Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Autonomous Systems Lab
Prof. Roland Siegwart

Master Thesis

Use of a DVS for High Speed Applications

Autumn Term 2018

Supervised by:
Florian Tschopp
Ignacio Alzugaray

Author:
Weizhen Huang

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Authored by (in block letters):

For papers written by groups the names of all authors are required.

Name(s): **First name(s):**

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date **Signature(s)**

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.

Contents

Preface

Bla bla …

Abstract

Hier kommt der Abstact hin ...

Symbols

Symbols

ϕ, θ, ψ	roll, pitch and yaw angle
b	gyroscope bias
Ω_m	3-axis gyroscope measurement

Indices

x	x axis
y	y axis

Acronyms and Abbreviations

ETH	Eidgenössische Technische Hochschule
EKF	Extended Kalman Filter
IMU	Inertial Measurement Unit
UAV	Unmanned Aerial Vehicle
UKF	Unscented Kalman Filter

Chapter 1

Introduction

Frame-based cameras that widely used in computer vision output images at a preset rate, even when the intensity values stay unchanged, resulting redundant data. When the scene moves too fast, an insufficient frame rate would cause motion blur. To address this problem, Lichtsteiner et al. [1] presents an event-based camera called DVS (dynamic vision sensor) that reports pixel-level log intensity changes at a rate of MHz scale. Unlike traditional cameras that outputs a frame until all the pixels are scanned, an event-based camera outputs are asynchronous: when the intensity change at a pixel reaches a threshold $\mid \ln I(t) - \ln I(t - \Delta t) \mid > C$, the sensor outputs an "event" $e = \{x, y, t, p\}$, which includes the pixel coordinate (x, y), the timestamp t and the polarity $p = \pm 1$ indicating positive or negative intensity change. The output is thus a stream of events instead of frames. Later Brandli et al. [2] presents DAVIS (dynamic and active pixel vision sensor) that has additional APS (active pixel sensor) circuits that provides absolute intensity information, while offering DVS outputs with higher resolution (240×180 against 128×128), higher dynamic range (130 dB against 120 dB) and lower latency (3 μs against 15 μs). Newer sensors also provide color information[3, 4] or have higher image resolution[5].

Since for an event-base camera there is no such thing as a frames, feature detection and tracking algorithms that work well for standard camera can not be directly applied to an event-based camera. There are several works that try to adopt feature detection and tracking pipelines. Zhu et al. [6] aggregate a small number of events to construct a frame, and perform Harris corner detector [7] on the synthesized frames, then track the features with implicit data association. The work of Tedaldi et al. [8] detects features on the APS outputs, and performs tracking with the DVS outputs. Instead of working with frames, Mueggler et al. [9] detect corners directly in the spatiotemporal event stream. There are also works that perform 3D reconstruction with known poses [10] or 6-DoF tracking with a known map [11], or both with the help of IMUS which is integrated on DAVIS [12]. Besides, [13, 14] perform 6-DoF tracking and 3D reconstruction purely based on event streams, with methods commonly applied in computer vision, for example DSI (disparity space image) or EKF (extended Kalman Filter). Newer works also combine machine learning and event-based camera [15, 16, 17]

Gallego and Scaramuzza [18] first proposes an interesting contrast maximization framework that is rather specific for event-based cameras. Without the help of any auxiliary variable such as optical flow or feature, this framework finds the optimal angular velocity that maximizes the contrast of a synthesized event frame via nonlinear optimization. Later in [19] they showed that the same framework can be applied to various important tasks in computer vision, such as motion, depth and optical flow estimation.

This work is an extension of [18, 19]. In these two works they showed how to estimate angular velocity in general scenes where only rotational motion is present, and 6-DoF motion in planar scenes with contrast maximization framework. In this work we will see that the same idea can be applied to perform SLAM (simultaneously localizing and mapping) in planar scenes, and motion estimation in general scenes with general motion.
In this work we only used the DVS output of DAVIS[2]. The algorithms are tested on the event-camera dataset recorded by Mueggler et al. [21]. The outline of this work is as follows:

Chapter 2

Per-Frame Optimization

The output of a DVS is a stream of events $\{e_k\}$, with $e_k = \{x_k, y_k, t_k, p_k\}$ denoting its pixel coordinate, timestamp and polarity, respectively. Figure 2.1 shows the DVS output interleaved with the APS output. We can clear see that the events are triggered at the shape boundaries: when the sensor (or the scene) is moving, these pixel locations will receive intensity changes. Thus DVS works naturally as an edge detector.

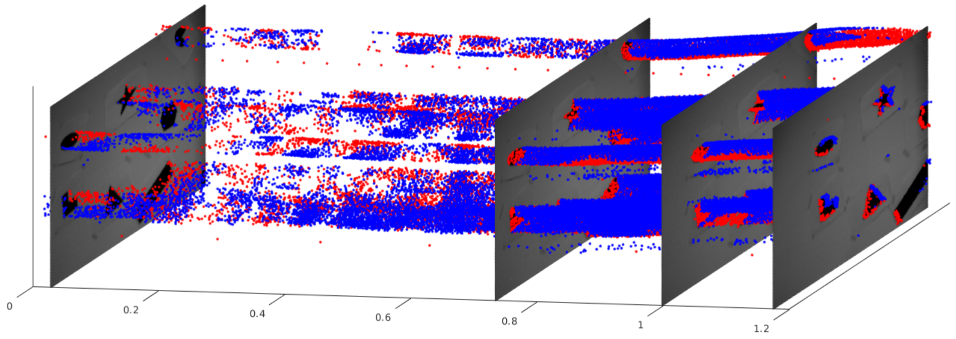


Figure 2.1: Event stream plus grayscale images. During this period 100 000 events and 27 frames are produced. Red and blue points denote the positive and negative events, respectively. DVS has a much higher output rate and is free from motion blur.

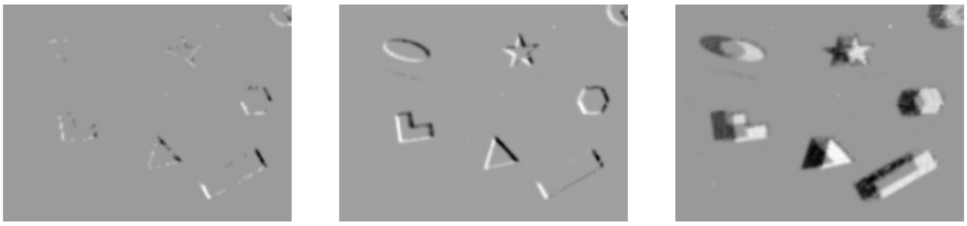
Note that events can be triggered either by scene or camera motion. In this work we assume a static scene and a moving camera.

If we aggregate all the events within a temporal window, depending on the size of the window we may get different results as in fig. 2.2. From now on we call such synthesized results as *frames*.

The temporal resolution of a DVS in [2] is 1 µs; each event has a distinct timestamp, which also indicates that events happen at distinct camera poses. Instead of simply mingling events from different timestamps, a better strategy is to project the events along the motion path, which is clearly visible in fig. 2.1, thus faithfully recover the scene edges at a given time.

