameras offer a notable enhancement due to their complementary features.

2.1.3 Egomotion Estimation with Depth-Augmented Events

The fusion of event and depth data for egomotion estimation is promising in the fields of Augmented Reality, Autonomous Vehicles and Robotics. The two foremost works to consider in the light of this context are Weikersdorfer et al. [37] and Zou et al. [38], both combine the event-based camera with a rigidly attached and calibrated depth sensor to perform event-based SLAM. The depth sensor often referred to as RGB-D is primarily used to simplify the mapping aspect, e.g., the construction of a 3D map of the scene. Events are augmented with depth information from the closest frame of the depth sensor. Weikersdorfer et al. [37] presents Event-Based 3D SLAM(EB-SLAM-3D), that combines a low-cost embedded dynamic vision sensor with a depth sensor, Asus Xtion Pro Live. In EB-SLAM-3D, a method that emerged from [31] employing particle filter for localization was proposed. eDVS combined with depth sensor results from a sparse stream of 3D point events, which are essentially data points that represent the 3D location of the edges of the objects in the scene. The EB-SLAM-3D processes this stream of 3D point events to generate a map. For every new 3D point event, the particle filter updates the "internal belief state" which is the current best estimation of the camera location. The "belief state" is continuously updated with each new event that enables fast processing thus, low latency.

More recently, Zuo et al. introduced DEVO [38] aiming for higher accuracy and robustness in VO applications, especially in challenging sequences with low illumination conditions, and high noise in event streams via stereo camera setup. DEVO uses thresholded time-surface maps for edge detection and semi-dense depth map extraction to handle 6-DoF motion estimation efficiently.

By combining the advantages of depth and event cameras, DEVO outperforms state-of-the-art localization methods for real-time applications.

2.2 Datasets

In the field of egomotion estimation with events, several datasets have been widely used to facilitate research. The Color Event Camera Dataset(CED), as introduced by Sheerlinck et al. [39] employs the DAVIS346 sensor, which is known for capturing high-speed and high-dynamic range visual data, considerably enhancing the richness of the information gathered for motion estimation tasks. However, it is crucial to note that this dataset lacks depth information. The absence of depth data in the CED dataset would limit a more comprehensive understanding of 3D environments. Depth cameras gather 3D information that, improves egomotion estimation accuracy and algorithm robustness.
The Multi Vehicle Stereo Event Camera dataset(MVSEC) is a comprehensive collection of data for the task of 3D perception in robotics. This dataset is unique in that it provides stereo event data, stereo grayscale data, IMU data, and ground truth from a Vicon motion capture system. The MVSEC dataset was created by the GRASP Laboratory at the University of Pennsylvania and is available [40].
Another useful dataset for egomotion estimation is VECtor: A Versatile Event-Centric Benchmark for Multi-Sensor SLAM. This dataset is available [41]. High-resolution, high-speed, and high dynamic range event data, high-frequency IMU data, and low-latency motion capture ground truth make the VECtor dataset unique. It also includes depth measurements.
For the purpose of this study, the Combined Dynamic Vision / RGB-D Dataset [42] is employed. This dataset, created by the Technical University of Munich(TUM), offers a unique combination of RGB, depth, and event data captured by an event camera, making it particularly advantageous for the task at hand. The dataset includes a wide range of real-world scenes, each with varying degrees of complexity and motion patterns. This diversity makes it an ideal tool for testing the robustness of the proposed method. The dataset includes synchronized, time-stamped data, which is a requirement for the precise integration of events and depth data within the context of egomotion estimation. Real-world trajectory data permits quantitative evaluation of the indicated approach. A depth-augmented Embedded Dynamic Vision Sensor(eDVS) records events with 128×128 pixel resolution. A PrimeSense Carmine 1.09-based active depth-sensing camera captures depth data. This dataset presents the event and depth suitable for use in egomotion estimation applications.

In conclusion, the current state of the art in egomotion estimation presents a robust and evolving landscape, where innovative methodologies like the fusion of events and depth are increasingly being explored. While traditional techniques have laid the groundwork for this field, recent advancements have begun to highlight the potential for enhanced accuracy and efficiency. Nevertheless, the existing state of the art also illuminates a series of challenges and

trade-offs, primarily concerning computational resources and the complexity of estimation methods. Monocular depth estimation, while potentially increasing the precision of egomotion estimation, concurrently increases computational demand. Similarly, augmenting the complexity of estimation methods can lead to more accurate results but may also increase computation time. Thus, while egomotion estimation has advanced considerably, there is still an opportunity for improvement. This study aims to contribute to this field's ongoing advancement and problems.

Chapter 3

Methodology

3.1 Overview

This section includes a comprehensive overview of the procedures and techniques used to conduct the research on Egomotion Estimation by Using Events and Depth.

Starting from a list of event data and depth maps, fusing these in order to acquire semi-dense event-depth maps, converting these depth maps to point clouds and input them to ICP algorithm, having the output of ICP as a transformation matrix for each consecutive point cloud, chaining the transformation matrices, drawing the path of motion from chained output of ICP to estimate the path of egomotion.

The primary objective of this chapter is to survey of strengths and weaknesses of both Event Cameras and Depth Cameras for Egomotion Estimation task, how to achieve egomotion estimation with selected method and how to optimize the estimation by going through all of the steps mentioned above.

3.2 Sensory Input

The sensory input that forms the basis of this study is obtained from two sources: an event camera and an RGB-D camera. Both of these sensors play a crucial role by providing unique set of information about the world around the agent. In Sec. 3.4 some fundamental methods for egomotion estimation will be explained. There will be a need for a sequence of consecutive input from the surroundings to achieve robust egomotion estimation. This section is dedicated to demonstrate the steps required to obtain the data essential to execute egomotion estimation.

3.2.1 Acquiring Events

Event camera, also known as DVS from Ch. 1.1.1, are sensors that are categorized as neuromorphic sensors. Neuromorphic sensors electronically mimic the neurobiological architectures present with the nervous system. To explain briefly, circuitry of DVS explained in Ch. 1.1.1, when a change in light intensity is detected by a pixel's photodiode, the change is processed through a logarithmic circuit to a differentiator. After the meaningful information extracted, events are sent to readout, ready-to-use.

How to visualize events?

Due to nature of event cameras, events give a unique representation of the world when a robot acquire its surroundings. There are many methods to extract information from events. Which method to use depends on the specific task that needs to be solved. Event cameras have a different working principle as explained in Ch. 1.1.1. Event camera asynchronously reports the pixel coordinates intensity changes rather than reporting value of intensity change in the pixel. Thus, event cameras can track changes in the μsec resolution. The event input is also flexible, data representations to suit numerous application are available.

Single Event

With this approach, events are processed immediately afterwards they are triggered, one by one. This approach eliminates the window latency [43] but also requires high computational power. Some filters or spiking neural networks can deal with these events individuality [44], [45].

Group of Events

Group of Events approach is considered in most real-life applications due to computational feasibility. For tasks that require very high speed this approach may introduce some latency. Gathering information from multiple events is advantageous for visualization. A group of events can be processed differently by chosen representation type. Those are listed in [7];

- **Event frame** also known as 2D Histogram, accumulates triggered events into an image. This is the most familiar representation type and can be used with conventional computer vision algorithms. Event frames are also useful as edge maps.
- **Time Surface**, which is called Motion History Maps in conventional computer vision, is a way of looking at the history of motion in the scene. This representation brings out the temporal details. Since Time Surface will be updated with each event triggered, some important features may be lost for some applications.

- **Voxel Grid** represents 3D pixels, *voxels*, as (x, y, t) where (x, y) are pixel coordinates and t is time stamp event occurred. This allows us to track the event flow by preserving temporal information.
- **3D point set** is useful for geometric methods since (x, y) and t are all accounted as geometric dimensions.
- **Point sets on image plane** represents events as continuously changing 2D points on the image plane. This was a common method in primal feature tracking methods.
- Constructing **motion compensated event images** depends on events and an assumption or hypothesis of motion. The logic behind this is, when a feature moves across a scene, it triggers events. These events can be warped to a reference point in time, creating a clearer, sharper image. This is a way of measuring how well the events match the predicted motion. The better alignment of warped events, the better the match. Motion-compensated images are easily understandable rather than having just events, as they reveal edges that are triggering the events. This method can be used to create familiar types of images for further processing, like feature tracking and can be applied to other event representations like point sets and time surfaces.
- **Reconstructed images** offer a more motion-stable representation than event frames or time surfaces. They are used for making inferences in high dynamic range environments.

  Event information is kept in a grid of pixels format. Thus, can be used to create Event Frames. Using a group of events provides the opportunity to process the data at hand with conventional computer vision techniques. How many events should be used to create event frames or likewise representations? There are different approaches;

  - **Constant time**: Events triggered on a determined time interval- This method establishes event frames based on a fixed temporal window. In other words, an event frame is created by adding up all of the events that happened within a given time period Δt. The number of events that occurred at the scene within the time window is represented by the intensity at each pixel in the event frame, which can be considered a 2D image. This approach is straightforward and simple to use, but it ignores spatial changes in event density and may lead to information loss if the chosen time interval is too long or too short.
  - **Fixed number of events** N_e: This approach addresses the issues of the constant time approach by accumulating a fixed number of events N_e in a batch into each event frame. Number of batches N_{batch} is

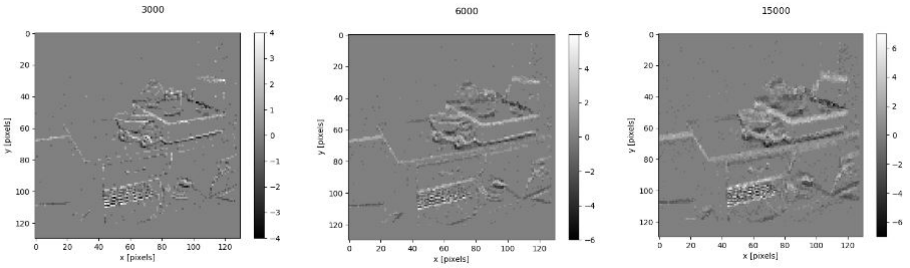

Figure 3.1: Event frames with various N_e, Left: N_e=3000 , middle: N_e=6000, right: N_e=15000.

equal to number of created event frames. This allows capturing more details in areas with high event densities. However, the timestamps of the events are ignored in this approach, and thus the temporal consistency of the data may not be preserved.

- **Adaptive area-event-number**: This is a more advanced approach that aims to balance the spatial and temporal resolution of the event frames. In this method, the image frame is divided into smaller areas or batches, and each batch accumulates a fixed number of events N_{batch}. This method provides a better representation of the data in both high-density and low-density areas, and also maintains the temporal consistency to a large extent.

The choice of how many events to use for creating event frames depends on the rate of motion in the scene, and the intended trade-off between spatial and temporal resolution all influence how many events are used to create event frames. In Fig. 3.1 there are event-frames created in different N_e. (N_e) differ depended on the task and movement speed of features. The best values for Δt or N_e in particular situations are often determined through empirical testing. Many applications mostly employ a time range of a few milliseconds or a few thousand events as a starting point. while these methods aid in the frame-like visualization and processing of event data, the asynchronous nature of the events is not fully exploited. To properly take advantage of the high temporal resolution and asynchronous nature of event data, more advanced techniques are required, such as event-based convolution or recurrent networks.

DAVIS346

The Dynamic Active-pixel Vision Sensor(DAVIS346) is a high resolution sensor, that is named after 346x260 pixels. Detailed specifications are given in Tab. 3.1, by the information from [46]. DAVIS346 is capable of simultaneously recording events and frames, enables processing of events with ease [47]. By allowing

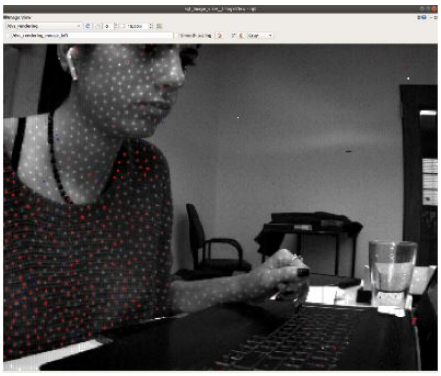

Figure 3.2: Sample APS image from the DAVIS346 captured while the RealSense IR camera is active.

to capture dynamic, event visual information alongside traditional grayscale intensity images, provides complementary information of the scene. DAVIS346 is a proper sensor to use within this study, due to capturing both frames and events. Capturing frames will facilitate the calibration process that is needed to achieve egomotion estimation task from the first step of the process.

There are some challenges to work with this sensor, such as infrared light coming from Intel RealSense D435 (which is the depth sensor used to record the dataset for this study). Sample APS image shown in Fig. 3.2 captured while the RealSense camera is active. The figure shows the dot pattern projected by the depth camera since the APS frames are sensitive to IR light, it is causing noise on event cameras recording as can be seen. IR filter has been attached in front of event camera lens to avoid this noise.

Specification	**DAVIS346**
Camera Type	DVS/APS Bi-modal
Sensor Type	DAVIS346b
Resolution	346 x 260 pixels
Pixel Size	18.5 x 18.5 μm
Field of View	70.0 x 52.5 °
Data Interface	USB 3.0
Power Supply	USB-Powered (5V)
Dimensions	79 x 30 x 29 mm
Weight	Approx. 60 grams
AER Data Output	Yes
APS Frame Readout	Global Shutter

Table 3.1: DAVIS346 technical specifications

3.2.2 Acquiring Depth

Depth acquisition can be achieved by a wide range of methods. Each method comes with characteristics that are advantages and disadvantages in different types of applications. *Stereo Vision* is a common technique that uses two cameras to capture images in slightly different perspectives, by this, mimicking the human visual system [48]. Another technique is *ToF*, involves measuring the time it takes for a light signal to travel from a source to an object and then reflect back to the receiver. This time measurement is utilized to calculate the depth or distance of the object [49]. This method provides direct measurements. Yet, it suffers from outdoor environments and reflective surfaces. *Structured light* approach uses a known light, often infrared, pattern and projects a grid or stripes onto the scene. By observing deformation of the pattern, depth can be derived [50]. This method is used in some 3D scanners and devices like Microsoft's Kinect. [51] It works well for close-range depth sensing but might not work as well over longer distances.

Depth Representation

As discussed in Ch. 3.2.2, depth can be acquired using various methods such as stereo vision, structured light, ToF or a combination of these methods. Depending on the method used, the process of capturing data and measuring time differs. Once the depth data is obtained, the process of the data into a representation form is needed for further analysis and visualization.

To represent the output of depth sensing process varies from point clouds, depth maps, meshes and other volumetric representations for representations to be understood with ease according to application. All of the representations, in particular, depth maps and point clouds are used to represent the 3D geometry of a scene, but they differ in terms of their representation and how they are generated.

A **depth map** is created by a monocular or stereo setup, encoding the distance from the camera to each pixel in the scene. In the case of a stereo camera setup, the depth map is computed by triangulating the disparities between two images captured by the cameras. Each pixel in the depth map corresponds to a calculated depth value.
In the case of Intel RealSense D435 which is a stereo depth camera that generates depth maps using a stereo camera setup. The camera has two IR cameras and an RGB camera, which are used to capture images of the scene from two different perspectives. The generation of the depth map involves the computation of the disparity between the corresponding pixels in the left and right images. The depth map is widely used because of its ease of visualization. Depth maps can be displayed on a screen using standard image display functions. It can also be saved in standard image formats or in a format that preserves the exact depth values if it will be used for further processing or analysis.

Point cloud, on the other hand, is a cluster of 3D points that correspond to the surface geometry of the scene. Each point in the point cloud is defined

by 3D coordinates (x, y, z) and, optionally, color (r, g, b). Point clouds are frequently generated by using a variety of 3D scanning approaches, including laser scanning, structured light scanning, stereo cameras, and depth mapping. This is a very flexible representation and commonly used for 3D modeling and other similar applications [52].
In the case of Intel RealSense D435, the depth map is transformed into a point cloud by mapping each pixel from the depth map into 3D space. This process involves projecting the pixel coordinates into their corresponding 3D coordinates. The distinction between an RGB-D image and a point cloud lies in the representation of the (x, y) coordinates. In a point cloud, these coordinates reflect the actual values in the real world, providing precise spatial information, whereas, in an RGB-D image, the (x, y) coordinates typically represent simple integer values corresponding to the pixels in the image. A simple model for point cloud to depth map calculation is as follows;

$$X = \frac{(x - c_x)\, d}{f_x} \tag{3.1}$$

$$Y = \frac{(y - c_y)\, d}{f_y} \tag{3.2}$$

$$Z = d \tag{3.3}$$

Where (c_x, c_y) is the optical center, and (f_x, f_y) are the focal lengths along the x and y axis. The point cloud can be saved to a file for further processing or displayed for visualization. Several libraries, such as the Point Cloud Library(PCL) [52] or Open3D(O3D) [53], provide functions for handling point clouds.

Meshes [54] are derived from depth information. Unlike a point cloud, which is just a collection of points, a mesh connects points to form a continuous surface composed of polygons (typically triangles). This provides a more solid representation of the 3D shape of the scene or object.

Volumetric representations [55] divide 3D spaces into grid of voxels, which are 3D equivalent pixel, and each voxel is assigned to a value based on depth. The data may be in a form of binary format (occupied voxel, unoccupied voxel), a signed distance or a probability.

In scope of this thesis, depth maps acquired from Intel RealSense D435 are used. Point clouds need to be generated to achieve precise egomotion estimation.

Intel RealSense D435

For fast moving or outdoor applications having a global shutter sensor is crucial. Global shutter technology is crucial for depth sensing, especially when dealing with moving objects or scenes, as it captures the entire scene at the same time, minimizing distortions. A rolling shutter captures the image line by line rather than all at once. This method can be more cost-effective and can offer certain advantages in light sensitivity, but it might introduce motion distortions if the scene or the camera is moving rapidly. Hence the need of capturing fast changes

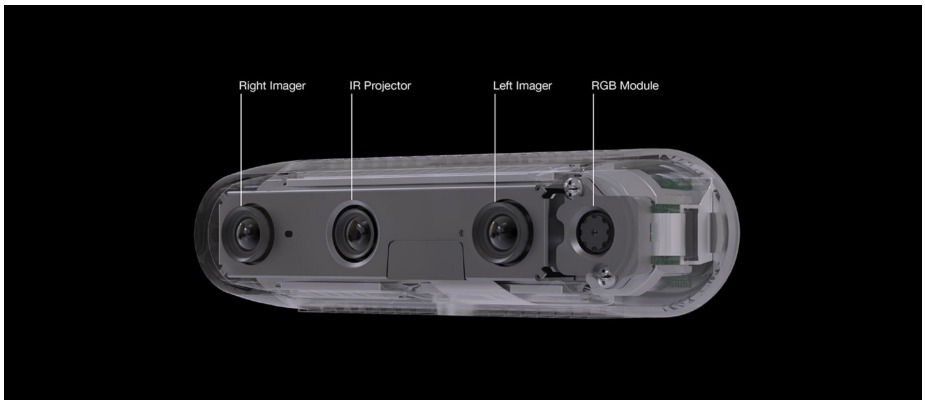

Figure 3.3: Intel RealSense D435 depth sensor PCB

in an environment features is a key aspect within this study, RealSense D435 Depth sensor stereo setup has a global shutter.

The Intel RealSense D435 depth camera, a versatile and highly efficient imaging device, has been employed in this study to acquire three-dimensional data, standard frame based images, RGB-D depth maps if necessary.

The D435 camera's ability to capture high-resolution depth and RGB images, along with its wide field of view and global shutter technology, has enabled the implementation of innovative solutions in areas such as robotics, automation, and augmented reality. The Intel RealSense D435 has 4 build-in sensors in its PCB shown in Fig. 3.3 taken from [56], in order; Right Imager, IR Projector, Left Imager, RGB Module. Since the locations of all these build-in sensors in the PCB is known, the camera itself has been calibrated. Meaning when the camera produces a depth map it can also attach RGB values to those exact pixels directly by knowing where RGB sensor is compared to other sensors.
Intel RealSense D435 depth camera employs a technique known as stereo vision to sense depth. It's somewhat similar to how human eyes work to perceive depth. The basic principle is to compare two slightly different perspectives with known distances between each other of the same scene to extract depth information. By capturing two slightly different images, calculating the disparity between the point and calculating depth according to these values yield accurate calculations of depth.
In addition to stereo vision with images, using IR in stereo vision is common and D435 includes an IR emitter. IR emitter splashes a dot pattern, which is random, over the scene and obtain depth based on warping those dots. The pattern should differ according to depth. If the dots are small the object is far if the dot is big the object is close. The pattern is a known structure so when that structure changes you can say okay that is because of the depth. In case of Intel RealSense, it uses infrared as an optional additional texture that can help but not mandatory. Nevertheless IR projector is advantageous in cases of

insufficient light conditions. Therefore It is an active method of depth sensing, as it relies on its own light source (the IR projector), as opposed to passive methods that depend solely on ambient light, gives the advantage of a better perception in changing light conditions.

D435 camera specs from the data sheet;

Specification	**Value**
Camera Type	Active stereo depth
Depth Sensor	Global Shutter, 3μm x 3μm pixel size
RGB Sensor	Rolling Shutter, 1.4μm x 1.4μm pixel size
Resolution	Up to 1280 x 720
Frame Rate	Up to 90 fps (for depth)
Field of View	86 x 57 °
Depth Range	0.2m to 10m
Interface	USB 3.0
Weight	Approx. 72 grams
Dimensions	90mm x 25mm x 25mm

Table 3.2: Intel RealSense D435 technical specifications

The information for the Tab. 3.2 is taken from [57].

Feature	**RGB Sensor**	**Depth Sensor**
Resolution	<1920 x 1080	< 1280 x 720
Frame Rate	<30fps	<90fps
Field of View (FOV)	69.4° x 42.5° x 77°	87° x 58° x 95°
Depth Range	-	0.105 m to 10 m
Technology	-	Active IR Stereo
Shutter	Rolling Shutter	Global Shutter

Table 3.3: Intel RealSense D435 RGB and Depth camera specifications

The Intel RealSense D435 uses a global shutter for the depth sensing module, which includes the left and right IR imagers. For the color sensor, however, the D435 uses a rolling shutter.

Depth values within the depth map shown in Fig. 3.4 are mapped with *jet* color map. The lowest values are depicted in dark blue while the highest values are represented in red.

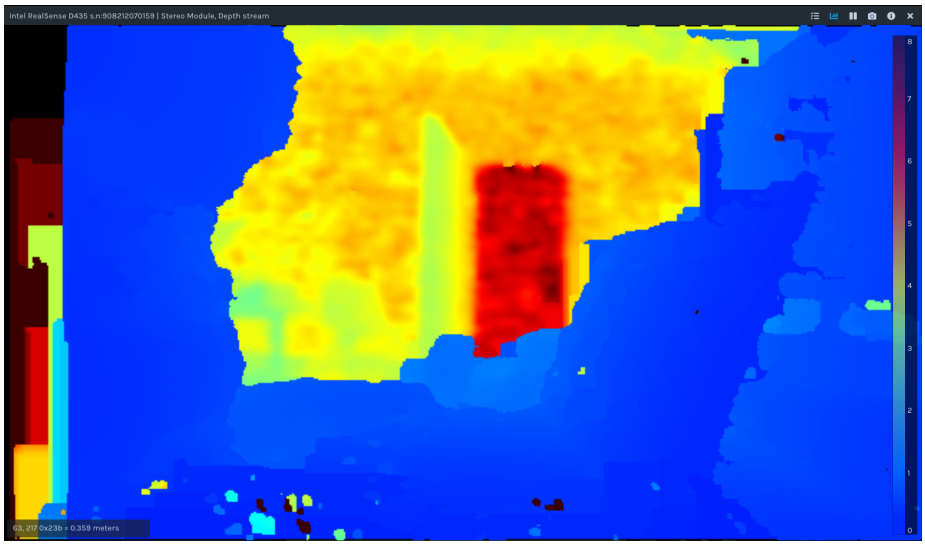

Figure 3.4: Colored Depth Map acquired from RealSense D435

3.3 Using Depth and Events to Obtain a Scene

Combining depth sensors and event cameras is an exciting research direction that leverages the advantages of both sensor types to achieve high accuracy and speed in perception tasks. Depth sensors provide accurate depth information, while event cameras offer high temporal resolution and low latency, which can be beneficial for dynamic environments and rapid motion. Within this study, an Intel RealSense D435 and a DAVIS346 have been attached to the designed rig as seen in Fig. 3.5 and Fig. 3.6.

By utilizing the distinct, yet, complementary data provided by an event camera and a depth sensor, one of the aims of this research is to generate a robust and comprehensive representation of a given scene.

Event cameras, such as the DAVIS346, record pixel intensity variations with remarkable temporal precision. These cameras, unlike their conventional equivalents, emit a continuous stream of events, each of which represents a change in pixel intensity at a specific spatial-temporal location. Their ability to handle a wide dynamic range and rapid motion makes event cameras ideal for collecting temporal data from a scene [58].

Nevertheless, event cameras are incapable of generating depth information, requiring the addition of depth data from a depth sensor such as the Intel RealSense D435. This sensor generates a depth map of the scene, with each pixel representing the distance between the sensor and objects in the scene. The depth map provides spatial information of the scene.

To combine depth and event data, the two must have synchronized timestamps. The alignment procedure begins by comparing the timestamps of the

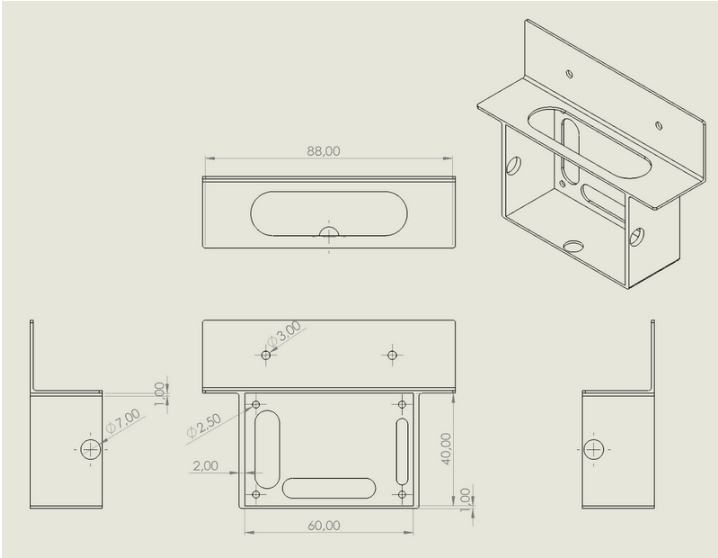

Figure 3.5: Designed rig

two sensor datasets. The depth map is then overlaid on the event data, and a depth value is assigned to each event. Point clouds differ from RGB-D images only in that the (x,y) coordinates represent actual real-world values instead of just integer values.

It is possible to create a point cloud from RGB-D images by using camera calibration. This involves using the intrinsic parameters of the scanning camera to calculate the real-world (x,y) coordinates. This produces a 3D point cloud by its 3D position and associated event [59].

This compiled data provides a complete representation of the scene, containing both its spatial configuration and dynamic variations. This information is then used by the Iterative Closest Point(ICP) algorithm to estimate camera egomotion.
Sensor specifications of the cameras that are used within this thesis are listed in Tab. 3.4.

3.3.1 Calibration

Calibrating a sensor involves making adjustments or getting parameters to make an adjustment to ensure it performs accurately and without errors. In the case of camera calibration and ego-motion estimation of a camera, parameters that are output of sensor calibration can be used for estimating the size of an object

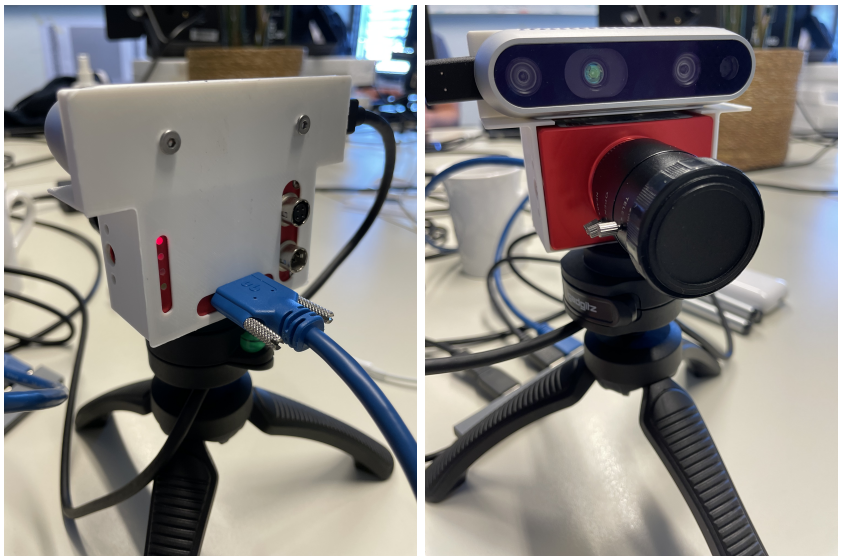

Figure 3.6: Camera configuration attached to the designed rig.

Specification	**DAVIS346**	**Intel RealSense D435**
Sensor type	Dynamic Vision Sensor(DVS) and Active Pixel Sensor(APS)	RGB and depth sensor
Resolution [pixels]	DVS and APS: 346 × 260	RGB: 1920 × 1080, Depth: 1280 × 720
Frame Rate	APS frame rate up to 25 fps	Up to 90 FPS
Data Output	Events (x, y, t, p) and APS frames	RGB and Depth frames
Interface	USB 3.0	USB 3.0
Depth Sensing	Not available (available indirectly)	Up to 10 meters
Field of View(FOV)	70° horizontal	69.4° horizontal
Power Consumption	<0.9 W (beneficial for long-term operation and mobile applications)	1.5 W (relatively high)
Shutter	DVS: none; APS: global shutter	Depth: global sh.; RGB: Rolling sh.
Streaming	DVS and APS frames	Depth and RGB
Latency	Ultra-low (up to 120 dB)	High (in μsec.)
Motion Blur	Negligible	Can be significant in motion scenes
Size [mm]	60 × 20 × 20	90 × 25 × 25
Weight [gr]	12	72

Table 3.4: Comparison of DAVIS346 and Intel RealSense D435 camera specifications

in an environment or determining the location of a camera. Camera parameters refer to the variables that describe the characteristics of a camera's lens and image sensor. These parameters can be grouped into three categories, *intrinsics, extrinsics, and distortion coefficients*. To estimate the camera parameters, it is a need to have a set of known 3D world points and their corresponding 2D image points. These correspondences can be obtained by using multiple images

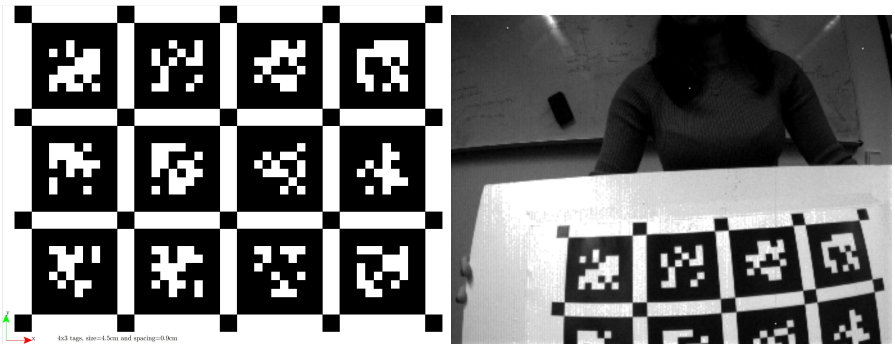

Figure 3.7: April grid design (left) and printed board (right) during camera calibration process.

of a calibration pattern, as can be seen in Fig. 3.7, such as a Checkerboard or April Grid [60], captured with the camera. Using these correspondences, one can solve for the camera parameters.

In computer vision, a general tool used for representing the frames and transformations between each other is by *homogeneous transformation matrices*. In this part of the thesis, transformations according to sensors used within this study are represented. There are four frames to be considered; World Frame, RGB Frame, Depth Frame, and Event Frame. In scope of this thesis, the problem with frame transformations is due to having multiple sensors; RGB sensor, depth sensor and dynamic vision sensor. These sensor frames are shown in Fig. 3.8 with their coordinate systems. Since both sensors are attached to the same PCB, the transformation between RGB sensor frame and Depth sensor frame is known. Transformations between RGB frame and Event frame should be attained from the output of the calibration tool. The mathematical background of these frame transformations is, **Transformation matrix** $\{{}^{\mathrm{D}}T_{\mathrm{RGB}}\}$ describes transformations between frame {RGB} relative to frame {D} where,

- *EV* implies **Event Camera (DVS)** frame
- *D* implies **Depth camera** frame
- *RGB* implies **Color camera** frame.

There is a determined Frame DVS is also known relative to frame RGB, and RGB frame is known relative to depth frame.

Translation vector that converts point ${}^{\mathrm{EV}}P$ into ${}^{\mathrm{RGB}}P$:

$$ {}^{\mathrm{RGB}}P = {}^{\mathrm{RGB}}T_{\mathrm{EV}} \cdot {}^{\mathrm{EV}}P \tag{3.4} $$

Then to transform ${}^{\mathrm{RGB}}P$ into ${}^{\mathrm{D}}P$:

$$ {}^{\mathrm{D}}P = {}^{\mathrm{D}}T_{\mathrm{RGB}} \cdot {}^{\mathrm{RGB}}P \tag{3.5} $$

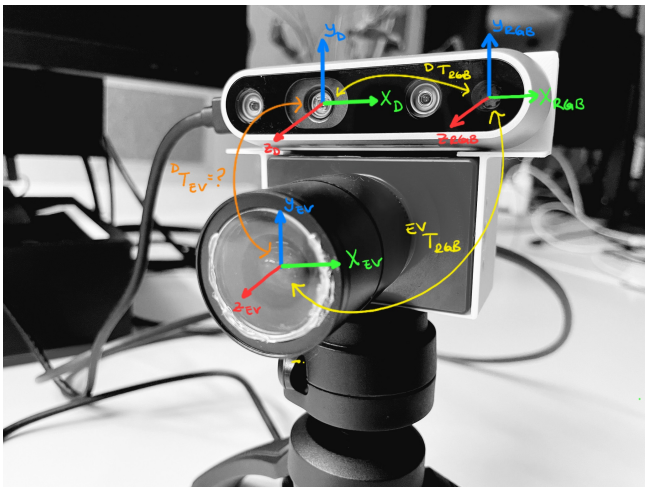

Figure 3.8: Transformation between frames

Combining 3.4 and 3.5 yields to,

$$^{\text{D}}P = {}^{\text{D}}T_{\text{RGB}} \cdot {}^{\text{RGB}}T_{\text{EV}} \cdot {}^{\text{EV}}P \tag{3.6}$$

Now frame transformation between the Depth and DVS can be defined as,

$$^{\text{D}}T_{\text{EV}} = {}^{\text{D}}T_{\text{RGB}} \cdot {}^{\text{RGB}}T_{\text{EV}}. \tag{3.7}$$

The familiarity between the sub- and superscript notation makes the manipulations simple. In terms of the known descriptions of frame $\{_{\text{RGB}}\}$ and frame $\{_{\text{EV}}\}$, transformation matrix between $^{\text{D}}T_{\text{EV}}$ can be expressed as,

$$^{\text{D}}T_{\text{EV}} = \left[\begin{array}{c|c} (^{\text{D}}R_{\text{RGB}} \cdot {}^{\text{RGB}}R_{\text{EV}})_{3x3} & (^{\text{D}}R_{\text{RGB}} \cdot {}^{\text{RGB}}P_{\text{EV}_{\text{ORG}}} + {}^{\text{D}}P_{\text{RGB}_{\text{ORG}}})_{3x1} \\ \hline 0_{1x3} & 1 \end{array}\right]_{4x4} \tag{3.8}$$

After calibrating the camera, evaluation of the accuracy of the estimated parameters in several ways. One way is to plot the relative locations of the camera and the calibration pattern. The reprojection errors and parameter estimation errors can also be calculated. Since the process is complex and open to errors, yet the calibration process is the first and most important step of sensor fusion, there are great tools to support the process.