2.1 From Events to Frame

But how do we compute the motion path? Without additional motion capture devices, we have no access to the ground truth pose; it's also impossible to know



(a) 300 (b) 3000 (c) 30 000

Figure 2.2: The image generated by aggeragating 300, 3000 and 30 000 events. White and black dots represent positive and negative events, respectively. Gray indicates that no event is present in this pixel, or positive and negative events cancel each other out.

the camera pose associated with each event, which happens at such a high rate. To simplify the problem, we assume that the camera moves at a constant velocity within a period, which is not a bad assumption if the temporal window is small enough. The next step is to find the velocity that best describes the events within this window.

More specifically, we group a set of events $\mathscr{E} \doteq \{e_k\}_{k=1}^N$ within a certain temporal window, and synthesize a *frame* by summing up the events. we simply sum along the time axis, the intensity at each pixel will be the sum of the polarities of all the events that are triggered at this pixel location within the window

$$\mathcal{I}(\boldsymbol{x}) = \sum_{k=1}^{N} \pm_k \delta(\boldsymbol{x} - \boldsymbol{x}_k), \tag{2.1}$$

with $\pm_k$ and $\boldsymbol{x}_k$ denoting the polarity and pixel coordinates of the kth event, respectively.

If the motion and scene parameters within this window, then shift the window to the next set of events and repeat this process. The temporal window size is defined by the event numbers N, which should be chosen small enough so that a constant velocity model could be applied within this window. We choose event numbers against a fixed time interval to define the window size, because this corresponds to the data-driven nature of an event-based camera: the more rapid the apparent motion of the scene is, the larger the event rate will be. If the scene stops moving, no events will be generated, the pose will also not be further updated.

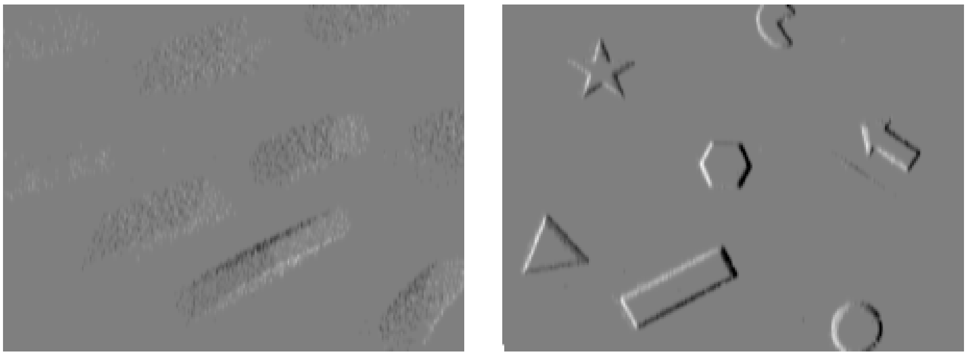
If After warping the events with $\boldsymbol{x}'_k = \boldsymbol{W}(\boldsymbol{x}_k, t; \theta)$, we substitute $\boldsymbol{x}_k$ in the above equation to $\boldsymbol{x}'_k$.

2.2 Measuring the Sharpness of an Image

There are several metrics one could choose from to measure the contrast of an image. A local contrast metric could be, for example, convoluting the image with a high pass filter

$$\mathcal{C}_H = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}, \tag{2.2}$$

and sum the pixel value of the filtered image. However, this metric only compare a pixel with its 8 neighbors, thus an image with scattered events (large noise) is also a valid configuration, as shown in fig. 2.3(a). The Michelson contrast [20], defined as $\mathcal{C}_M = (\mathcal{I}_{\max} - \mathcal{I}_{\min}) \,/\, (\mathcal{I}_{\max} + \mathcal{I}_{\min})$, only considers the highest and lowest



(a) An optimized frame using sharpening filter metric

(b) An optimized frame using variance metric

Figure 2.3: Comparison between two different metrics

luminance in the image and is thus more suitable to quantify contrast for periodic functions.

We choose to measure the contrast by the variance of the image, defined by

$$\mathrm{Var}\left(\mathcal{I}\left(\boldsymbol{x};\boldsymbol{\theta}\right)\right) \doteq \frac{1}{\mid \Omega \mid} \int_{\Omega} \left(\mathcal{I}\left(\boldsymbol{x};\boldsymbol{\theta}\right) - \mu\left(\mathcal{I}\left(\boldsymbol{x};\boldsymbol{\theta}\right)\right)\right)^{2} d\boldsymbol{x}, \tag{2.3}$$

where $\mu\left(\mathcal{I}\left(\boldsymbol{x};\boldsymbol{\theta}\right)\right)$ is the average image intensity. If the scene has more or less balanced positive and negative events, the average will be close to zero. Actually, since we only want to distinguish the edges with the larger blank background, which indeed has zero intensity, we can also simply substitute $\mu\left(\mathcal{I}\left(\boldsymbol{x};\boldsymbol{\theta}\right)\right)$ with 0.

The goal of measuring the contrast of an image is to find the configuration that aligns events triggered from the same visual stimuli; thus the variance eq. (2.3) would be a very suitable metric, as a squared metric favors the configuration that projects as many events as possible to the same pixel.

Intergral sum pixels

Chapter 3

6DoF Pose Tracking in Planar Scenes

Throughout this work the following notation is employed: W denotes the world frame, C_1 or C_2 denotes a camera frame. T_{AB} is the transformation from frame A to frame B, measured in frame A.
$\boldsymbol{X}$ the position of the event with respect to world or camera frame, $\boldsymbol{x}$ the calibrated coordinates of the event.

3.0.1 Planar Homography

The warp function $\boldsymbol{x}' = \boldsymbol{W}(\boldsymbol{x}, t; \boldsymbol{\theta})$ does not only depend on the motion parameters, but also the scene parameters, which is the unknown depth. In the case of a planar scene the problems simplifies, since a plane $\mathbf{P}$ can be parameterized by two sets of parameters: $\boldsymbol{n} \in \mathbb{S}^2$ the unit surface normal of $\mathbf{P}$ with respect to the current camera frame, and d the distance from the camera center to $\mathbf{P}$. The warp function then becomes

$$\begin{aligned} \boldsymbol{X}' =& \boldsymbol{R}(t)\boldsymbol{X} + \boldsymbol{T}(t) && (3.1)\\ \boldsymbol{X} =& \boldsymbol{R}(t)^\top \left(\boldsymbol{X}' - \boldsymbol{T}(t)\right) && (3.2)\\ \boldsymbol{X} =& \boldsymbol{R}(t)^\top \left(\boldsymbol{I} + \boldsymbol{T}(t)\boldsymbol{n}^\top/d\right)\boldsymbol{X}', && (3.3) \end{aligned}$$

thus $\boldsymbol{x}' \sim \left(\boldsymbol{R}(t)^\top \left(\boldsymbol{I} + \boldsymbol{T}(t)\boldsymbol{n}^\top/d\right)\right)^{-1}\boldsymbol{x}$. Here $(\boldsymbol{R}(t), \boldsymbol{T}(t)) \in SE(3)$ denotes the relative pose between two cameras at which the current event being warped and the first event within the window happened, and t is the relative timestamp with respect to the first event. Under a constant velocity model with linear velocity $\boldsymbol{v} \in \mathbb{R}^3$ and angular velocity $\boldsymbol{\omega} \in \mathbb{R}^3$, the translation is given by