In Fig. 3.9 an example of reprojection error of event and RGB sensors output from Kalibr[61] is shown. Kalibr is an open-source calibration tool developed by the Robotics and Perception Group at the University of Zurich and ETH Zurich. Kalibr is widely used in academia and industry for various applications, including but not limited to SLAM, VO and other perception tasks in robotics. Kalibr's open-source nature and versatile capabilities make it a preferred choice for many in the robotics and computer vision communities. However, using

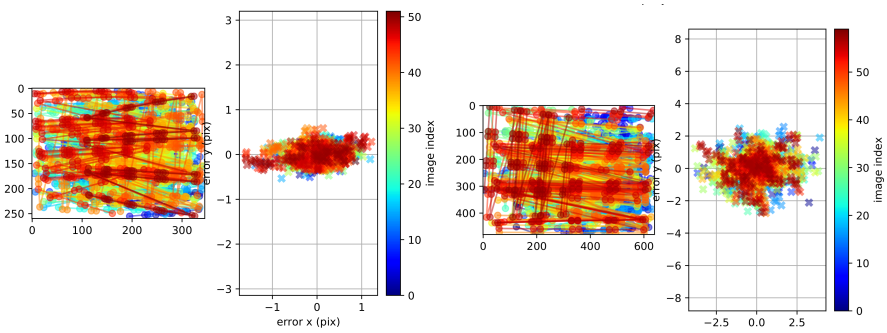

Figure 3.9: An example of calibration results of Kalibr: DAVIS364 (left), Intel RealSense D435 (right)

Kalibr requires some familiarity with robotics concepts, camera models, and the ROS(Robot Operating System) [62] ecosystem.

Kalibr supports several projection models(camera models) [63]. The camera models of the sensors within this thesis fit to *Pinhole Camera Model* [11]. Kalibr represents Pinhole Camera model parameters intrinsics vector as, $[fu, fv, pu, pv]$ where, fu, fv are focal-lengths on image plane and pu, pv are principal points on image plane.

In the context of computer vision and imaging, distortion refers to the deviation of a camera's response from an ideal pinhole model. There are two primary types of distortion: radial and tangential. Radial distortion makes straight lines appear curved near the edges of the image, and is typically more significant, especially in wide-angle lenses. Tangential distortion occurs when the lens and the image plane are not parallel, which can cause the image to appear tilted. Distortions are happening due to lens characteristics. With DAVIS346, a C-mount lens can be seen from Input Image sequence, Capture image at time T and $T + 1$, shown on the right side of Fig. 3.6 has been used. Lens parameters are 4-12 mm Focal Length, the maximum aperture of $f/1.5$, and the iris (aperture) control for adjusting aperture and can focus range of distance near or far to achieve sharp focus.

The lens used could be modeled with a *rad-tan* type of distortion. Kalibr supports the radial-tangential (rad-tan) distortion model where, distortion coefficients are $[k_1, k_2, r_1, r_2]$.

Determining the distortion and all other parameters correctly is a critical aspect in case of visual sensor fusion in order to evaluate the estimation results correctly in case of localization. Since will be discussed in Sec. 3.4 the search is not for the point cloud transformation matrix, we want to achieve the frame transformation according to point cloud transformation. Thus calibration parameters are needed in this context.

Angle of View, Field of View

Camera calibration provides parameters needed to achieve an accurate mapping from the 3D world to the 2D plane. The Angle of View(AoV) or Field of View(FoV) plays a critical role in the mapping process. AoV/FoV determine the scope of the observable world.

AoV is defined by the Eqn. 3.9.

$$\text{AoV}^\circ = \arctan\left(\frac{\text{sensor width}}{2f}\right) \tag{3.9}$$

FoV is defined by the Eqn. 3.10.

$$\text{FoV} = 2\tan\left(\frac{\text{AoV}}{2}\right) \times \text{Distance to Subject} \tag{3.10}$$

The FoV is a measurement that relies on knowing the distance from the optical center of the lens to the subject. By having knowledge of the focal length and the distance to the subject, it is possible to calculate the angle of view, and subsequently determine the field of view. Having a large field of view is advantageous at this point, it allows for capturing a wider range of textured regions in the environment. This is particularly important for achieving stable vision-based motion estimation, as the presence of more textured areas facilitates accurate tracking and estimation of motion [9].

3.3.2 Synchronization of Sensors

Synchronizing sensors involves aligning their data outputs in time. This is particularly important when using multiple sensors to capture a scene. Synchronization can be addressed both from a hardware and software perspective.

While hardware synchronization often provides more precise results, especially for high-frequency sensors, software synchronization serves as an important tool in handling scenarios where hardware-level synchronization is impractical or insufficient. On the hardware side, synchronization can be achieved through methods such as using an external clock source, which drives all connected sensors to sample data simultaneously, or through the use of a sync pulse or trigger that signals all sensors to take a reading at the same moment. Some sensors even have built-in features to facilitate synchronization.

On the other hand, software synchronization can be a feasible alternative when hardware synchronization is not possible or does not provide the required level of coordination. Software approaches may involve the use of timestamps to align the data post-collection, based on when each sensor recorded a reading. Interpolation might also be used when sensors sample at different rates, providing estimated readings for slower sensors as if they were sampling at the same rate as their faster counterparts. Using ROS for sensor data synchronization is also a common strategy.

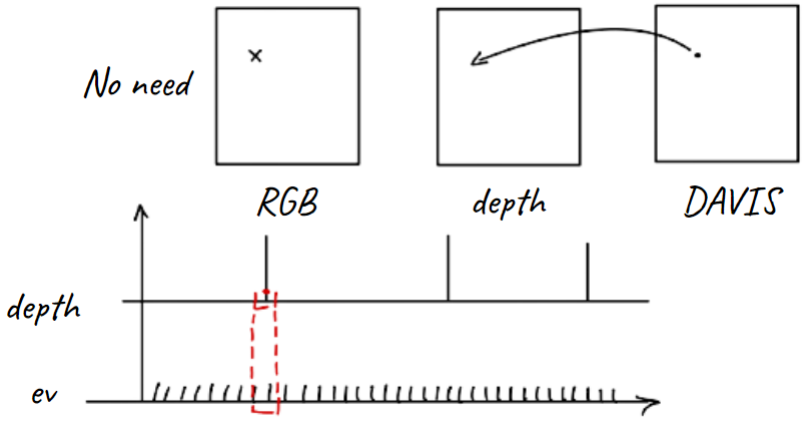

Figure 3.10: Approach to fuse events and depth.

In scope of this study, event camera and depth sensor needed to be synchronized. In Fig. 3.10 the time stamps of the measurements from both sensors are presented. The data coming from these sensors have been recorded via ROS, therefore time stamps were synchronized. Within this thesis, time synchronization was provided by ROS. Even if the time steps are synchronized, the difference between the data acquisition rates of these two sensors is still a problem. The almost continuous data capture rate of the event camera and the discontinuous nature of the depth sensor require different approaches to fuse these data. Depth data corresponding to the pixel where each event occurred is taken from the depth camera.

3.4 Egomotion Estimation

Egomotion estimation refers to the task of estimating the 6D motion (position and orientation) of a camera or sensor platform relative to its initial position and orientation.

ICP(Iterative Closest Point) is a method that can be used to estimate the egomotion of a camera or sensor platform from a sequence of point cloud data (or any data that the library/data process method that we use allows). The basic idea is to estimate the rigid body transformation between two consecutive point clouds by finding correspondences between their points and minimizing the distance between them using ICP.

To briefly explain how ICP can be used for egomotion estimation, Point cloud registration is the first step to register the consecutive point clouds using ICP to estimate the rigid body transformation between them. This involves

finding correspondences between the points in the two point clouds, rejecting outliers, and minimizing the distance between the corresponding points.

Once the rigid body transformation between the two point clouds has been estimated, the egomotion of the camera or sensor platform can be computed. This involves extracting the translation and rotation components from the transformation matrix. Here, the transformation matrix is obtained from ICP. Translation $\mathbf{t}_{3x1}$ and rotation $\mathbf{R}_{3x3}$ components are shown under Sec. 1.1.1, Eqn. 1.1. $\mathbf{R}$ and $\mathbf{t}$ components need to be extracted in order to draw the path of the estimation. The upper left $3x3$ elements of $\mathbf{T}_{4x4}$ extracted are $\mathbf{R}_{3x3}$ and upper right $3x1$ elements of $\mathbf{T}_{4x4}$ extracted are $\mathbf{t}_{3x1}$.

Since the transformation of two point clouds is unique between each other and the process is incremental, there is a need for the accumulation of these transformations to yield the estimated egomotion.

Accumulation: The estimated egomotion can be accumulated over time to obtain the trajectory of the camera or sensor platform.

ICP-based egomotion estimation has several advantages and the ability to handle non-linear motions. However, it also has some limitations, such as the need for accurate correspondences between the points in the two point clouds and the sensitivity to initialization parameters.

Point Cloud Registration

Point Cloud Registration(PCR) is a process of aligning two or more sets of points in different coordinate frames by finding a transformation that minimizes the difference between them. PCR techniques are used for 3D reconstruction, 3D localization and pose estimation [64].

Every sensor, including cameras, functions within its own specific coordinate frame when an autonomous vehicle equipped with a variety of sensors explores the environment. As a result, it is necessary to initially determine the transformation among these sensor frames. Once this transformation has been completed, the moving agent can then be localized using global coordinates.
These types of transformations are often used in computer vision applications when the goal is to align multiple perspectives of a scene or to merge various point clouds into a single, unified representation. However, PCR is often fraught with difficulties due to factors like varying densities or limited overlap between point clouds, or the presence of symmetric or incomplete point sets. These challenges can compromise the accuracy and effectiveness of the registration process, and as a consequence, inspired a lot of studies to enhance the effectiveness of PCR algorithms. Event data can often be processed to produce better understandable and more accurate outcomes since special features of event cameras assist in reducing these difficulties.
PCR challenges can be categorized into two main types: same-source and cross-source. Both groups have unique characteristics, that is demanding various algorithmic strategies for successful point cloud registration.

Algorithms of same-source point cloud registration are to register point clouds from the same source or sensor. PCR often involves problems including noise, outliers, and partial point cloud overlap. Erroneous points that do not adequately represent the scene are referred to as noise and outliers. Alignment becomes more difficult when there is partial overlap, which occurs when just a portion of two point clouds share the same physical spot.
On the other hand, when point clouds are acquired from many types of sensors or sources, cross-source PCR techniques are required. The difficulties in this situation include not just those associated with noise, outliers, and partial overlap, but also, scale and density fluctuations.
Because separate point clouds from different sources have different point densities, some point clouds may appear more or less detailed than others due to density variations. Scale variations happen when a single point in a point cloud represents a variable size or distance from one source to another. This could be because the sensors used had varied calibrations or resolutions.
Numerous PCR methods exist and can be evaluated and compared across several key aspects such as accuracy, robustness, convergence, sensitivity to initialization, type of transformation, and overall approach.
Depending on the nature of changes occurring in the scene, the type of transformation can be very important in some cases. The choice of transformation is crucial, for instance, whether a task necessitates a rigid scene or if there are moving textures, e.g., human motion analysis or dynamic object tracking.
Non-rigid registration techniques might not be the greatest fit in situations when the underlying structure is constant, e.g., camera pose estimation, robot localization, or rigid object detection. In such cases, rigid registration methods, which encompass only translation and rotation, are likely to be more suitable.

To be more clear;

- **Rigid transformation** refers to the alignment of point clouds without changing their shape or scale. This type of transformation consists of translation (moving the point cloud in the x, y, and z directions) and rotation (changing the orientation of the point cloud). Rigid transformations preserve the relative distances between points in the point cloud.An example of rigid motion in real life is moving a chair across the room. The chair may change its position and orientation but the underlying structure remains constant.

- **Non-rigid transformation** allows for more complex deformations, such as scaling, bending, or twisting of the point cloud. This type of transformation is suitable for applications where the underlying structure of the point cloud may change, like matching objects with varying shapes. Non-rigid registration methods often involve additional constraints or regularization terms to ensure smooth and plausible transformations. An example of non-rigid motion in real life is waving a flag. As the flag waves, its shape deforms, meaning the distances between different points on the flag can change.

3.4.1 Iterative Closest Points

Iterative Closest Point(ICP) is a simple algorithm, yet it is an effective method for egomotion estimation that has been used in various studies [65]. ICP minimizes the difference between two point clouds by iteratively using the beforehand estimated transformation [21]. The operation of ICP revolves around the iterative optimization of transformation estimation between two point clouds until a convergence point is reached. ICP establishes the correlation between the source and target point cloud points at each iteration of this process. Then calculates the transformation required to match the source and target point clouds using this correspondence. This iterative cycle continues until the transformation estimation either converges to a local minimum or the predefined maximum number of iterations is reached.
ICP calculates the transformation needed to align two different point clouds. Data association, also known as alignment, and transformation, based on the data association or alignment, are the two initial steps involved in this process. Consider a scenario where two sets of points are obtained, each from a distinct location while a platform moves around the surroundings. The goal here is to identify a rigid body transformation that can combine these two point clouds into a single reference frame and is made up of a rotation matrix and a translation vector.
As a result, a certain position shown in both scans will point to the same spot on world coordinates.

The basic alignment problem can be formalized as follows;
Given two input vectors refer to points in a point cloud denoted as:

$P_Y = \{y_1, \ldots, y_i\}$, $P_X = \{x_1, \ldots, x_j\}$,

The set of correspondences is given as: $C = \{(i, j)\}$

where, e.g., in y_1 in point cloud P_Y corresponds to t x_{10} in point cloud P_X.

The task is to find a translation vector $\mathbf{t}$ and rotation matrix $\mathbf{R}$ that minimize the sum of the squared errors, defined by the Euclidean distance between corresponding points in P_X and P_Y. This minimization problem can be expressed as follows;

$$\min_{\mathbf{R},\mathbf{t}} \sum_{(i,j)\in C} \|y_i - \mathbf{R}x_j - \mathbf{t}\|^2, \tag{3.11}$$

Here, the objective is to minimize the sum of distances between each point in P_Y and the corresponding point in P_X, after the next one has been transformed into the reference frame of P_Y.
The process involves identifying the optimal $\mathbf{R}$ and $\mathbf{t}$ that best align the two point sets. In practice, the resulting error is often non-zero due to noise in the measurements.

To concentrate on the points that have correspondences with the other point cloud, the point clouds P_x and P_Y are reordered based on the correspondence set C. This can be achieved using an indexing system such that x_1 in P_X corresponds to y_1 in P_Y, x_2 in P_X corresponds to y_2 in P_Y, and so on.
The transformed point sets X_n and Y_n obtained are now aligned thus, the set C is no longer necessary.

The goal becomes to find the rigid body transformation $\hat{x_n} = \mathbf{R}x_n + \mathbf{t}$ that transforms x_n to $\hat{x_n}$.

Here, $\hat{x_n}$ is the estimated target, x_n is the source and actual target is the Y_n shown in Fig. 3.11.

The newly transformed point set $\hat{x_n}$ should be as close as possible to the point set y_n. The process minimizes the sum of squared point-to-point distances.

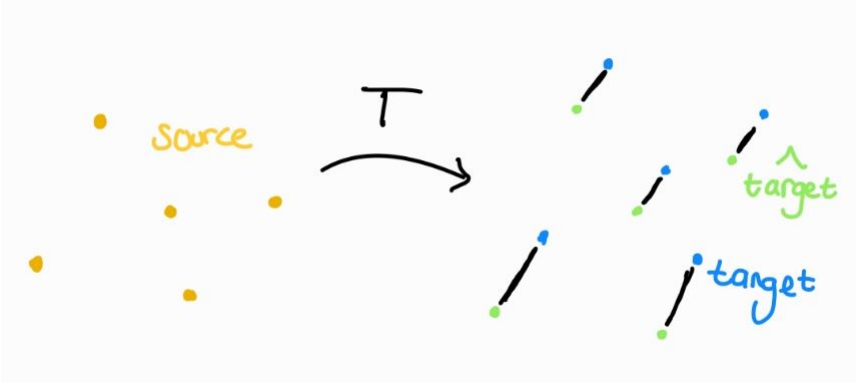

Figure 3.11: Source is transformed through transformation matrix (T) and data association between predicted target $\hat{target}$ and actual target.