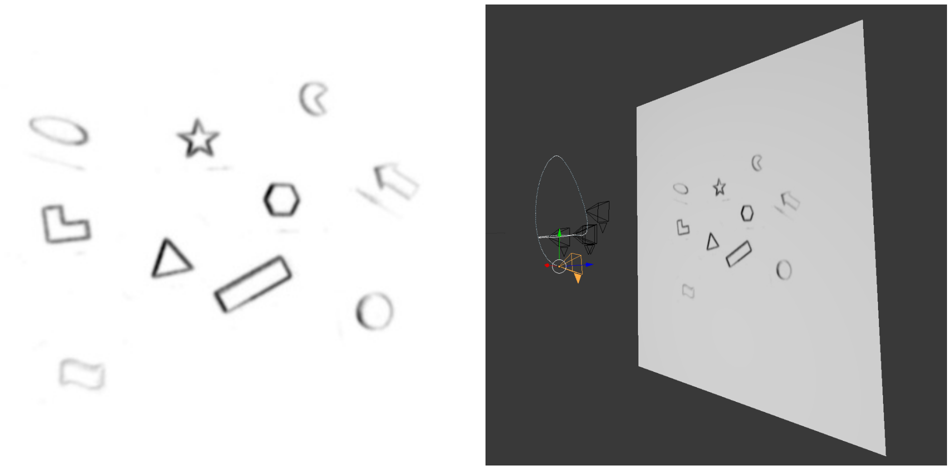
$$\boldsymbol{T}(t) = \boldsymbol{v}t, \tag{3.4}$$

the rotation matrix is given by the *exponential map* exp: $\mathfrak{so}(3) \rightarrow SO(3)$:

$$\boldsymbol{R}(t) = \exp(\boldsymbol{\omega}^\wedge t), \tag{3.5}$$

where $^\wedge$ is the *hat* operator

$$\boldsymbol{\omega}^\wedge = \begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\end{bmatrix} = \begin{bmatrix}0 & -\omega_3 & \omega_2\\ \omega_3 & 0 & -\omega_1\\ -\omega_2 & \omega_1 & 0\end{bmatrix} \in \mathfrak{so}(3). \tag{3.6}$$



(a) The texture of a map (b) A map in the global frame

Figure 3.1: Map

3.1 From Frames to Map

The contrast maximization procedure in the above section optimizes the relative pose between successive frames. We show in this section that the same idea can be applied to perform global pose tracking in planar scenes. We first explain how the map is defined, and how to track a known map, then we shown how this map is built by selecting a set of keyframes.

3.1.1 Map

A map is a plane with three components: the normal direction $\boldsymbol{n}_w$, the distance d_w to the origin, and the texture; the texture of a map represents all the edges on the plane. Figure 3.1 shows the an example of such map. Figure 3.1(a) also shows the set of keyframes used to construct the map. We will talk more about keyframes in section 3.1.3. The global coordinate is chosen as the camera coordinate of the first frame.

3.1.2 Tracking

Suppose a map is present, then the normal direction $\boldsymbol{n}_w$ of the plane and the distance d_w to the origin are known. Also the pose of the current frame $(\boldsymbol{R}_{wc}, \boldsymbol{T}_{wc}) \in SE(3)$ is determined by the motion estimation from the last frame (a quick note to the terminology we are using: whenever we say the *pose* of a frame, we always refer to the camera *pose* at which the first event within the frame happens). The parameters left to be estimated for each frame is $\phi = (\boldsymbol{\omega}, \boldsymbol{v}) \in \mathbb{R}^6$. By substituting $\boldsymbol{n}$ with $\boldsymbol{n}_c = \boldsymbol{R}_{cw}\boldsymbol{n}_w$, and d with $d_c = d_w + \boldsymbol{T}_{wc} \cdot \boldsymbol{n}_w$ in eq. (3.3), we get the homography matrix within each frame as

$$\boldsymbol{H}_1 = \boldsymbol{R}^\top \left(\boldsymbol{I} + \boldsymbol{T}\boldsymbol{n}_c^\top / d_c\right) \tag{3.7}$$

and $\boldsymbol{x}'_c \sim \boldsymbol{H}_1^{-1} x_c$.

A nonlinear optimizing problem naturally suffers from local optima. Without a good initialization, the motion computed with the method in section 2.1 could sometimes be a local optimum delivering an image that appears sharp, despite being wrongly estimated (see fig. 3.2). In order to make sure that the estimated motion from the per frame contrast maximization also conform to the global map. Thus we perform another optimization, where we project the events of the current frame to the global map. The parameter set is still $\left(\boldsymbol{\omega}^\top, \boldsymbol{v}^\top\right)$, and we use the output from last procedure as an initial guess.

The procedure described in the first paragraph of this section can be understood as projecting the events on a *blank canvas*. Similarly, in the projecting-to-map procedure we project the events on the *texture* of the map, and measure the strength of the synthesized image with the same variance function as in eq. (2.3), thus finding the set of the parameters that best align the events in the current frame to their correspondences in the texture.

The projection from an event to the map is

$$\boldsymbol{x}_w \sim \boldsymbol{R}_n \boldsymbol{H}_2^{-1} \boldsymbol{H}_1^{-1} \boldsymbol{x}_c, \tag{3.8}$$

with $\boldsymbol{R}_n$ the transformation from the orientation of the global frame to the orientation of the map, computed by

$$\boldsymbol{K} = (\boldsymbol{n}_w \times \boldsymbol{z})^\wedge \tag{3.9}$$

$$\boldsymbol{R}_n = \boldsymbol{I} + \boldsymbol{K} + \boldsymbol{K}^2/(1 + \boldsymbol{n_w} \cdot \boldsymbol{z}), \tag{3.10}$$

where $\boldsymbol{z} = (0, 0, -1)$ denotes the plane fronto-parallel to the camera, and

$$\boldsymbol{H}_2 = \boldsymbol{R}_{cw} \left(\boldsymbol{I} + \boldsymbol{T}_{wc} \boldsymbol{n}^\top / d_w\right) \tag{3.11}$$

is the projection from the current frame to the global frame, with $\boldsymbol{R}_{wc}, \boldsymbol{T}_{wc}$ being the pose of the current frame. $\boldsymbol{H}_1$ is the planar homography for each frame as in eq. (3.7). But the $\boldsymbol{R}(t)$ and $\boldsymbol{T}(t)$ might be different since we are refining these parameters.

After projecting, we obtain an image composed of the map texture and the events of the current frame. We maximize the contrast of this image using the cost function defined in eq. (2.3). The optimized velocity is used for propagating the pose to the next incoming frame via

$$\boldsymbol{T}_{wc_2} = \boldsymbol{R}_{wc_1} \boldsymbol{v} \Delta t + \boldsymbol{T}_{wc_1} \tag{3.12}$$

$$\boldsymbol{R}_{wc_2} = \boldsymbol{R}_{wc_1} \exp(\boldsymbol{\omega}^\wedge \Delta t), \tag{3.13}$$

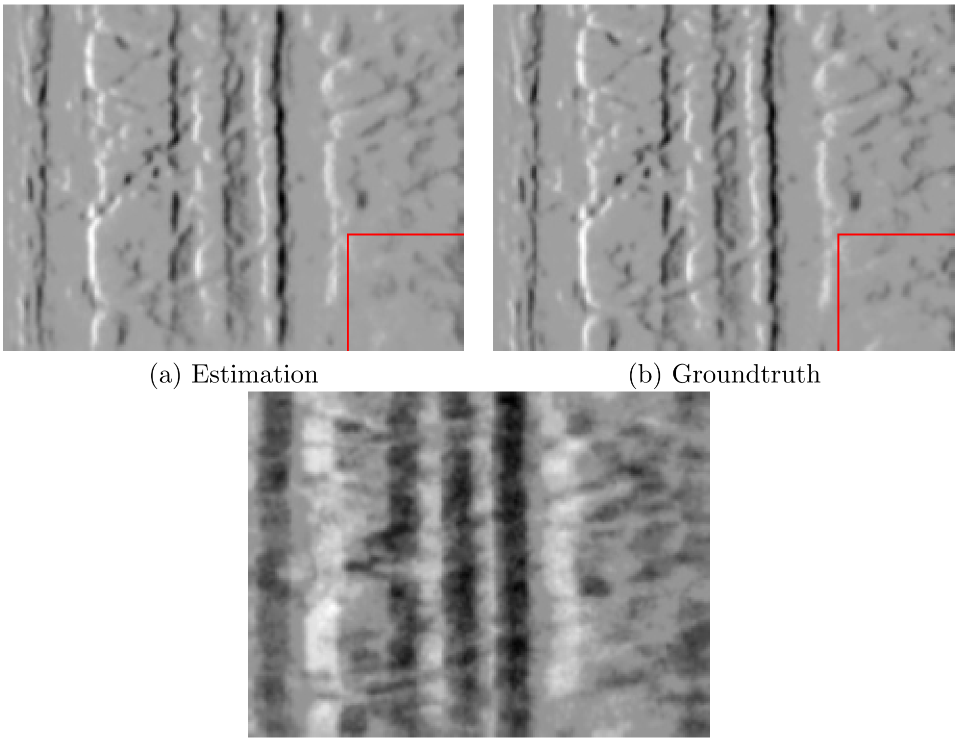
where c_1 and c_2 denotes the current frame and the next frame, respectively, and Δt is the temporal size of the current frame.

3.1.3 Mapping

After having collected the first N events, we start the mapping process. For the first frame we estimate the full parameter set $\phi = \left(\boldsymbol{\omega}^\top, \boldsymbol{v}^\top / d_w, \varphi, \psi\right)^\top \in \mathbb{R}^8$, where (φ, ψ) parametrize the unit vector of the plane normal, and $\boldsymbol{v}^\top / d_w$ account for the scale ambiguity problem introduced in linear velocity estimation from monocular camera. We can for example determine the scale of the scene by setting d_w to $1m$, then scales of the consecutive frames are also determined by $d_c = d_w + \boldsymbol{T}_{wc} \cdot \boldsymbol{n}_w$.

From the optimized first frame we initialize the map, by projecting the frame to the planar scene via a rotation $\boldsymbol{R}_n$ as in eq. (3.10) (you might have implemented this part wrong). Then we can track the next frames with this map using the method described in section 3.1.2.

As the camera moves, there might be new information available in the scene, so that the map needs to be expanded. After optimization for each single frame, we also measure the *per-pixel-overlap* between the frame and the map,S that is, how many percent pixels in the frame can be explained by the map. The overlap might be small, when the camera is just exploring a new area so that only part of the frame



(a) Estimation

(b) Groundtruth

(c) Without motion compensation

Figure 3.2: An example of local optima. This is the dataset *slider_hdr_close* with a window size of 50000 events. (a) shows the optimized image with *linear velocity* $(0.231, 0.109, 0.256)$, *angular velocity* $(0.405, -0.130, -0.278)$ and *plane normal* $(-0.579, 0.282, -0.765)$. (b) shows the result using groundtruth parameters with *linear velocity* $(0.163, 0, 0)$, *angular velocity* $(0, 0, 0)$ and *plane normal* $(0, 0, -1)$. Both images appear mostly identical, though at the lower right corner, for example, one can still recognize the difference. Also both images look much sharper than the image without motion compensation in (c). It is worth mentioning that the contrast of the estimation is actually slightly larger than that of the groundtruth

and the map is overlapping; another possibility is that the map is not accurate enough to explain the current frame, which often happens when only one frame is used until now to construct the map. When the overlap reaches a certain threshold (0.8 for example), we try to insert a new *keyframe*.

When the system decides that a new keyframe is needed, we collect all the keyframes until now, plus the current frame, which is a keyframe candidate, and optimize the poses and velocities of these frames, as well as the map together. Suppose there are k frames (including the keyframe candidate), the to be optimized parameter set is

$$\phi = \left(\boldsymbol{R}_{(1\sim k)}, \boldsymbol{T}_{(1\sim k)}, \boldsymbol{\omega}_{(0\sim k)}, \boldsymbol{v}_{(0\sim k)}/d_w, \varphi, \psi\right) \in \mathbb{R}^{12k-4}.$$

Here $1 \sim k$ means from frame 1 to frame k, and we skip the pose optimization of frame 0, which is the first frame, since it defines the global coordinate. We project all the events of these k frames to the plane parametrize by (φ, ψ) with eq. (3.8), and again optimize the contrast of the synthesized image the obtain the optimal parameter set.

Note that in the mapping process we drop the polarity of the event, since different keyframes might include events from the same visual stimuli but triggered when moving in different directions, when summed together the polarities might cancel each other out. For the same reason, we also don't use the polarity when matching a frame to the map.

After this step, we also measure the *per-pixel-overlap* between the keyframe candidate and the image synthesized by the other keyframes, with the newly optimized parameter set. If this is a good match (define a good match), we continue the tracking. Otherwise, we consider the pose estimation of the current frame to be already off. In this case we preserve the former map, and use this map to start the *relocalization* procedure.

3.1.4 Relocalization

3.2 Experiments

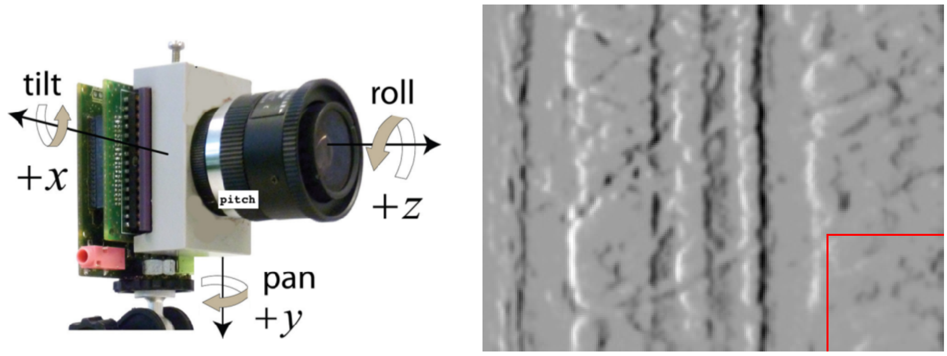
The algorithm is tested on four planar datasets provided by the *Robotics and Perception Group* at UZH [21], which includes *poster_6dof, poster_translation, shapes_6dof, shapes_translation*. The *shapes* dataset contains several simple shapes on the wall, whereas the *poster* contains richer rock texture. A planar assumption is redundant for the rotation datasets and thus the results are not listed for comparison here.

We define the camera axes as in fig. 3.3. The camera coordinate frame of the first synthesized frame is set as the global coordinate frame. The x, y and z axes are also pitch yaw, roll axes, respectively.

Figure 3.6 shows the comparison of the results of the proposed method against ground truth on the whole *shapes_6dof* sequence. We observe that the orientation and translation estimation along z axis is the most accurate among all the axes; when zoomed in (fig. 3.7), it is clear to see that the roll estimation is almost indistinguishable from the ground truth. We split the quantitative error estimation for orientation along 3 axes (section 3.2), in order to show that this holds true for all the datasets we tested on. In these datasets, the global z axis is close to the plane normal; we believe that the events generated by a motion along the plane normal direction is more obvious, and also harder to be explained by a combination of other motions, and are thus more reliable.