To begin with point clouds, P_x and P_y
A point in 3D space is determined as;

$$P_x = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{3.12}$$

Rotated point in 3D space is;

$$P_x = {}^{y}R_x P_y \tag{3.13}$$

For transformed points (transformed and rotated) to simplify operations we use homogeneous coordinates. Where h implies to Homogeneous Coordinates and eu implies Euclidean Coordinates;

$$P_{eu} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \rightarrow P_h = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{3.14}$$

$$I_{eu} = \begin{pmatrix} u/\lambda \\ v/\lambda \\ w/\lambda \end{pmatrix} \leftarrow I_h = \begin{pmatrix} u \\ v \\ w \\ \lambda \end{pmatrix} \tag{3.15}$$

Here, I refers to the image plane.

$$P_{y_h} \rightarrow P_{x_h} = {}^{y}T_x P_{h_y} \tag{3.16}$$

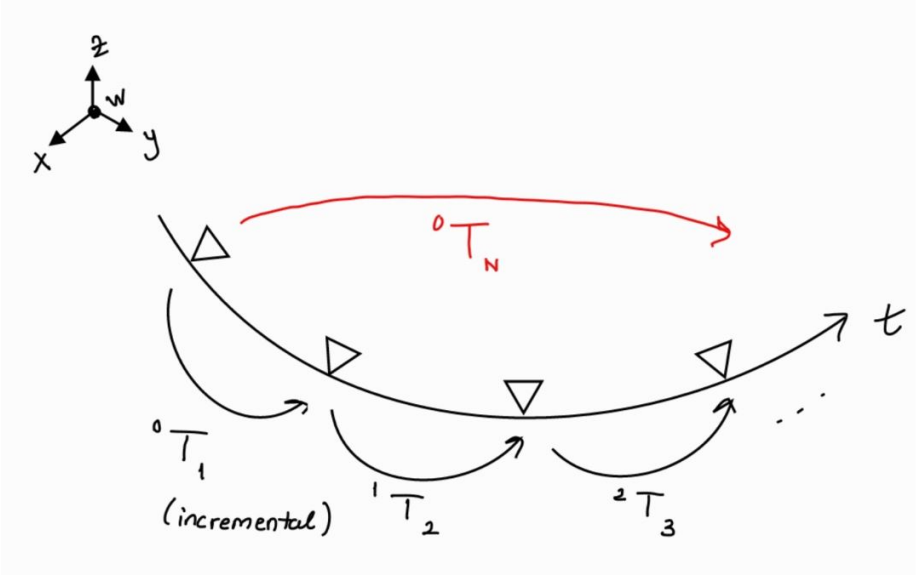

Figure 3.12: Incremental movement of a camera

In Fig. 3.12, incremental camera poses taken consecutively are shown. Absolute poses are obtained by *"chaining"* the incremental poses;

$$T_{i,0} = T_{i,i-1} T_{i-1,i-2} \dots T_{2,1} T_{1,0} \tag{3.17}$$

Iterative Closest Points(ICP) and similar "two-frame" pose estimation methods produce the incremental motion/poses.

How are those points transformed?

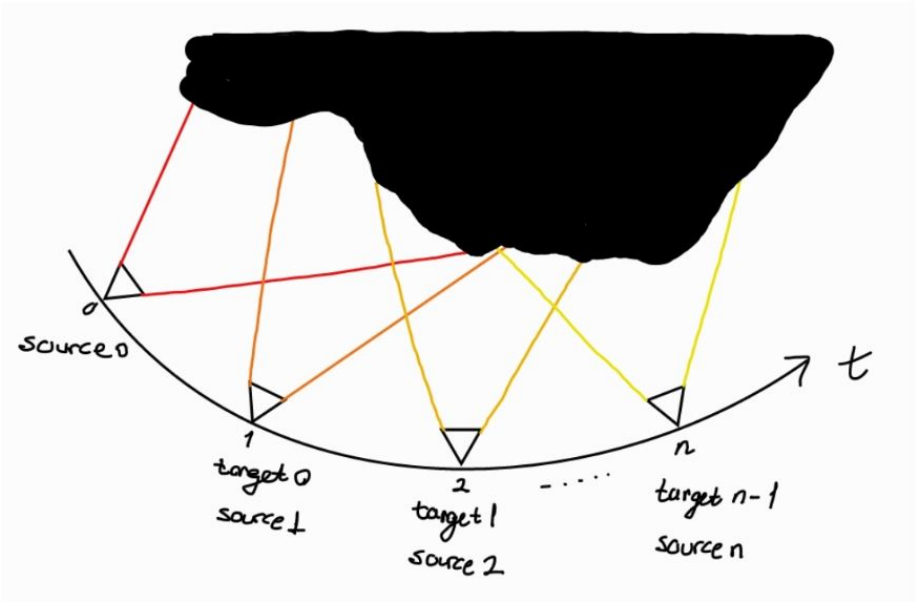

Figure 3.13: Source and Target changes for each step of ICP.

$$
\begin{aligned}
P_{h_{\text{target}}} &= {}^{\text{source}}T_{\text{target}} P_{h_{\text{source}}} \\
\text{target}_0 &= \text{source}_1 \\
\text{target}_1 &= \text{source}_2 \\
&\vdots
\end{aligned}
$$

While chaining the transformation matrices in order to achieve egomotion estimation, consecutive point clouds are changing roles as above. The visual representation can be seen in Fig. 3.13

Overlook to point cloud alignment problem,

Assumes data association is known(ICP Step 1)

$$\{{}^{h}P_{\text{target},i}, {}^{h}P_{\text{source},i}\}_{i=1}^{n_p} \mapsto \boxed{\text{Alg. “Step 2 for ICP”}} \mapsto T_{\text{target,source}} \tag{3.18}$$

$$J(T) = \sum_{i=1}^{n_p} W_i d^2({}^{h}P_{\text{target,i}}, {}^{h}\hat{P}_{\text{source,i}}) \tag{3.19}$$

where, $W_i > 0$.

All the steps above is assumed there is no noise in the source.
More complicated solution to the problem would be to assume that there is noise in both source ad target, and try to minimize the so-called symmetric transfer error [66]

Advantages of ICP:

- Effectiveness: Point clouds have been successfully aligned using ICP. ICP can reliably converge to the precise alignment when the first transformation guess is not far from an accurate answer.
- Simplicity: The algorithm's concept is straightforward and simple to use.
- Flexibility: ICP can be customized to fit a variety of needs and data kinds. Its performance can be improved for certain tasks by utilizing various distance measurements, outlier rejection techniques, or optimization techniques.
- Computational cost: Since the point cloud size is restricted to the number of events used to create the scene, the need for computational power is low.

Disadvantages of ICP:

- Sensitivity to initial guess: The performance of ICP is highly dependent on the initial guess of the transformation. If the initial guess is far from the true transformation, ICP may not converge to the correct solution, leading to an erroneous alignment.
- Outliers and noise: ICP is sensitive to outliers and noise in the data. These can significantly reduce the performance of ICP and may lead to incorrect alignments.

How to implement ICP?

There are several ICP toolboxes for different platforms which have different dependencies. Such as; Pyoints [67], PCL [52]. At last, it was easy to use Open3D [68] library for Python with bunny dataset [69]. Alignment results of Fig. 3.14 are shown in Fig. 3.15. Since ICP is dependent on the initial transformation matrix given, the first alignment is formed according to this initializer. Thus bunnies transformed with an initial transformation matrix rotates the source bunny 90° clock-wise.

Here we are trying to find a rotation matrix and the translation vector so that we can align those two point clouds.

3.4.2 From Point Cloud to Point Set

Point Cloud generated from a variety of depth sensing methods are essentially a bundle of data in 3D space. Each provides information about the 3D location

Figure 3.14: Bunnies, Unaligned

Figure 3.15: Bunnies, First Alignment

of a point $p = (x, y, z)$. To use point clouds effectively for egomotion estimation task, back-projection of the point cloud, yields the point set should be obtained.

The process of back projecting a point cloud to a point set involves several steps, each transformation must be followed to ensure the final representation is suitable to estimate the exact position of the image plane (2D representation of the scene).

Projection, Reprojection, Back Projection

Projection is the process that transforms a 3D point in the real world onto 2D point in an image, based on camera parameters mentioned in Sec. 1.1.1.

Back-projection is the reverse process of projection, where a 2D point in an image is translated back into an estimated 3D point in the world, often requiring additional data or assumptions.

Reprojection involves applying the projection process to a point that has previously been back-projected to verify the accuracy of the original back-projection, effectively comparing the reprojected point with the original 2D point in the image.

In this case, it will be used for DVS and Color camera. To determine the depth (distance from the camera) of each pixel in the image, the system looks for matching pixels in the other images and uses triangulation to calculate the depth of the corresponding scene point.

Here the search is not for the point cloud transformation matrix. To achieve the frame transformation of the image plane according to point cloud transformation there are a few steps;

1st step:

$$\begin{pmatrix} \mathbf{Z}u_d \\ \mathbf{Z}v_d \\ \mathbf{Z} \end{pmatrix} = \mathtt{K}_d \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \\ \mathbf{Z} \end{pmatrix}, \tag{3.20}$$

where $\mathbf{Z}$ refers to 3rd dimension, depth, and $\mathtt{K}_d$ is intrinsic matrix of Depth Camera, u_d, v_d are x, y coordinates in the corresponding image plane and $\mathbf{x}, \mathbf{y}$ are points in 3D space.

2nd step:

$$\mathtt{K}_d^{-1} \begin{pmatrix} u_d \\ v_d \\ \mathbf{1} \end{pmatrix} = \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \\ \mathbf{Z} \end{pmatrix} \tag{3.21}$$

3rd step:

$$\begin{pmatrix} \mathbf{x_{EV}} \\ \mathbf{y_{EV}} \\ \mathbf{Z_{EV}} \\ \mathbf{1} \end{pmatrix} = \mathbf{T_d} \begin{pmatrix} \mathbf{x_d} \\ \mathbf{y_d} \\ \mathbf{Z_d} \\ \mathbf{1} \end{pmatrix}, \tag{3.22}$$

where, $\mathbf{x_{EV}}, \mathbf{y_{EV}}, \mathbf{Z_{EV}}$ are points in 3D in perspective of Event Camera.

4th step:

$$\mathbf{Z_{EV}} \begin{pmatrix} \mathbf{x_{EV}} \\ \mathbf{y_{EV}} \\ \mathbf{Z_{EV}} \\ \mathbf{1} \end{pmatrix} = \mathtt{K}_{EV} \begin{pmatrix} \mathbf{x_{EV}} \\ \mathbf{y_{EV}} \\ \mathbf{Z_{EV}} \end{pmatrix} \tag{3.23}$$

3.5 Implementation

- Process two consecutive point clouds.
- Align each one with the next one.
- Estimate a transformation between the point clouds.
- Calculate and visualize the errors in alignment.
- Accumulate the transformations to track overall motion.

Parameters that can be changed according to fixed number of events approach are,
i_{EV} = Number of events in a batch
Total number of events = N_{EV}
i_{Batch} = Number of batches according to $\frac{\Delta N_{EV}}{i_{EV}}$

For the methodology in the context of my research, fused event data and depth data, the latter being acquired from a depth sensor, structured light sensor, depending upon the specific requirements and constraints of the application has been investigated.

Algorithm 1 Pseudo Code for Egomotion Estimation with ICP

```
Initialize max_correspondence_distance, trans_init
while True do
    if (iEv + num_events_batch > num_events) then
        break
    end if
    Select subset of events, x, y, depth data
    Create batches array with x, y, depth data
    Increment iEv by num_events_batch, increment iBatch
    Read point cloud from the file using last_used_index
    for each i in range(last_used_index+1, iBatch) do
        Read point cloud for current_point_cloud
        procedure GET_TRANSFORMATION_MATRIX(source,current_point_cloud)
            Define reg_p2p using registration_icp method
            return reg_p2p
        end procedure
        Extract transformation matrix from reg_p2p
        if transformation matrix is not identity then
            Print transformation info, update last_used_index and source
        else
            Print frame skipping info
        end if
    end for
    Extract transformation matrix T from reg_p2p
    Transform source using T to get target_pred
    Extract correspondence set from reg_p2p
    Compute error between target_pred_inliers and target_inliers
    Plot histogram of error_per_correspondence
    Update iEv to i * num_events_batch
    Create img, count arrays of zeros with img_size dimensions
    for each j in range(0, length of error_per_correspondence) do
        Update img and count arrays
    end for
    Compute average error per pixel in img
    Save img as png file
    Update T_rest by multiplying with T
    Append T_rest to T_new and corresponding x, y, z to pos
end while
```

Chapter 4

Results

In this section results, outputs and evaluations obtained through various methods are demonstrated. The main goal is to assess the robustness and accuracy of the algorithm for camera pose estimation leveraging the unique advantages of event cameras, which lead to low latency, high dynamic range, and robustness to varying levels of illumination, and depth cameras, for better scene understanding. First, the datasets used for evaluation are explained, followed by the calibration results of the sensors used and a detailed analysis of the algorithm's performance using several indicators: Root Mean Square Error (RMSE), Percentage of Inliers, Correspondence Error, Transformation Error, Heat Map Errors, Overlay of Point Clouds, Visual Inspection metrics, etc.

The Anaconda virtual environment which includes Python 3.9 and Open3D library version 0.16.0 has been used. The source code and results can be reached from the public GitHub Repository dedicated to this master thesis.

4.1 Egomotion Estimation

Egomotion estimation is the process of estimating camera self-motion. This estimation of self-motion can be handled via various methods. In Sec. 3.4, egomotion estimation via ICP has been explained in detail. The resultant motion estimation is extracted from a series of transformation matrices given by the output of ICP. In order to validate the accuracy of the resultant path obtained via chaining these transformation matrices, the estimated motion and ground truth should be registered (e.g., aligned). Alignment results can be investigated basically by plotting the ground truth camera trajectory in comparison with the estimated trajectory of the camera (3D motion). Another visual evaluation method is the projection of the point clouds to check if the location of the re-projected data agrees with the 2D image of events. The concept of back projection is explained in detail in Sec. 3.4.2. Those evaluation methods can be augmented through different aspects. However, to check how well the data and the estimated motion agree, besides visual inspection, there is a need for nu-

merical quantification. Throughout this section, several methods for evaluation of the studied egomotion estimation method are investigated.

- **Root Mean Square Error (RMSE)** is the most common figure of merit to evaluate the performance of egomotion estimation algorithms. On Tab. 4.3 results of different sequences from the Combined Dynamic Vision RGB-D Dataset and EVDfuse dataset are demonstrated. The Percentage of Inliers shown in Tab. 4.3 indicates the number of data points that were correctly matched, in scenarios where there are outliers in the data, such as noise. The RMSE computes the square root of the average of the squared distances between two sets of n corresponding points: O_i and P_i. These point sets are the target point cloud and its prediction, the transformation of the source point cloud (e.g., via an alignment transformation).

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} \|P_i - O_i\|^2}, \tag{4.1}$$

  If correspondences are not given, each point in the transformed source point cloud is matched to the target point cloud's closest point.

- **Correspondence Error** is the square root of correspondent points between source (predicted) and target (actual) point cloud and also shown in Tab. 4.3.

- **Visual Inspection** refers to the process of visually examining the results. Visual inspection can help identify obvious errors or inconsistencies that might be difficult to debug or might not be finely captured by quantitative metrics. Visual inspection consists visual representation of the estimated trajectory overlap on top of the ground truth trajectory.

- **Transformation Error** is the difference between the estimated transformation and ground truth transformation. To demonstrate Transformation Error, EVO mentioned in Sec. 4.2.3 has been employed. In the context of the EVO tool, the transformation error refers to the discrepancy between the estimated transformation (e.g., rotation and translation) and the ground truth transformation. This error can be quantified using various metrics, such as absolute pose error (APE) for evaluating the absolute trajectory accuracy, or relative pose error (RPE) for evaluating the local consistency of the trajectory.

- **Overlap of Point Clouds** is a visualization technique where the estimated point cloud (e.g., the reconstructed 3D map) is overlaid on the ground truth point cloud. This can help visually assess the accuracy of the reconstruction. Differences in the structure or alignment of the two point clouds can indicate errors in the estimation.

- **Heat-map Error** is a representation of the error of individual pixels according to ground truth. The data across snapshots could be aggregated

and thus a single histogram of error distribution for the whole dataset may be achieved.

4.2 Algorithm

The algorithm to obtain the ICP results are shown in Fig. 4.1. In more detail it is explained as follows:

1. Extract the information from the event-depth file.
2. Determine the number of events that will be used to produce snapshots, as explained in Sec. 3.2.1
3. Create event-depth batches and make them semi-dense depth maps, as in the example of Fig. 4.2.
4. Create a point cloud file (.ply) for those snapshots/scenes created in batches.
5. Feed point clouds to ICP algorithm
    - Load the Point Cloud Data: The first step is to load the source and target point clouds from the .ply files using the O3D library's function for point cloud reading. These point clouds contain the 3D coordinates of the points in the scene and are represented as numpy arrays.
    - Initialize the Transformation Matrix: The next step is to initialize the transformation matrix that will align the source point cloud with the target point cloud. The initial transformation matrix is usually set to the identity matrix, which means that the source point cloud is initially aligned with the target point cloud. Such an initialization does not work well if the batches of events are too close to each other. Hence, the snapshots were selected 10 batches apart, to provide with significant camera motion between snapshots.
    - Run the ICP Algorithm: The ICP algorithm is then run iteratively to find the transformation matrix that best aligns the source point cloud with the target point cloud. At each iteration, the algorithm computes the closest points between the source and target point clouds and uses them to update the transformation matrix. The algorithm continues to iterate until a convergence criterion is met (or the mentioned criteria of iteration number in Sec. 3.4.1), such as a maximum number of iterations or a minimum change in the transformation matrix.