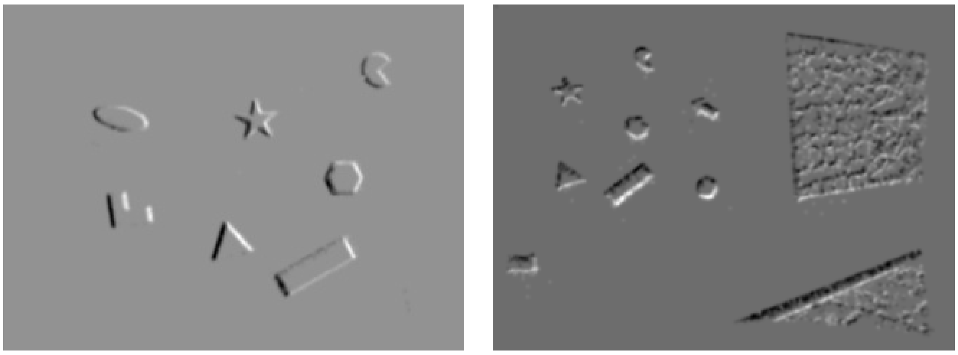
We also observe in fig. 3.6 that in the latter half of the sequence there are larger fluctuations. There are multiple reasons for that:

1. The window size is not suitable



(a) (b)

Figure 3.3: Axes definition



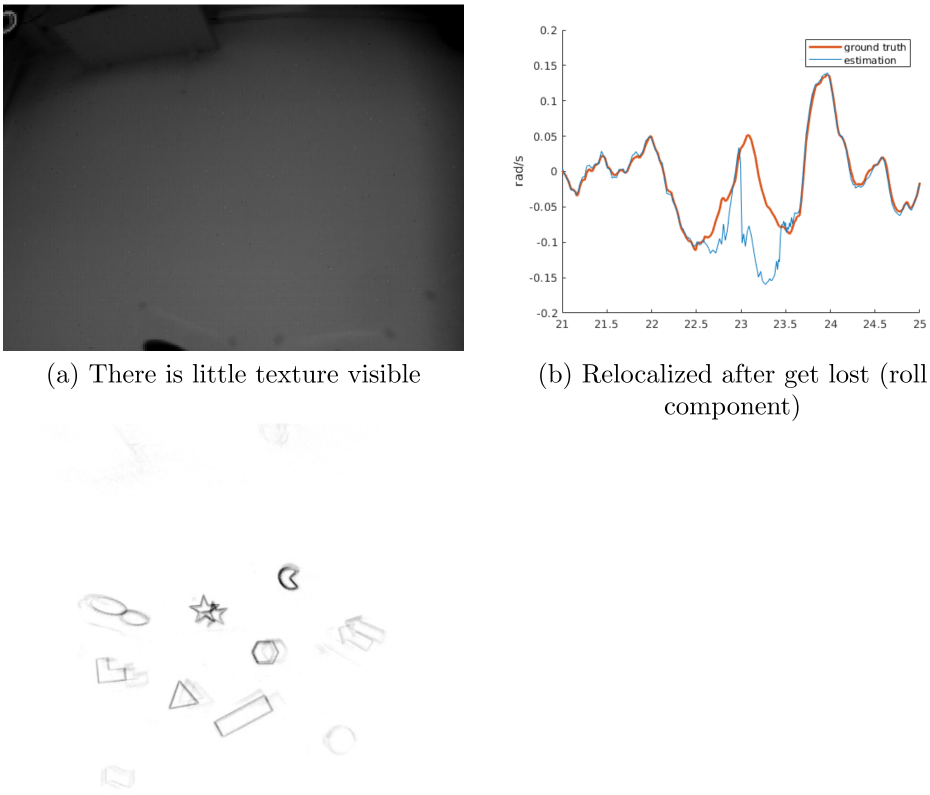
(a) (b)

Figure 3.4: A window size of 4000 for a specific dataset *shapes_6dof* but different scene complexity

The window size is manually chosen for each dataset. A range of $2000 - 5000$ usually works good for the *shapes* datasets, since the scene consists of simple regular shapes. Figure 3.4(a) shows what the camera frustum normally contains. However, in the rapid movement phase, the camera very often moved to a position where other scenes which don't belong to *shapes* also becoms visible. Figure 3.4(b) shows the scene of *shapes_6dof* at around 47.8 seconds. At this moment the poster on the wall is also visible, which actually belongs to the *poster* sequences. The poster has much more complex textures, when performing pose estimation on the *poster* sequences, we normally choose a window size of $30000 - 50000$. Apparently, a window size of 4000 is no longer enough in this case, which causes the motion estimation in this period being inaccurate.

2. There are few textures available

   A very representative example is that when tracking the dataset *shapes_translation*, we always get lost at around $23sec$. Investigating the scene around this time instance (fig. 3.5(a)), we found that very few or even no texture is available. It's also not possible to perform a reliable pose estimate during this period. For the *per-frame* velocity estimation proposed in [18, 19], where each frame is uncorrelated ?, this is usually not a serious problem, since when the textures becomes available again, the optimization pipeline is still able to continue, it just might be harder to start from a bad initialization. However, for the pose estimation where a cumulative result is needed, a few bad estimated frames



(a) There is little texture visible

(b) Relocalized after get lost (roll component)

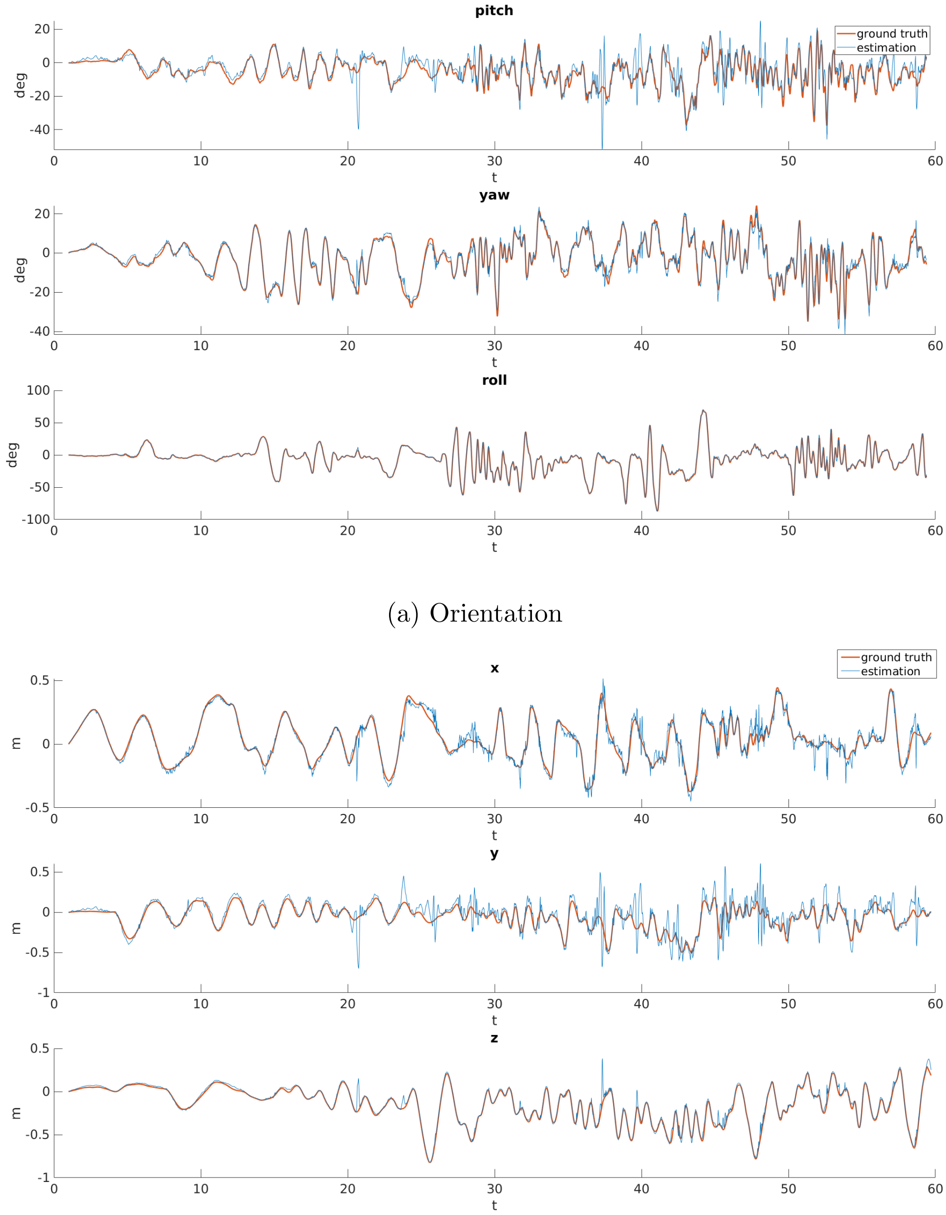
(c) Polluted map after tracking is lost

Figure 3.5: When tracking the dataset *shapes_translation*, we always get lost at around 23*sec*, but there is still chance to relocalize

might destroy the whole following sequence.

The *bundle adjustment* and *relocalization* parts of the pipeline is designed to deal with such situations. However, a bad initial guess of the pose $(\boldsymbol{R}, \boldsymbol{T})$ is generally harder to start with than a bad initial guess of velocities $(\boldsymbol{\omega}, \boldsymbol{v})$. When the pose estimation from last frame is already too off, after optimization there might still be little overlap between the current frame and the pose. This is when the *bundle adjustment* tries to insert a keyframe and track from the new keyframe. The current implementation doesn't reinitialize the map after the tracking is lost. The advantage is that there is still chance to come back to the original map after tracking a wrong "local map" (fig. 3.5(b)); the disadvantage is that the global map will most likely be polluted by the "local map" (fig. 3.5(c))

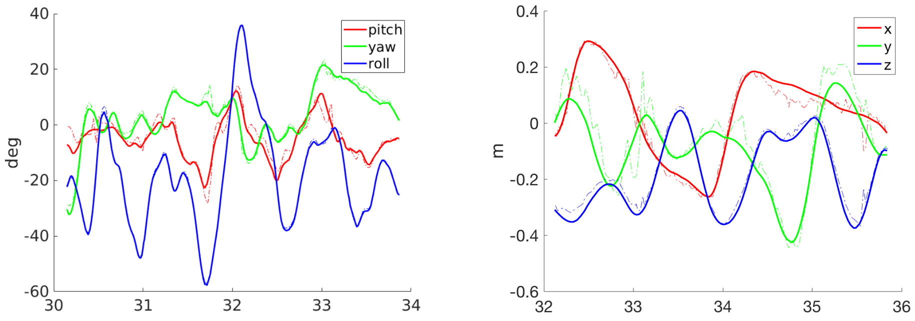
A quantitative error estimation is included in section 3.2. The window sizes for *shapes* and *poster* sequences are 4000 and 30000, respectively. Also, we used the *Nelder-Mead Simplex* algorithm provided by *GNU-GSL*[22] to perform the optimization on *shapes* sequences, and the *Fletcher-Reeves conjugate gradient*[23] algorithm to perform optimization on the *poster* dataset. Both methods work. The analytic differentiation is faster and more stable. However, when the initial guess is further from the ground truth, the numeric differentiation seems to work better than the analytic differentiation, since the jacobians used for analytic differentiation



(a) Orientation

(b) Translation

Figure 3.6: *shapes_6dof* sequence. Comparison of the estimated pose (blue line) against ground truth (red line).



(a) Orientation (b) Translation

Figure 3.7: *shapes_6dof* sequence. Comparison of the estimated pose (dashed line) against ground truth (full line) within segments of 8 seconds duration.

are strictly local (see appendix). whereas a properly chosen step size for numeric differentiation

- median

- percent

- scene depth no drift

Dataset	Median Orientation Error/deg			Median Translation Error/m	Traveled Distance	Note
	pitch	yaw	roll			
shapes_6dof	1.89	1.01	0.85	0.047	46.89	Tracked all 60s
shapes_translation	0.85	0.45	0.27	0.019	14.08	Lost after 25.0s
poster_6dof	1.14	1.22	0.63	0.054	12.50	Lost after 22.8s
poster_translation	0.55	0.57	0.25	0.022	11.14	Lost after 26.0s

Table 3.1: Quantitative evaluation on planar sequences

Despite of the rapid movements, in the above datasets the camera actually only moves in a relative small region in front of the textured plane, as depicted in fig. 3.8. However, this method also applies when camera travels a longer distance with respect to the scene depth, as shown in fig. 3.9. Also, although this method is designed for planar scenes, in a complex scene with rotation only movement, where no scene parameters are needed, this method can still be applied to build a global map, delivering a result similar to *image stitching*. Figure 3.10 is such an example.

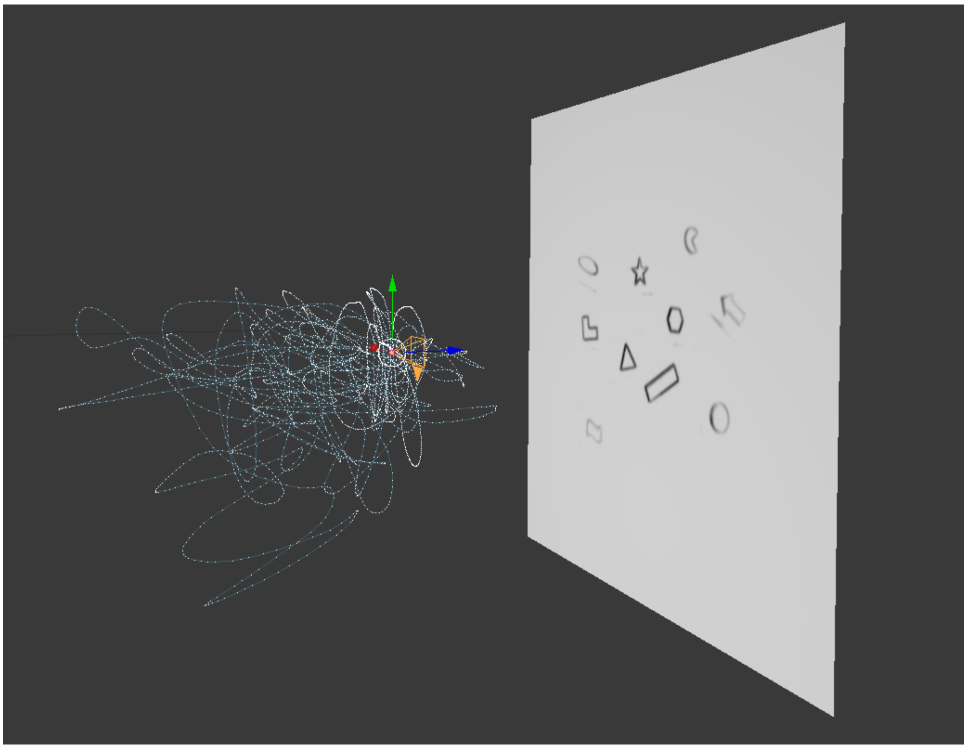


Figure 3.8: The motion path of the dataset *shapes_6dof*. Despite a trajectory length of about $50m$, in the whole $60s$ the camera actually stays in a relative small region compared to the scene depth, and there is no observable increase of drift using the method in this work. ref to error estimation