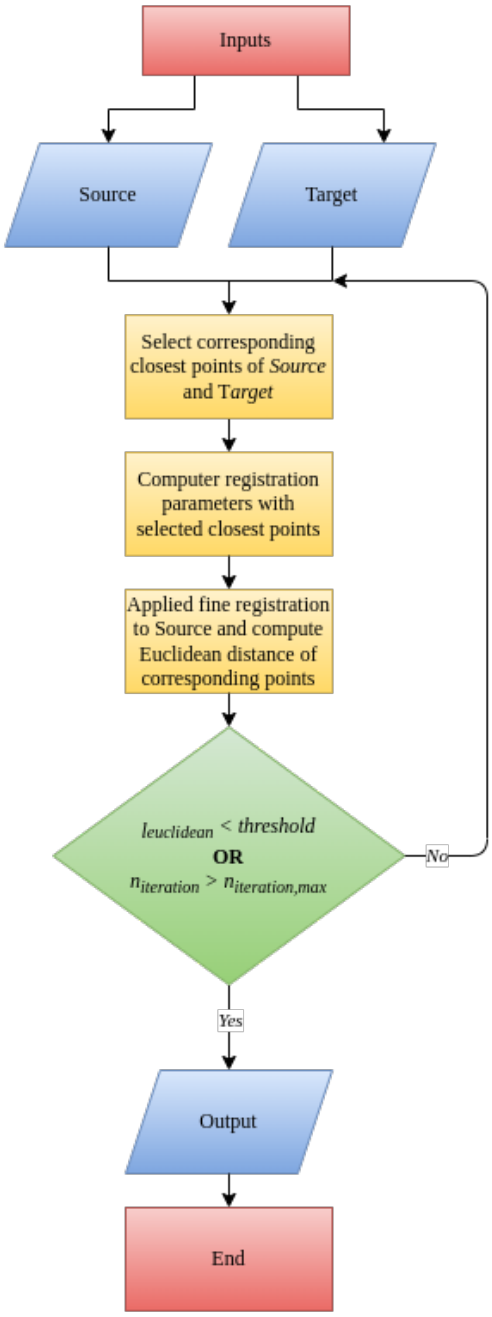

Figure 4.1: Flow chart of registration of point clouds.

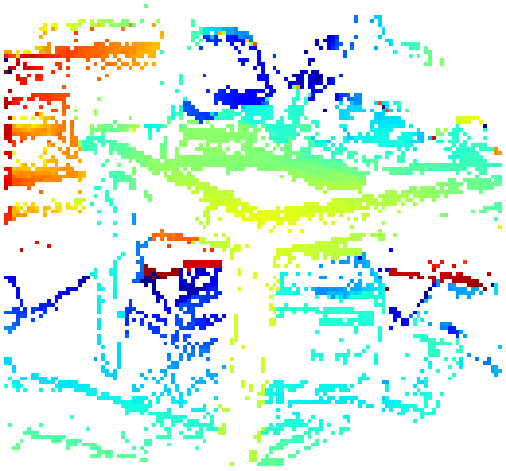

Figure 4.2: Semi-dense depth maps of TUM Combined.

6. Evaluate the Registration Quality: After the ICP algorithm has converged, it is important to evaluate the registration quality to determine if the alignment is acceptable. O3D provides several functions for this, which can be used to visualize the registered point clouds and compute various metrics of the aligned point clouds. Output metrics of O3D will be explained in more detail in Sec. 4.2.1.

7. Save the Aligned Point Cloud: Finally, if the registration quality is acceptable, the aligned point cloud can be saved to a new .ply file.

4.2.1 Output Metrics

Output Metrics are the direct results produced by a model or algorithm. In the context of an egomotion estimation problem solved with ICP, output metric is the estimated position and orientation of the 3D space, a *Transformation Matrix*.

When running the ICP algorithm in Open3D, there are several output metrics that can be computed to evaluate the quality of the registration between the source and target point clouds. Here are some of the commonly used metrics;

- **Fitness score**: This measures how well the source point cloud aligns with the target point cloud. It is computed as the ratio of the number of correspondences between the two point clouds to the total number of points in the source point cloud.

- **Inlier RMSE**: This measures the root-mean-square deviation between the source and target point clouds after the transformation has been ap-

plied. It is computed only for the correspondences that are considered to be "inliers" based on a given distance threshold.

- **Transformation matrix**: The transformation matrix that maps the source point cloud to the target point cloud can be output as a 4x4 homogeneous transformation matrix.
- **Correspondences**: The ICP algorithm can output the set of correspondences between the source and target point clouds, which can be useful for further analysis or visualization.
- **Convergence criteria**: The ICP algorithm can output the number of iterations taken to converge, the final change in the transformation matrix, and the final fitness score and inlier RMSE.

These output metrics can help assess the quality of the registration between the source and target point clouds and guide the selection of the parameters used in the ICP algorithm. For rapidly changing scenes, we have event output which is continuous in time, a less computationally complex model can be used since the snapshot itself and the scale of the scene will not be changed, because the snapshots will be created if there are events. Furthermore, if there is an event there is movement or changes in brightness, in short, a change in the scene that is worth to give attention to.

4.2.2 Evaluation Metrics

Evaluation Metrics are used to measure the performance of the model or algorithm. These metrics help in understanding how well the model or algorithm is working by comparing the predicted output metrics with the ground truth.

An example, the registration evaluation results of ICP are as follows; Registration result with fitness=0.949, inlier rmse=10.21622, and correspondence set the size of 2847. What do these figures mean?

Fitness, which measures the overlapping area (number of inlier correspondences/number of points in target). "The higher the better".

Inlier RMSE, which measures the RMSE of all inlier correspondences. "The lower the better".

Correspondence Set Size shows how many points could be aligned. "The higher the better".

The algorithmic parameters that can affect these output metrics are:

1. The Number of events used. Due to the nature of the event camera, there is no data in the form of images from a DVS. Yet from events, an image can be created. When creating images there is a decision between two

common choices: using a constant *number of events* or a constant *time interval duration*.

2. The Maximum correspondence distance. At first, the ICP algorithm matches the closest points and tries to match those as correctly as possible iteratively. This parameter is given in millimetres.

3. The Transformation initializer. The ICP algorithm takes this transformation matrix for initialization and tries to update it according to the registration results.

4. The Number of iterations until stopping or convergence.

As an example of alignment after 3000 iterations of a source-target pair, the transformation matrix looks like this:

$$\begin{bmatrix} 9.99754e^{-01} & 2.19242e^{-02} & 3.30448e^{-03} & 3.47626 \\ -2.18655e^{-02} & 9.99618e^{-01} & -1.68660e^{-02} & 3.86421e^{01} \\ -3.67300e^{-03} & 1.67896e^{-02} & 9.99852e^{-01} & -1.18105e^{02} \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.2}$$

How to tune these parameters for Egomotion Estimation?

ICP minimizes the difference between two point clouds by aligning their closest points. Two key parameters for ICP are the Maximum Correspondence Distance and the Transformation Initializer. ICP requires proper tuning of these parameters.

Maximum Correspondence Distance is the maximum distance between corresponding points in the source and target point clouds. Points with larger distances than a predetermined threshold will be ignored during the ICP process. Selection of this parameter can significantly affect the results. If there is no initial guess for the distance between the source and target point clouds' correspondent points, starting with a large value is beneficial. Based on the results, this initial guess of correspondence pair distance may be gradually decreased or increased. The correspondence pair distance parameter also depends on the scale of the application. A too small value may result in an insufficient number of point pairs for alignment, while too large a value may lead to incorrect matches and a poor final alignment.

Transformation Initializer helps to provide an initial guess of the transformation. The transformation initializer needs to be set based on a guess (or prior knowledge) of the initial relative position of the point clouds. In the absence of an initial guess, an identity matrix (I_{4x4}) can be used. Another strategy is to use a coarse-to-fine approach: first, perform a rough alignment using another method (like a feature-based alignment), then refine the transformation using ICP.

Algorithm parameter adjustment requires iterative refinement. Starting with a good guess, examining the outcomes, and then changing the parameters. It is also important to be mindful of the computational implications of these choices.

More iterations and smaller distances can increase the computational load of the algorithm.

In scope of this thesis, an identity matrix, I_{4x4}, is used as an initializer and various correspondence distances are chosen to see the effect on the estimated egomotion.

4.2.3 Evaluation with EVO

EVO [70] has been used to evaluate the egomotion estimation results in the context of transformation error. It is a Python package compatible with ROS designed for the evaluation of odometry and SLAM. The package provides a small library for evaluating and comparing the trajectory output of odometry algorithms. The tool is available on GitHub and is widely used in the robotics community due to its versatility and robustness. It supports various trajectory formats including TUM [71], KITTI [72], EuRoC MAV[73], and ROS bag files with `geometry_msgs/PoseStamped` or `nav_msgs/Odometry` topics.

The following executables can be called globally from the command line to use evo; `evo_ape` (for absolute pose error), `evo_rpe` (for relative pose error), `evo_traj` (for analyzing, plotting or exporting one or more trajectories) and `evo_config` (for global settings and config file manipulation).

4.3 The Combined Dynamic Vision / RGB-D Dataset

The Combined Dynamic Vision/RGB-D Dataset [42], which will be called "TUM Combined" from now on, is used to evaluate egomotion estimation using ICP by fusing event and depth data. This dataset is advantageous for the task due to its unique combination of RGB, depth, and event data acquired by an event camera as mentioned in Sec. 2.2. Notably, the dataset contains real-world trajectory data, permitting quantitatively evaluate the proposed method.

The dataset includes several files and folders:

- events.tsv: A .tsv file with one line (e.g., row) per event. The provided data is: PrimeSense image coordinates (x,y), PrimeSense depth measurement, EDVS image coordinates (x,y), Timestamp (in µsecs), eDVS polarity flag.
- events-pe.tsv: A .tsv file where each line corresponds to a 3D point event. The provided data is: Timestamp (µsecs), 3D point in camera coordinates (x,y,z).
- gt.bvh: Ground truth data from an external tracking system in the .bhv motion tracking file format.
- depth (folder): Folder with depth frames. Each frame is saved as a file. Frames are stored as binary dumps in row-major format. For each pixel, a 16-bit integer is stored indicating the pixel depth.

- rgb (folder): Folder with color frames. Each frame is saved as one file. Frames are stored as binary dumps in row-major format. For each pixel, three 8-bit integers are stored indicating the red, green and blue color value for the pixel.

$x_{\text{PrimeSense}}$	$y_{\text{PrimeSense}}$	$depth$	x_{EDVS}	y_{EDVS}	t	p
369	388	1776	76	103	8409	1
368	249	1543	78	66	8409	0
163	249	1713	21	64	8409	0
283	255	1639	53	67	10295	1
248	435	1529	44	114	12386	1
339	264	1536	69	70	18303	1
148	275	3692	18	70	21710	1
314	360	1564	62	96	21710	1
198	301	1370	30	79	21710	0
277	241	1647	52	63	21710	1

Table 4.1: events.tsv

Since depth values are specified in (milli)meters and x, y values are in pixels, there needs to be a conversion between them using the intrinsic parameter matrix:

$$\begin{pmatrix} x_{\text{pix}} \\ y_{\text{pix}} \\ 1 \end{pmatrix} = \mathtt{K} \begin{pmatrix} x_{\text{meters}} \\ y_{\text{meters}} \\ 1 \end{pmatrix}, \tag{4.3}$$

where $\mathtt{K}$ is the 3×3 matrix

$$\mathtt{K} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 128 & 0 & 64 \\ 0 & 128 & 64 \\ 0 & 0 & 1 \end{pmatrix}. \tag{4.4}$$

4.4 EVDfuse Dataset

EVDfuse is a dataset that has been recorded at TU Berlin Faculty IV, in the Motion Capture room of the Science of Intelligence Excellence Cluster. It contains four sequences later divided into different sub-sequences with various lengths. The scene consisted of cylinders located at various distances from the camera and a checkerboard, as can be seen in Fig. 4.3. The full dataset can be reached from public GitHub Repository of EVDfuse. In Tab. 4.2, Ev implies the number of events, D represents the number of depth maps included, GT is how many timestamps of ground truth values exist, and $Num.Est.$ is the number of transformations between consecutive frames which is related to the number of events chosen to create a frame in this sequence.

Figure 4.3: MoCap room of TU Berlin's Science of Intelligence Excellence Cluster, Faculty IV, while recording the EVDfuse Dataset.

To acquire depth maps there is a need for frames from different perspectives of the scene. In Fig. 4.5, Intel RealSense stereo vision records in different perspectives and calculates the difference between these two images. After disparity calculations, depth is achieved.

Scene	Take	Duration	Ev	D	GT	Num. Est.
6D	0	6.2 s	1233294	76	746	164
6D	1	7.0 s	3052253	152	840	406
6D	2	7.0 s	2734487	157	840	364
6D	3	7.0 s	3232933	149	840	164
6D	4	7.0 s	3417448	140	840	455
6D	5	7.0 s	2486781	170	840	331
6D	6	7.0 s	2335588	160	840	311
6D	7	7.0 s	3421084	158	840	456
6D	8	7.0 s	3969835	109	840	529
6D	9	7.2 s	4496125	152	868	364

Table 4.2: EVDfuse scene6D data statistics.

In Fig. 4.4 one of the semi-dense depth maps created with $N_e = 7500$ from the EVDfuse dataset, scene6/take00 is shown. In Fig. 4.6 path of motion is represented by order from 1-9 of EVDfuse dataset scene 6D.

4.4.1 Calibration Results

Built-in topics published via ROS made possible the knowledge of the calibration between Depth and EV frames, as illustrated in Fig. 4.7. For Intel RealSense D435 there are two cameras to detect intrinsics; RGB and Depth.

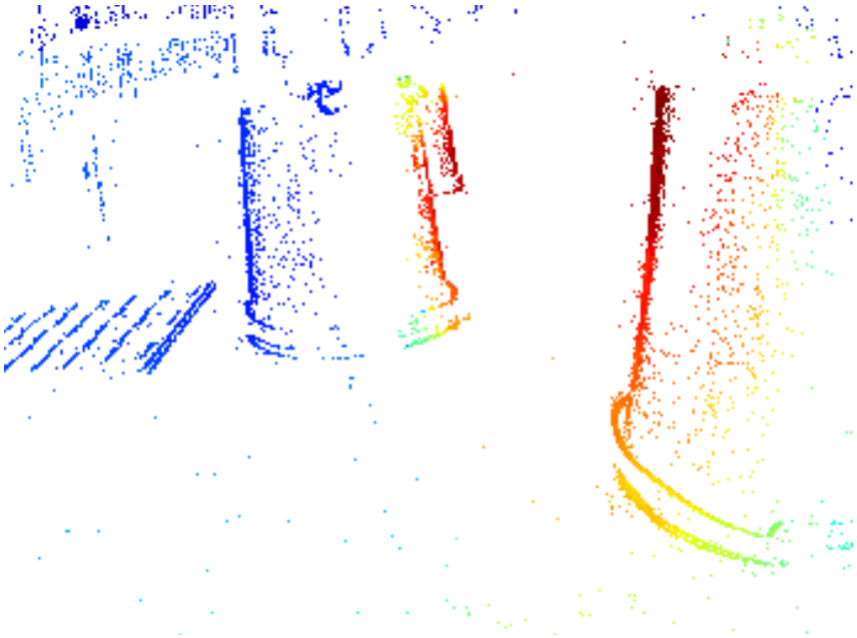

Figure 4.4: Semi-dense depth map created by EVDfuse.

- **RGB Camera Intrinsics:**
  - K (Camera Matrix): $\begin{bmatrix} 614.24780 & 0 & 326.02340 \\ 0 & 614.41760 & 238.85572 \\ 0 & 0 & 1 \end{bmatrix}$
  - R (Rotation Matrix): $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
  - P (Projection Matrix): $\begin{bmatrix} 614.24780 & 0 & 326.02340 & 0 \\ 0 & 614.41760 & 238.85572 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
- **Depth Camera Intrinsics:**
  - K (Camera Matrix): $\begin{bmatrix} 419.17794 & 0 & 427.87561 \\ 0 & 419.17794 & 241.68835 \\ 0 & 0 & 1 \end{bmatrix}$
  - R (Rotation Matrix): $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

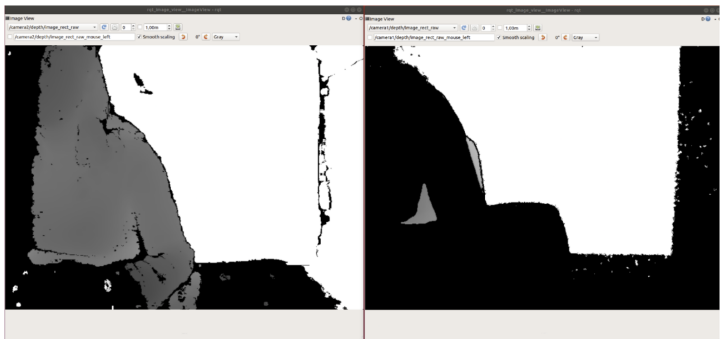

Figure 4.5: Intel RealSense D435 right and left imager outputs.

$$- \text{ P (Projection Matrix: } \begin{bmatrix} 419.17794 & 0 & 427.87561 & 0 \\ 0 & 419.17794 & 241.68835 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$^{\text{DVS}}T_{\text{Color}} = \begin{bmatrix} 0.9999 & 0.0041 & 0.01252 & 0.03356 \\ -0.0043 & 0.9998 & 0.01817 & 0.03458 \\ 0.01244 & -0.01822 & 0.9997 & 0.03485 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.5}$$

The transformation matrix from Depth to Color frame, acquired via Kalibr [61] toolbox, is as follows:

$$^{\text{Depth}}T_{\text{Color}} = \begin{bmatrix} 0.9999 & 0.0014 & -0.0066 & 0.0147 \\ -0.0015 & 0.9999 & -0.0017 & 0.0002 \\ 0.0066 & 0.0017 & 0.9999 & 0.0004 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.6}$$

To derive the transformation between Depth and Event camera frame, Eqns. 3.7 should be followed. The results of deriving the equation with Eqn. 4.6 are:

$$^{\text{DVS}}T_{\text{Depth}} = \begin{bmatrix} 0.999 & 0.002 & 0.019 & 0.0188 \\ -0.003 & 0.997 & 0.019 & 0.034 \\ 0.005 & -0.019 & 0.999 & 0.0347 \\ 0. & 0. & 0. & 1. \end{bmatrix} \tag{4.7}$$

To briefly explain, it is important that the transformation matrices are particularly accurate in the case that lenses and attachment of the cameras to each other are same. Even minimal changes would yield inaccuracies on the next steps that will be followed, such as AoV calculation, depth estimation, motion estimation, mapping, etc. Hence, sensor calibration is a critical first step when multiple sensors are integrated into a system for any application.