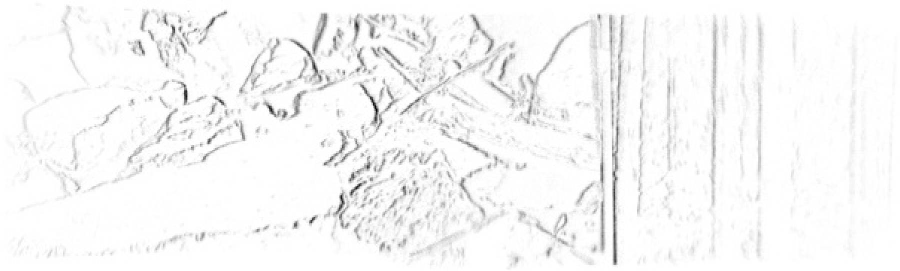


Figure 3.9: The dataset *slider_hdr_far*, with a scene depth of $0.584m$, and a camera movement in the positive x direction only. Note that this is a much wider map than what we have seen before. The figure shows the result at $4.8sec$ within the whole range of $6.3sec$; afterwards the algorithm is lost in the forest, where almost all the textures are vertical, causing severe local optima problem. If we constrain the motion estimation to translation only, it delivers a much better result.

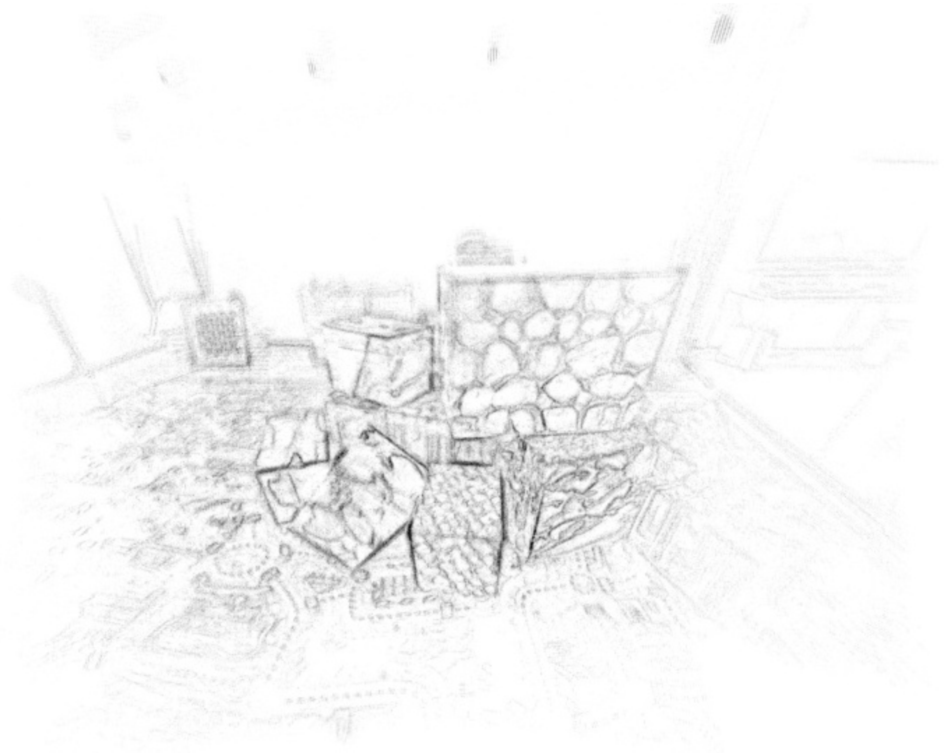


Figure 3.10: The dataset *boxes_rotation*. The motion in this dataset is rotation-dominated

Chapter 4

A Discussion to 6DoF Motion Estimation in General 3D Scenes

For general 3D scenes with complex scene structure, we may consider dividing the scene into multiple patches, and assume each patch is a plane.

Without knowing which parts of the scene correspond to real planes in space, we may simply divide the image into 4×3 regular grids, assign each event to the grid it belongs with a different set of parameters (2 for plane normal and 1 for depth), and optimize the frame with one single motion model. Accounting for the scale ambiguity, this amount to $(3 \times 12 + 5 =)41$ parameters for each *per-frame* optimization. The scale ambiguity is solved by parametrizing the linear velocity with 2 parameters $(\phi, \psi) \in \mathbb{S}^2$ indicating its direction, the same as the parametrization of a plane normal. This parametrization can not describe zero velocity; however, for the hand-held datasets we tested on this is not a problem. normalize & fix one depth value? Note that we don't need to optimize each patch individually and sum the costs together, since the intensity computation eq. (2.1) naturally sums all the events together so that we only need one cost function for one image. Also, the regions with fewer events have more pixels with zero intensity and thus have lower contribution to the cost function eq. (2.3), so no extra weighting is needed.

In fig. 4.1, we compare the synthesized images before and after motion correction of some selective images in various sequences. After motion correction, the edges appear much sharper.