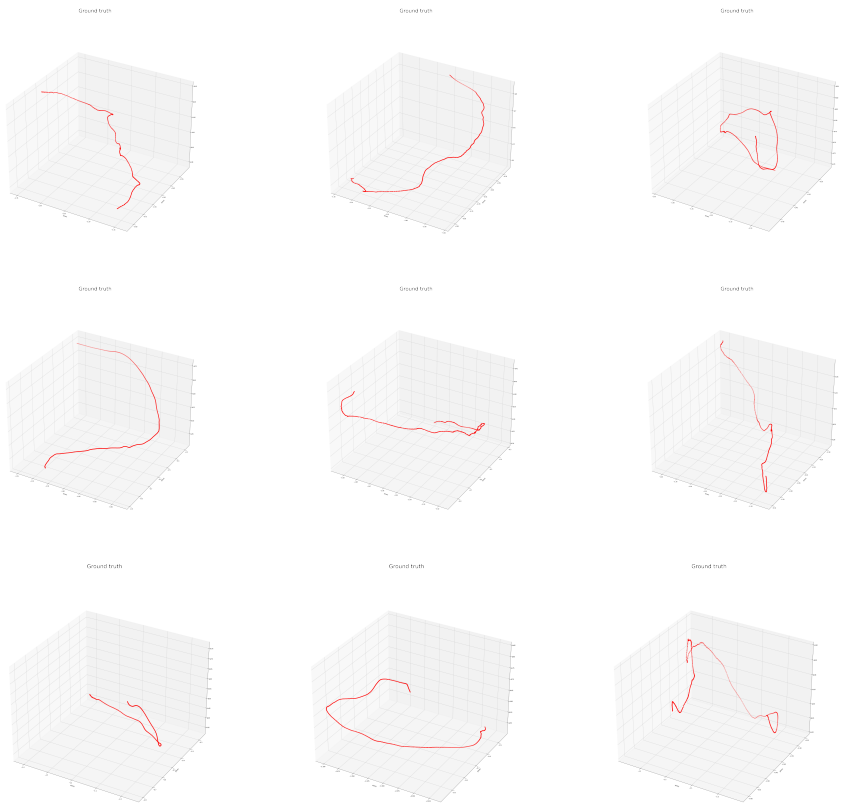

Figure 4.6: Path of motion EVDfuse dataset scene 6D. From upper left corner(1) to below right corner(9)

Steps for recording data and calibration

1. Run cameras, other sensor if there are any.
2. See topics published.
3. `rosbag record -a` (record all) or specific topics.
4. Begin calibration, `rosrun kalibr kalibr_calibrate_cameras`

The following parameters are used for Kalibr:

- `--bag filename.bag` - ROS bag containing the data.
- `--topics TOPIC_0 ... TOPIC_N` - list of all camera topics in the bag.
- `--models MODEL_0 ... MODEL_N` - list of camera/distortion models to be fitted, matches the ordering of `--topics`

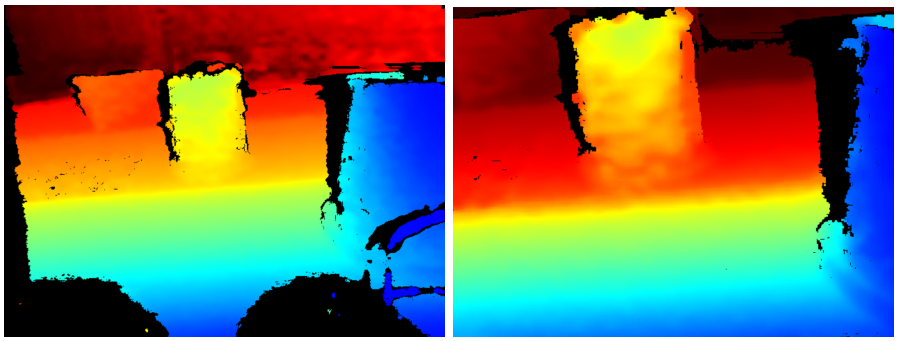

Figure 4.7: Left: Depth map before transformation, Right: Depth map after alignment of frames Depth with RGB frame.

- `--target target.yaml` - the calibration target configuration (see Calibration targets).

Using April Grids for calibration is recommended [74]

How to comment on calibration results

The output of Kalibr and many other calibration tools is a summary of the calibration results for the cameras considered along with the target configuration that was used for calibration. To be able to examine each output parameter following terminology must be explained:

- Camera-system parameters: (The values following "±" represent the uncertainty in these estimates.)
    1. distortion: These are the distortion coefficients for the camera. The distortion model used here is 'rad-tan' (radial and tangential).
    2. projection: These are the intrinsic parameters of the camera, including the focal lengths (f_x, f_y) and the optical centre or principal point (c_x, c_y).
    3. reprojection error: This is a measure of the discrepancy between the observed feature points and the predicted feature points given the estimated camera parameters. These errors are a common way to quantify the accuracy of camera calibration.
- baseline: This represents the transformation between the two cameras.
    1. q: This is the quaternion representation of the rotation between the two cameras and avoids certain issues that can arise with other representations like Euler angles.
    2. t: This is the translation vector between the two cameras.

- Target configuration: This provides information about the calibration target that was used. In this case, it was an April Grid, which is a type of fiducial marker that's often used in camera calibration. The April Grid [74] has 4 rows and 5 columns of tags, each of which is 0.051 meters in size, with a spacing of 0.2 meters between tags.

The calibration results for the `2022-10-27-16-13-21.bag` rosbag are the following. When running the Multiple Camera Calibration executable, 91 images were automatically selected by Kalibr.

- DAVIS346 as cam0, with (/dvs/image_raw) topic:
  - distortion:
    $\begin{bmatrix} -0.35206692 & 0.16161456 & -0.00323881 & 0.00138018 \end{bmatrix}$
    $\pm \begin{bmatrix} 0.00576598 & 0.01315156 & 0.00049594 & 0.00032174 \end{bmatrix}$
  - projection:
    $\begin{bmatrix} 341.55907485 & 342.68427818 & 178.62601943 & 147.90003194 \end{bmatrix}$
    $\pm \begin{bmatrix} 0.42975634 & 0.47111872 & 1.10915735 & 0.83801306 \end{bmatrix}$
  - reprojection error (in pixels):
    $\begin{bmatrix} -0.000210 & 0.000007 \end{bmatrix} \pm \begin{bmatrix} 0.371268 & 0.268896 \end{bmatrix}$
- Intel RealSense D435 as cam1, with (/camera/color/image_raw) topic:
  - distortion:
    $\begin{bmatrix} 0.12560804 & -0.28144178 & 0.00015631 & -0.00047208 \end{bmatrix}$
    $\pm \begin{bmatrix} 0.00501953 & 0.01436008 & 0.00060931 & 0.00069523 \end{bmatrix}$
  - projection:
    $\begin{bmatrix} 606.29976939 & 608.4095701 & 323.58776936 & 242.59985758 \end{bmatrix}$
    $\pm \begin{bmatrix} [0.3835958 & 0.54352318 & 1.12631367 & 0.99809569] \end{bmatrix}$
  - reprojection error (in pixels):
    $\begin{bmatrix} 0.000071 & -0.000003 \end{bmatrix} \pm \begin{bmatrix} 0.469412 & 0.335000 \end{bmatrix}$
- baseline:
  - q:
    $\begin{bmatrix} [0.00910142 & -0.00624109 & 0.00213327 & 0.99993683] \end{bmatrix}$
    $\pm \begin{bmatrix} [0.00245888 & 0.00330849 & 0.00022623] \end{bmatrix}$

- t (in meters):
  $\begin{bmatrix}0.03356659 & 0.03458064 & 0.03485542\end{bmatrix}$

  $\pm \begin{bmatrix}0.00014253 & 0.00014475 & 0.000445\end{bmatrix}$

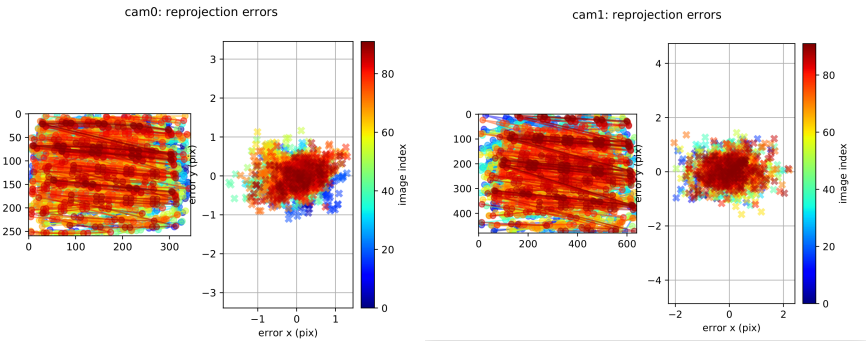

Figure 4.8: An example of calibration results of Kalibr: DAVIS364 (left), Intel RealSense D435 (right).

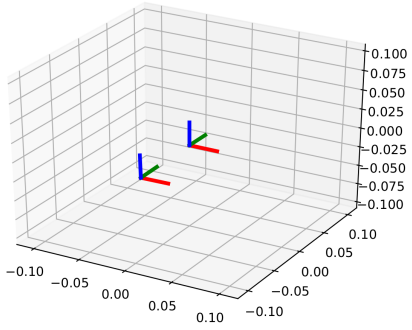

Figure 4.9: Camera positions determined by Kalibr.

Further calibration was conducted on DAVIS APS images only, automatically selecting 44 images out of approximately 2800. The results show a significant decrease in the standard deviation of the reprojection error from 0.46 pixels to 0.11 pixels, indicating a marked improvement in the calibration. The results are as follows.

- DAVIS346 as cam0, with (/dvs/image_raw) topic:
  - distortion:
    $\begin{bmatrix}-0.35406162 & 0.16186196 & 0.00047246 & 0.00137192\end{bmatrix}$

± [0.00949994 0.01929236 0.00187364 0.00053345]

- projection:
  [344.99602244 345.20546997 174.06296325 143.72289745]

  ± [2.67557617 2.29978359 1.88410216 2.53304563]

- reprojection error (in pixels):
  [0.000001 −0.000000] ± [0.110750 0.127638]

When Kalibr was run only on the color camera, 46 images were used out of approximately 2100. Despite this, the standard deviation of the reprojection error remained unchanged, as follows.

- Intel RealSense D435 as cam0, with (/camera/color/image_raw) topic:
  - distortion:
    [0.13792232 −0.32325063 0.00075118 0.00074803]

    ± [0.00662108 0.01868444 0.00077972 0.0008301]

  - projection:
    [610.32698461 610.88769497 323.94211514 239.51787207]

    ± [2.98063061 2.70151172 1.41285088 2.35045653]

  - reprojection error (in pixels):
    [0.000012 −0.000007] ± [0.442280 0.312437]

Looking at the results, the calibration appears to have been performed successfully. The reprojection errors are quite small, which typically indicates a good calibration. The uncertainties in the estimates are also relatively small. In the calibration process of the `2022-10-27-16-13-21.bag` rosbag, Kalibr was run on both topics, resulting in the utilization of 91 images. The camera parameters for each camera, including distortion, projection, and reprojection error, were documented. Notably, the reprojection error was markedly different for both cameras, with the first camera demonstrating less deviation.

The calibration results for the individual cameras and their combined use suggest that the calibration procedure has a major impact on the output quality of the scene projection. Individual camera calibration resulted in an improvement for the DAVIS346 camera, but the overall system failed to indicate this improvement. This result highlights the significance of whole-system calibration as opposed to a singular focus on individual components. Future calibration arrangements should aim to include more comprehensive measures to guarantee the optimal operation of the entire system.

4.5 Experimental Results on TUM Combined and EVDfuse

For the evaluation of the motion estimation algorithm, the Combined Dynamic Vision / RGB-D Dataset [42] has been used. Events and depth taken from the dataset are saved as a ".ply" file format to be able to work on 3D geometry. In the following, the results produced by using sequence `scene2_take01` are presented.

4.6 Visual Inspection

In this subsection, the results obtained from the egomotion estimation process are visually inspected. Graphical plots are used to illustrate the trajectories generated from the output, transformation matrix, by the ICP algorithm. These visual results are compared with ground truth data, examples are shown in Fig. 4.10, providing a qualitative assessment of the performance of the ICP with chosen parameters and setup in estimating egomotion.

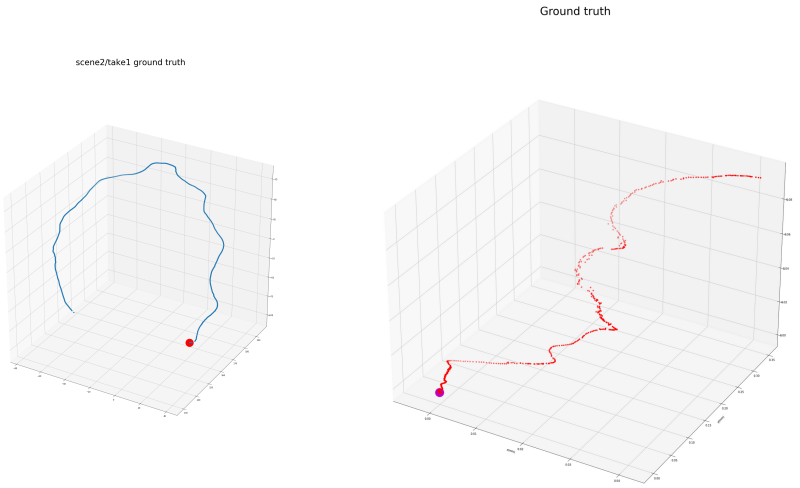

Figure 4.10: Left: Visualization of scene2/take01 ground truth, Right: Visualization of scene6D/take00 ground truth.

The translation error can be plotted via EVO easily, as mentioned in Sec. 4.2.3; see Figs. 4.11 and 4.12.

Fig. 4.13 shows that the information about trajectories can be extracted. Ground truth and estimation results seem to be on different scales. Since the estimated data has been interpolated by not changing the start and end points, the number of poses of both estimation and ground truth data are the same. The number of poses to be equal is crucial for a better comparison between the estimation and ground truth. The path (camera trajectory) lengths are

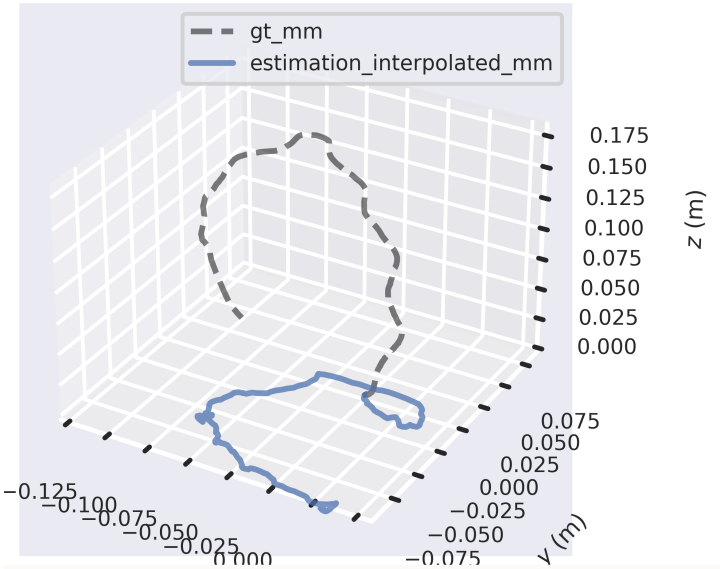

Figure 4.11: Estimation and Ground Truth Plot TUM Combined.

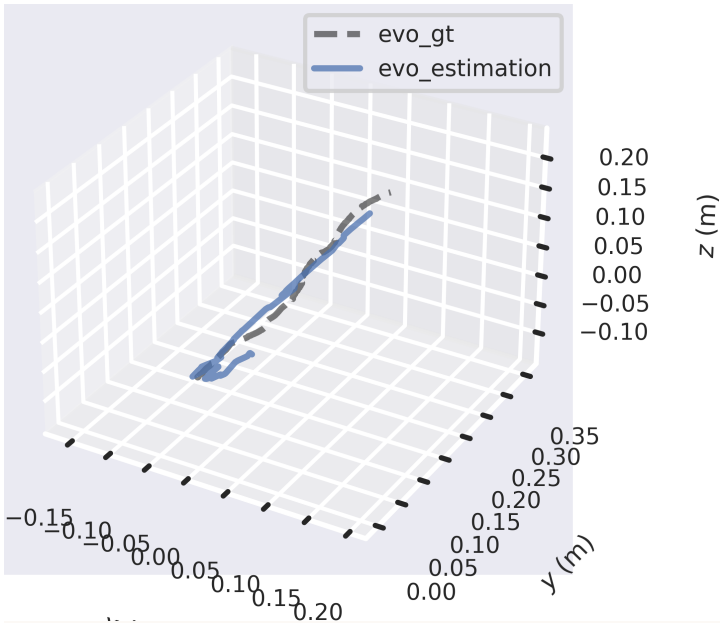

Figure 4.12: Estimation and Ground Truth Plot EVDfuse.

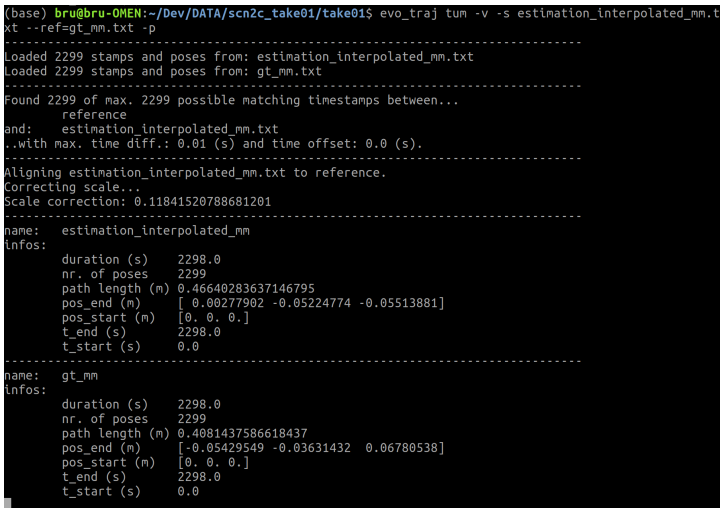

Figure 4.13: Execution of `evo_traj`.

different, with a path length of estimation equal to 0.466 m and a ground truth path length equal to 0.408 m. Estimation is more fluctuating compared to ground truth. The starting point(s) is at the world origin, $(0, 0, 0)$ coordinates, yet the ending points are different as there is a difference in the world coordinates of estimation and ground truth data.

4.7 Point Cloud Matching

Here, the process and results of matching point clouds are discussed. Algorithms like ICP may be used to align and match two point clouds. The quality of the match can be evaluated using metrics like correspondence error, fitness score, and inlier RMSE.

ICP implemented on Combined Dynamic Vision/RGB-D Dataset gives the output of Fig. 4.14 for first alignment results. ICP implemented on EVDfuse Dataset gives the output of Fig. 4.15 after 400 iterations with a 1 mm correspondence distance.

Figure 4.14: Point cloud registration results from TUM Combined scene2/take01, Left: Before alignment, Right: After alignment.

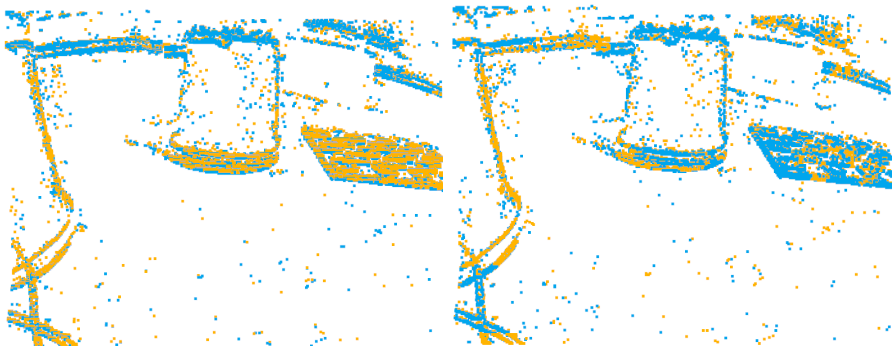

Figure 4.15: Point cloud registration results from EVDfuse scene1/take00, Left: Before alignment, Right: After alignment.

4.8 Correspondence Error, Fitness Score and Inlier RMSE

This subsection delves into the quantitative evaluation of the implemented method, ICP. The correspondence error, which measures the average distance between corresponding points in two point clouds, is used to indicate the alignment of the point clouds. The fitness score, measuring the overlap between two point clouds, is used to indicate the quality of the match. The Inlier RMSE, a measure of the average error in the inlier points after a model fitting process, is also calculated. Fig. 4.16 shows evaluation graphs over consecutive point clouds that are created from the TUM Combined dataset. The number of correspondences over batches and fitness scores over batches have the same curves due to the fact that fitness over batches is calculated as, (number of inlier cor-

respondences/number of points in target). In Fig. 4.17 evaluation graphs over consecutive point clouds that are created from the EVDfuse dataset can be seen. To compare these two results, despite EVDfuse point clouds being created with 7500 points and TUM Combined created with 5000 points, EVDfuse gives better results with ICP parameters, 40 mm correspondence distance, initializing matrix identity, and the number of iterations 100.

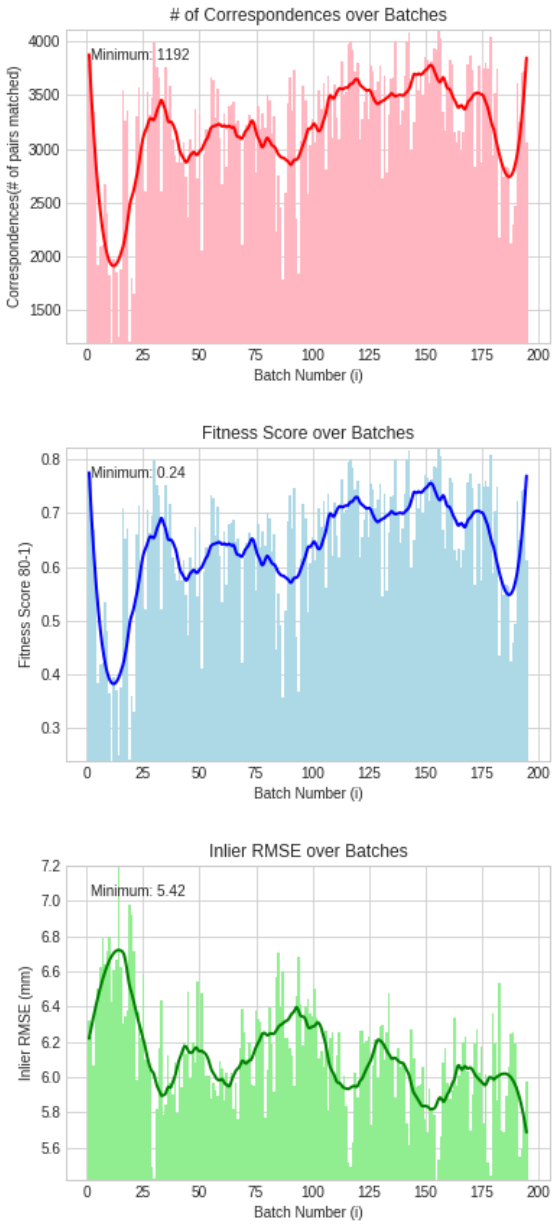

Figure 4.16: Left: Number of Correspondences, Middle: Fitness Score, Right: Inlier RMSE. From TUM Combined scene 2 take 01.

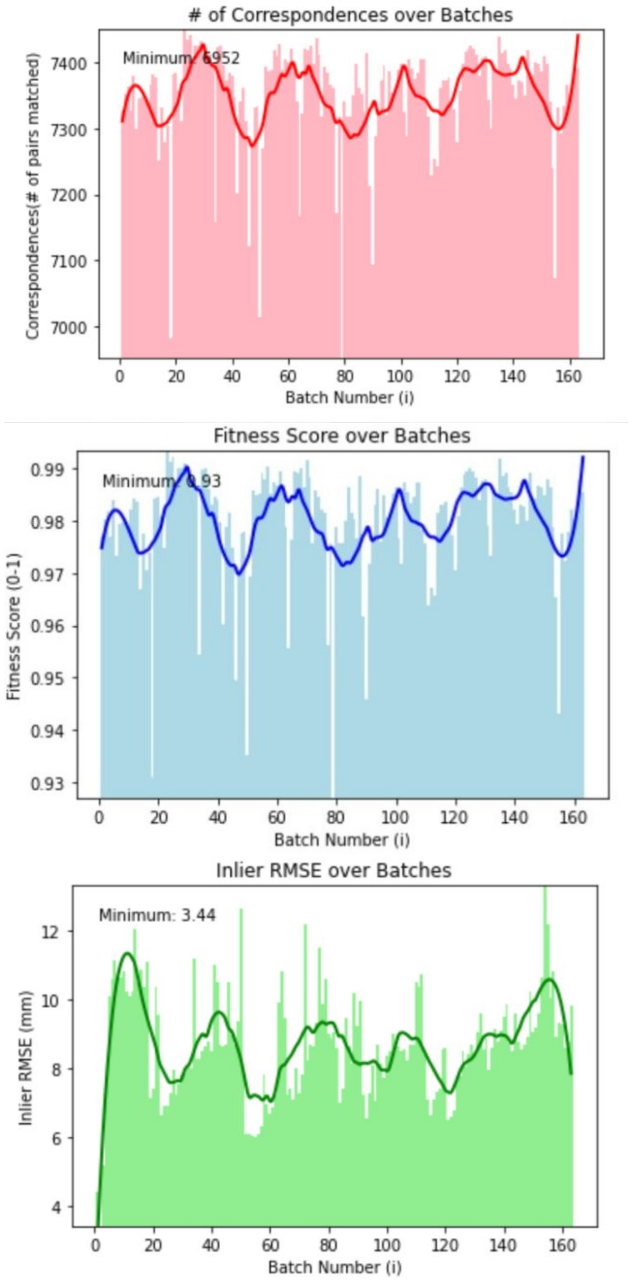

Figure 4.17: Left: Number of Correspondences, Middle: Fitness Score, Right: Inlier RMSE. From EVDfuse, scene 6D take 0.

4.9 Absolute Pose Error & Relative Pose Error

The Absolute Pose Error (APE) and Relative Pose Error (RPE) are utilized to evaluate the accuracy of the odometry algorithms. APE measures the discrepancy between the estimated pose and the actual pose at the same timestamp, while RPE measures the discrepancy in the movement between two poses. These metrics provide a measure of how accurately the pose (position and orientation) of the sensor over time is estimated by the implemented method.

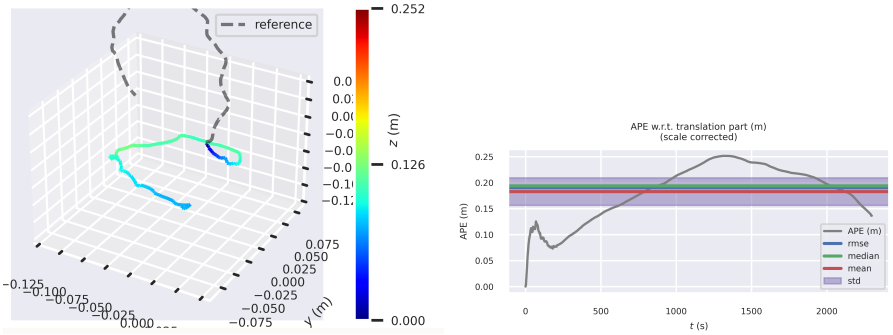

Figure 4.18: Estimation and Ground Truth Absolute Pose Error of TUM Combined scene2/take01

In Fig. 4.18 the APE error graph shows a reasonable slope since the world planes of estimation and ground truth of TUM Combined do not match.

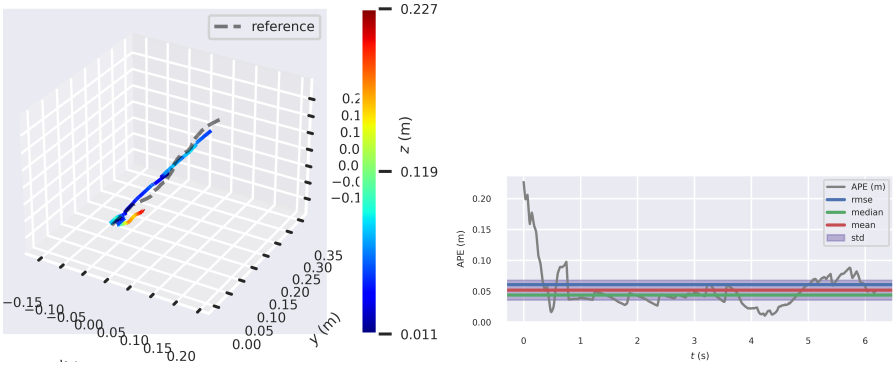

Figure 4.19: Estimation and Ground Truth Absolute Pose Error of EVDfuse scene1/take0.

In Fig. 4.19 EVDfuse dataset with ICP shows an unsuccessful matching at first. Rotations were rapid at first but after some frames the ICP stands robust. The graph of APE is fluctuating but again, after some frames the standard deviation of the error becomes small.

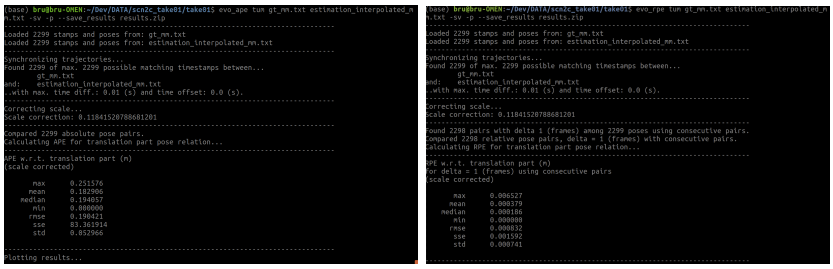

Figure 4.20: Left: Execution of `evo_ape`, Right:Execution of `evo_rpe`

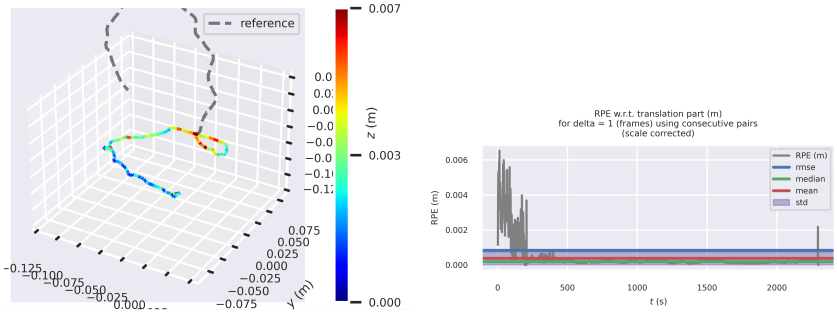

Figure 4.21: Estimation and Ground Truth Relative Pose Error Graph of TUM Combined scene2/take01.

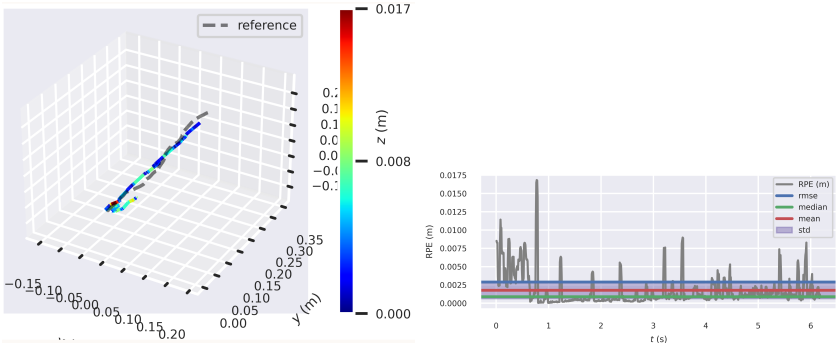

Figure 4.22: Estimation and Ground Truth Relative Pose Error Graph of EVD-fuse scene1/take00.

When executing `evo_traj`, `evo_ape` and `evo_rpe` there are several parameters that can be given to the tool in cases of scale mismatching, etc., as shown in Fig. 4.20.

4.10 Heat-Map Error

Heat maps are used to visualize the error in the estimated egomotion. Each point on the heat map represents an event/pixel, with the color indicating the magnitude of the error at that pixel. This provides a visual way to identify where the algorithm performs well and where it encounters difficulties.

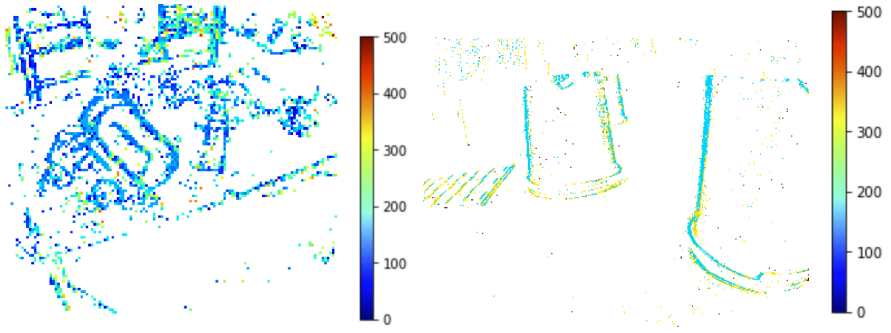

Figure 4.23: Left: Heatmap of errors TUM Combined scene2/take01, Right: Heatmap of errors EVDfuse scene1/take00.

4.11 Comparison of Results & Discussion

The methods mentioned in Sec. 4.1 and their results are shown throughout this chapter. Those metrics are commonly used to evaluate the performance of egomotion estimation algorithms, such as those used in SLAM or visual odometry. Each method and metric provides a different perspective on the performance. Those parameters together can provide a comprehensive evaluation of the algorithm's accuracy and consistency.

To explain the Tab. 4.3, d_{max} is the maximum correspondence distance, i_{max} is the maximum number of iterations per each consecutive point cloud, T_{init} is the initializer and last 3 columns are evaluation results in the order of, Root Mean Square Error, Average Fitness Score and Average Number of Correspondences. All sequences of EVDfuse dataset has been used for egomotion estimation.

In the table, the most notable results are shown. From all of the evaluations carried out, it can be clearly seen that our own dataset gives better results while different combinations of parameters have been investigated. As discussed before, parameters, d_{max}, i_{max} and T_{init} should be chosen according to the application.

ICP parameters in Tab. 4.3 last 10 columns, "EVDfuse all sequences" were $d_{max} = 40$ mm, $T_{init} = T_{(0.1)}$ and $i_{max} = 400$.

$$\text{where, } I_{(0.1)} = \begin{pmatrix} 1 & 0 & 0 & 0.1 \\ 0 & 1 & 0 & 0.1 \\ 0 & 0 & 1 & 0.1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Indoor, fast changing scenes gives better results in mm scale. Increasing the correspondence distance in mm scale would yield better results according to speed of the changing scene. Thanks to event cameras continuous nature, initializing with an identity matrix yield good results. Increasing the number of iterations was not efficient moreover, unnecessary. Increasing the number of iterations between consecutive point clouds did not effect the results much, the reason is, the scenes are already changing fast and this yield creating frames from events which yield nearly the same point clouds.

	Trial	Scn&Take	T_init	$d_{\max}$	$i_{\max}$	RMSE (mm)	Avg. Fitness Score	Avg. # Correspondences
TU Munich	1	2_1	identity	10	10	6.1191	0.638	216/5000
	2	2_1	identity	20	10	8.6243	0.917	221/5000
	3	2_1	identity	30	10	9.8225	0.96	241/5000
	4	2_1	identity	10	10	6.0332	0.66	161/5000
	5	1_1	identity	20	10	8.5965	0.91	273/5000
	6	1_1	identity	30	10	9.8473	0.96	313/5000
EVDfuse	7	6D_0	identity	30	10	7.7720	0.96	119/7500
	8	6D_0	identity	30	100	7.6394	0.96	115/7500
	9	6D_0	identity	10	10	4.8382	0.81	63/7500
	10	6D_0	identity	10	300	4.8382	0.81	64/7500
	11	6D_0	identity	40	10	8.6832	0.98	136/7500
EVDfuse all sequences	12	6D_0	$T_{(0.1)}$	1	400	8.6849	0.980	135/7500
	13	6D_0	$T_{(0.1)}$	1	400	8.6782	0.982	160/7500
	14	6D_0	$T_{(0.1)}$	1	400	8.8756	0.977	156/7500
	15	6D_0	$T_{(0.1)}$	1	400	8.6849	0.980	135/7500
	16	6D_0	$T_{(0.1)}$	1	400	8.9949	0.987	156/7500
	17	6D_0	$T_{(0.1)}$	1	400	8.8741	0.974	162/7500
	18	6D_0	$T_{(0.1)}$	1	400	7.5231	0.983	142/7500
	19	6D_0	$T_{(0.1)}$	1	400	8.7734	0.981	162/7500
	20	6D_0	$T_{(0.1)}$	1	400	9.4976	0.975	177/7500
	21	6D_0	$T_{(0.1)}$	1	400	8.8756	0.977	156/7500

Table 4.3: Results on EVDfuse and TUM Combined datasets.

Chapter 5

Conclusion and Outlook

This study was primarily focused on enhancing the egomotion estimation algorithm by fusing events and depth. Throughout Ch. 4, several validation methods to interpret the results have been analyzed. From these several evaluation methods, RMSE, Correspondence Error, and Transformation Error have united this study on a common basis of egomotion estimation research. While Point Cloud Overlap, Back Projected Point Clouds, and Heat-map Errors gave a visual ease of understanding, the Time Series of Mean Errors showed the time distribution of the errors. The effectiveness and efficiency of the preferred method of Iterative Closest Points for egomotion estimation were evaluated using these different approaches. The outlook for this area of research is compelling and rich with potential. An appreciable contribution to the domains of Autonomous Vehicles, Augmented Reality/Virtual Reality, and Real-time applications can be provided by this study.

In the context of this study an event camera and a depth camera were employed. In order to reduce the cost of the configuration of the proposed system, depth estimation from a monocular event camera could be employed. Monocular depth estimation might be used but it may also increase the computational effort. Future studies could aim to strike a balance between methodological complexity and computation time. While the potential for increased accuracy was demonstrated by a raised complexity of the estimation methods, a need for optimizing computational resources also emerged. In order to balance computational complexity and cost, the depth camera was the most efficient selection among other sensor infrastructures. It would be beneficial for future studies to explore this balance further, possibly through the development or adaptation of more efficient algorithms.

As a future suggestion, Gallego et al. [75] discuss a unifying framework for addressing multiple challenges in computer vision, such as optical flow estimation, depth estimation, rotational motion estimation, and motion estimation in planar scenes by using event cameras. Even though 6-DOF egomotion es-

timation was not explicitly demonstrated in the article since it had difficulties converging, the other tasks described (optical flow estimation, depth estimation, and rotational motion estimation) are closely related and could be revisited to enable joint depth and egomotion estimation. The proposed method, Contrast Maximization (CMax), has the potential to contribute to egomotion estimation for arbitrary scenes. Thus, it may lead to future studies in the area of localization.

As a last word, it is anticipated that the outcomes of this study will spark further research in the field, thereby advancing the understanding and application of using event information, fusing events with other sensors, carrying out robot localization, mapping, and control operations through perception (e.g., computer vision) techniques.

Bibliography

[1] M. O. Aqel, M. H. Marhaban, M. I. Saripan, and N. B. Ismail, "Review of visual odometry: types, approaches, challenges, and applications," *SpringerPlus*, vol. 5, no. 1, pp. 1–26, 2016.

[2] S. Poddar, R. Kottath, and V. Karar, "Evolution of visual odometry techniques," *arXiv preprint arXiv:1804.11142*, 2018.

[3] G. Gallego, T. Delbruck, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. Davison, J. Conradt, K. Daniilidis, and D. Scaramuzza, "Event-based vision: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 1, pp. 154–180, 2022.

[4] J. Muybridge, "The horse in motion," *Nature*, vol. 25, no. 652, pp. 605–605, 1882.

[5] C. Posch, T. Serrano-Gotarredona, B. Linares-Barranco, and T. Delbruck, "Retinomorphic event-based vision sensors: Bioinspired cameras with spiking output," *Proc. IEEE*, vol. 102, pp. 1470–1484, Oct. 2014.

[6] M. Mahowald, "The silicon retina," in *An Analog VLSI System for Stereoscopic Vision*, pp. 4–65, Springer, 1994.

[7] G. Gallego, T. Delbrück, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. J. Davison, J. Conradt, K. Daniilidis, *et al.*, "Event-based vision: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 1, pp. 154–180, 2020.

[8] C. Mei and P. Rives, "Single view point omnidirectional camera calibration from planar grids," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, pp. 3945–3950, 2007.

[9] V. Usenko, N. Demmel, and D. Cremers, "The double sphere camera model," in *Int. Conf. 3D Vision (3DV)*, pp. 552–560, 2018.

[10] B. Khomutenko, G. Garcia, and P. Martinet, "An enhanced unified camera model," *IEEE Robot. Autom. Lett.*, vol. 1, no. 1, pp. 137–144, 2015.

[11] J. Kannala and S. S. Brandt, "A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 8, pp. 1335–1340, 2006.

[12] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision.* Cambridge University Press, 2003. 2nd Edition.

[13] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128×128 120 dB 15 μs latency asynchronous temporal contrast vision sensor," *IEEE J. Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, 2008.

[14] C. Posch, D. Matolin, and R. Wohlgenannt, "A QVGA 143 dB dynamic range frame-free PWM image sensor with lossless pixel-level video compression and time-domain CDS," *IEEE J. Solid-State Circuits*, vol. 46, pp. 259–275, Jan. 2011.

[15] C. Brandli, R. Berner, M. Yang, S.-C. Liu, and T. Delbruck, "A 240x180 130dB 3μs latency global shutter spatiotemporal vision sensor," *IEEE J. Solid-State Circuits*, vol. 49, no. 10, pp. 2333–2341, 2014.

[16] G. Taverni, D. P. Moeys, C. Li, C. Cavaco, V. Motsnyi, D. S. S. Bello, and T. Delbruck, "Front and back illuminated Dynamic and Active Pixel Vision Sensors comparison," *IEEE Trans. Circuits Syst. II*, vol. 65, no. 5, pp. 677–681, 2018.

[17] T. Brodský, C. Fermüller, and Y. Aloimonos, "Structure from motion: Beyond the epipolar constraint," *Int. J. Comput. Vis.*, vol. 37, pp. 231–258, June 2000.

[18] D. Scaramuzza and F. Fraundorfer, "Visual odometry [tutorial]. Part I: The first 30 years and fundamentals," *IEEE Robot. Autom. Mag.*, vol. 18, pp. 80–92, Dec. 2011.

[19] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. D. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1309–1332, 2016.

[20] E. Mueggler, C. Bartolozzi, and D. Scaramuzza, "Fast event-based corner detection," in *British Mach. Vis. Conf. (BMVC)*, 2017.

[21] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239–256, 1992.

[22] R. A. Newcombe, A. J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon, "KinectFusion: Real-time dense surface mapping and tracking," in *IEEE ACM Int. Sym. Mixed and Augmented Reality (ISMAR)*, pp. 127–136, Oct. 2011.

[23] K. L. Lim and T. Bräunl, "A review of visual odometry methods and its applications for autonomous driving," *arXiv preprint arXiv:2009.09193*, 2020.

[24] J. Mo and J. Sattar, "Extending monocular visual odometry to stereo camera systems by scale optimization," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6921–6927, IEEE, 2019.

[25] W. Wang and D. Wang, "Land vehicle navigation using odometry/ins/vision integrated system," in *2008 IEEE Conference on Cybernetics and Intelligent Systems*, pp. 754–759, IEEE, 2008.

[26] C. Debeunne and D. Vivet, "A review of visual-lidar fusion based simultaneous localization and mapping," *Sensors*, vol. 20, no. 7, p. 2068, 2020.

[27] M. Franchi, A. Ridolfi, L. Zacchini, and B. Allotta, "Experimental evaluation of a forward-looking sonar-based system for acoustic odometry," in *OCEANS 2019-Marseille*, pp. 1–6, IEEE, 2019.

[28] V. Kirnos, V. Antipov, A. Priorov, and V. Kokovkina, "The lidar odometry in the slam," in *2018 23rd Conference of Open Innovations Association (FRUCT)*, pp. 180–185, IEEE, 2018.

[29] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Int. Joint Conf. Artificial Intell. (IJCAI)*, pp. 674–679, 1981.

[30] B. K. Horn and B. G. Schunck, "Determining optical flow," *J. Artificial Intell.*, vol. 17, no. 1, pp. 185 – 203, 1981.

[31] D. Weikersdorfer and J. Conradt, "Event-based particle filtering for robot self-localization," in *IEEE Int. Conf. Robot. Biomimetics (ROBIO)*, pp. 866–870, 2012.

[32] D. Weikersdorfer, R. Hoffmann, and J. Conradt, "Simultaneous localization and mapping for event-based vision systems," in *International Conference on Computer Vision Systems*, pp. 133–142, Springer, 2013.

[33] Z. Ni, A. Bolopion, J. Agnus, R. Benosman, and S. Régnier, "Asynchronous event-based visual shape tracking for stable haptic feedback in microrobotics," *IEEE Trans. Robot.*, vol. 28, no. 5, pp. 1081–1089, 2012.

[34] D. R. Valeiras, G. Orchard, S.-H. Ieng, and R. B. Benosman, "Neuromorphic event-based 3D pose estimation," *Front. Neurosci.*, vol. 9, p. 522, 2016.

[35] B. Kueng, E. Mueggler, G. Gallego, and D. Scaramuzza, "Low-latency visual odometry using event-based feature tracks," in *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, pp. 16–23, 2016.

[36] A. Z. Zhu, N. Atanasov, and K. Daniilidis, "Event-based feature tracking with probabilistic data association," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, pp. 4465–4470, 2017.

[37] D. Weikersdorfer, D. B. Adrian, D. Cremers, and J. Conradt, "Event-based 3D SLAM with a depth-augmented dynamic vision sensor," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, pp. 359–364, 2014.

[38] Y. Zuo, J. Yang, J. Chen, X. Wang, Y. Wang, and L. Kneip, "DEVO: Depth-event camera visual odometry in challenging conditions," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, pp. 2179–2185, 2022.

[39] C. Scheerlinck, H. Rebecq, T. Stoffregen, N. Barnes, R. Mahony, and D. Scaramuzza, "CED: color event camera dataset," in *IEEE Conf. Comput. Vis. Pattern Recog. Workshops (CVPRW)*, pp. 1684–1693, 2019.

[40] A. Z. Zhu, D. Thakur, T. Ozaslan, B. Pfrommer, V. Kumar, and K. Daniilidis, "The multivehicle stereo event camera dataset: An event camera dataset for 3D perception," *IEEE Robot. Autom. Lett.*, vol. 3, pp. 2032–2039, July 2018.

[41] L. Gao, Y. Liang, J. Yang, S. Wu, C. Wang, J. Chen, and L. Kneip, "VECtor: A Versatile Event-Centric Benchmark for Multi-Sensor SLAM," *IEEE Rob. Autom. Lett.*, vol. 7, pp. 8217–8224, June 2022.

[42] "Combined Dynamic Vision / RGB-D Dataset kernel description." https://ebvds.neurocomputing.systems/EBSLAM3D/index.html. Accessed: 2022-10-22.

[43] C. Scheerlinck, N. Barnes, and R. Mahony, "Continuous-time intensity estimation using event cameras," in *Asian Conf. Comput. Vis. (ACCV)*, pp. 308–324, 2018.

[44] H. Kim, A. Handa, R. Benosman, S.-H. Ieng, and A. J. Davison, "Simultaneous mosaicing and tracking with an event camera," in *British Mach. Vis. Conf. (BMVC)*, 2014.

[45] M. B. Milde, O. J. N. Bertrand, H. Ramachandran, M. Egelhaaf, and E. Chicca, "Spiking elementary motion detector in neuromorphic systems," *Neural Computation*, vol. 30, pp. 2384–2417, Sept. 2018.

[46] "Inivation DAVIS346," 2021. [Online; accessed 1. Jul. 2023].

[47] C. Brandli, L. Muller, and T. Delbruck, "Real-time, high-speed video decompression using a frame- and event-based DAVIS sensor," in *IEEE Int. Symp. Circuits Syst. (ISCAS)*, pp. 686–689, 2014.

[48] D. MARRI, "A computational theory of human stereo visionf," *Proceedings of the Royal Society of London. Series B, Biological Sciences*, vol. 204, no. 1156, pp. 301–328, 1979.

[49] S. Fuchs and S. May, "Calibration and registration for precise surface reconstruction with time-of-flight cameras," *International Journal of Intelligent Systems Technologies and Applications*, vol. 5, no. 3-4, pp. 274–284, 2008.

[50] Z. Zhang, R. Deriche, O. Faugeras, and Q.-T. Luong, "A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry," *Artificial intelligence*, vol. 78, no. 1-2, pp. 87–119, 1995.

[51] M. Corporation, "Microsoftkinect camera." Available at: https://www.microsoft.com/en-us/d/azure-kinect-dk/8pp5vxmd9nhq (Accessed: 02.07.23.

[52] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2011.

[53] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3d: A modern library for 3d data processing," *arXiv preprint arXiv:1801.09847*, 2018.

[54] M. Kazhdan and H. Hoppe, "Screened poisson surface reconstruction," *ACM Transactions on Graphics (ToG)*, vol. 32, no. 3, pp. 1–13, 2013.

[55] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pp. 303–312, 1996.

[56] "IntelRealsense Depth Camera D435 kernel description." https://www.intelrealsense.com/depth-camera-d435/. Accessed: 2023-04-10.

[57] "Intel RealSense Depth Camera D435 Datasheet," 2023. [Online; accessed 1. Jul. 2023].

[58] T. Delbruck, V. Villanueva, and L. Longinotti, "Integration of dynamic vision sensor with inertial measurement unit for electronically stabilized event-based vision," in *IEEE Int. Symp. Circuits Syst. (ISCAS)*, pp. 2636–2639, 2014.

[59] L. A. Camunas-Mesa, T. Serrano-Gotarredona, S. H. Ieng, R. B. Benosman, and B. Linares-Barranco, "On the use of orientation filters for 3D reconstruction in event-driven stereo vision," *Front. Neurosci.*, vol. 8, p. 48, 2014.

[60] "Combined Dynamic Vision / RGB-D Dataset kernel description." https://github.com/ethz-asl/kalibr/wiki/calibration-targets. Accessed: 2022-10-22.

[61] "Kalibr kernel description." https://github.com/ethz-asl/kalibr. Accessed: 2022-09-10.

[62] "melodic - ROS Wiki," July 2023. [Online; accessed 2. Jul. 2023].

[63] "Kalibr Supported Models kernel description." https://github.com/ethz-asl/kalibr/wiki/supported-models#references. Accessed: 2022-09-10.

[64] X. Huang, G. Mei, J. Zhang, and R. Abbas, "A comprehensive survey on point cloud registration," *arXiv preprint arXiv:2103.02690*, 2021.

[65] W. Burger and B. Bhanu, "Estimating 3d egomotion from perspective image sequence," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, pp. 1040–1058, 1990.

[66] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second ed., 2004.

[67] "Pyoints documentation," apr 2019. [Online; accessed 1. Jul. 2023].

[68] "Open3D: A Modern Library for 3D Data Processing — Open3D 0.16.0 documentation," Oct. 2022. [Online; accessed 1. Jul. 2023].

[69] "The Stanford 3D Scanning Repository," July 2023. [Online; accessed 1. Jul. 2023].

[70] M. Grupp, "evo: Python package for the evaluation of odometry and slam.." https://github.com/MichaelGrupp/evo, 2017.

[71] "Computer Vision Group - File Formats," July 2023. [Online; accessed 1. Jul. 2023].

[72] "The KITTI Vision Benchmark Suite," July 2023. [Online; accessed 1. Jul. 2023].

[73] "kmavvisualinertialdatasets – ASL Datasets," July 2023. [Online; accessed 1. Jul. 2023].

[74] "Kalibr Supported Models kernel description." https://github.com/ethz-asl/kalibr/wiki/downloads. Accessed: 2022-09-10.

[75] 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 *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, pp. 3867–3876, 2018.