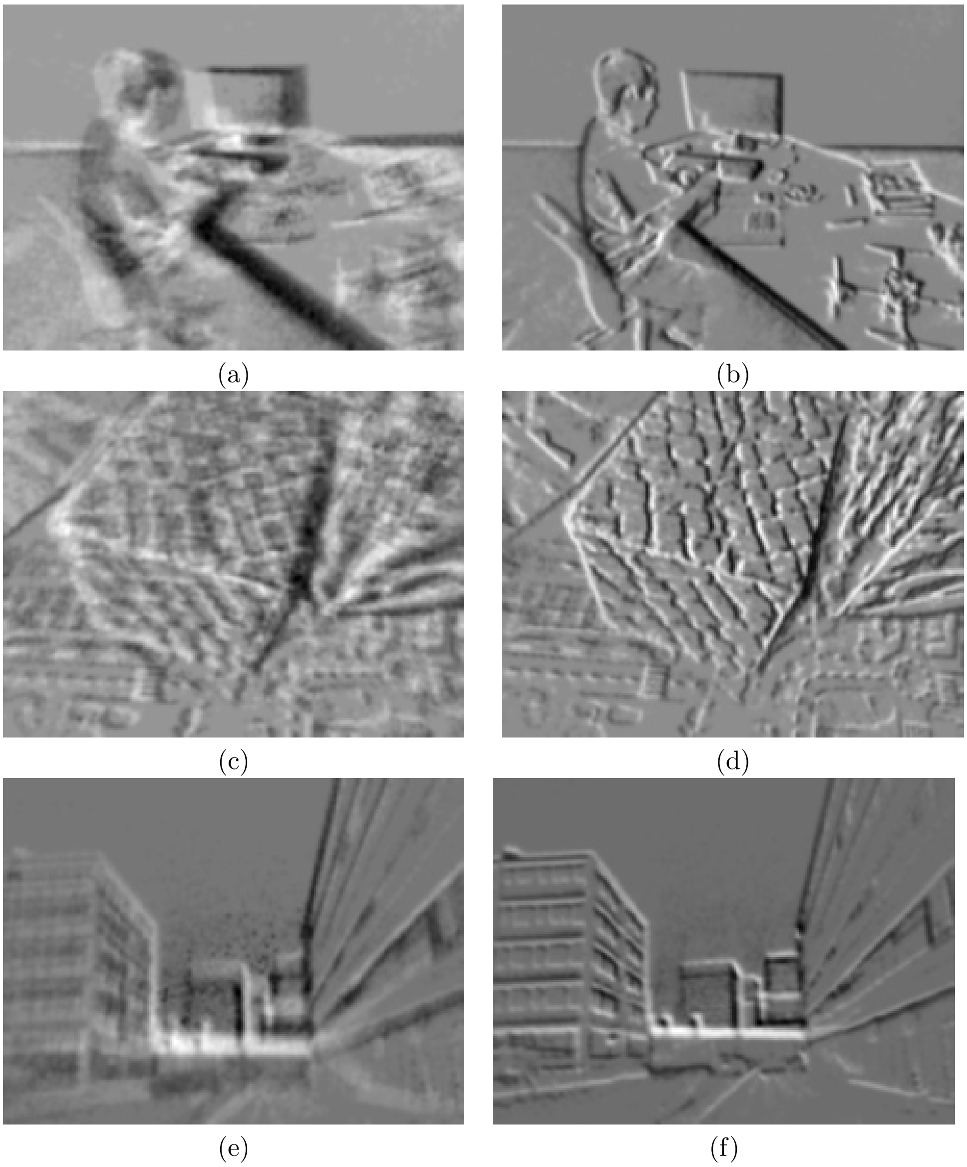
Since each frame has a different scale, without a global map as in section 3.1, it's hard to measure the accuracy of the linear velocity estimation. We first give the estimation of angular velocity in chapter 4. Comparing line 1 with line 2, line 3 and line 4, we can see that for general 3D scenes, planar patches assumption delivers better result than assuming the whole scene as one single plane. Notice that planar patches assumption for general scene has a similar error magnitude as planar assumption for planar scenes (line 6 and 8). However, the angular velocity estimated by either planar or planar patches assumption can not achieve the accuracy when estimating scenes with rotation dominated movements (line 5 and 7).

Without known scale for each frames, we can compare the angles between ground truth and estimated linear velocities. It turns out that for either planar scenes or planar patches assumption for general scenes, the mean angle is around 50 °, which is much larger than that of angular velocity, which is around 20 °. We guess that the window size is not large enough to provide a sufficient baseline for linear velocity estimation. But the window size can also not be chosen too large, otherwise the constant velocity assumption would fail. We already see in **??** that the inaccuracy problem can be solved by building a map and track on the map over a distance. For general scenes —

No.	Dataset	Median Orientation Error (°)			Maximal Velocity	Parametrization
		pitch	yaw	roll	(°/s)	
1	dynamic_6dof	20.88	21.60	18.08	509.37	6DoF, 12 patches
2	boxes_6dof	34.69	27.49	20.21	428.19	6DoF, 12 patches
3	boxes_6dof	60.16	31.33	21.11	428.19	6DoF, planar assumption
4	boxes_rotation	9.39	9.84	12.55	679.78	3DoF (angular velocity)
5	poster_6dof	32.23	31.55	22.37	905.58	6DoF, planar assumption
6	poster_rotation	11.62	8.47	13.23	934.88	3DoF (angular velocity)
7	poster_rotation	36.45	27.00	26.14	905.58	6DoF, planar assumption

Table 4.1: Comparison between planar assumption, planar patches assumption and rotation only estimation. The maximal velocity computes the maximal velocities among all three axes

tried initialize with multiple frames, didn't work very well. similarly sliding window didn't work; too many events didn't work



(a) (b) (c) (d) (e) (f)

Figure 4.1: Test results on 3 sequences with 6DoF motion. *Left*: Accumulated events without motion compensation. *Right:* Images after motion compensation via dividing scene into 12 regular grids. (a)(b) *dynamic_6dof* sequence, with ground truth velocities $\boldsymbol{\omega} = (-0.145; -3.761; 1.925)^{\mathrm{rad}}/_{\mathrm{s}}, \boldsymbol{v} = (0.313; 0.855; 0.326)^{\mathrm{m}}/_{\mathrm{s}}$. (c)(d) *boxes_6dof* sequence, with ground truth velocities $\boldsymbol{\omega} = (-1.357; 0.981; 4.577)^{\mathrm{rad}}/_{\mathrm{s}}, \boldsymbol{v} = (2.805; 0.538; -0.676)^{\mathrm{m}}/_{\mathrm{s}}$. (e)(f) *outdoors_running* sequence, with no ground truth information available.

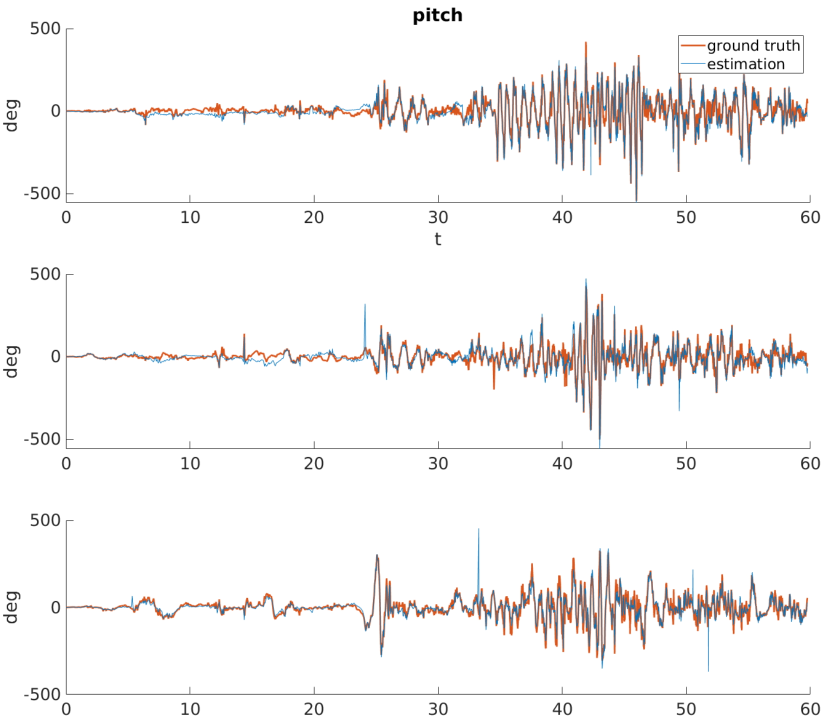


Figure 4.2: boxes 6dof rotation

Chapter 5

Conclusion and Outlook

We found that the calibration matrix provided in [21] is not very accurate, edges close to the upper left corner of the image always appear curved. We checked their provided calibration dataset, and found that the checkerboard has barely any corners in the upper left part of the image, which should be the reason why the undistortion in that area is not working well. This might be one of the causes why sometimes the image-to-model matching is not performing well. We also tried to cropped part of the image, but the resolution of DAVIS is relatively low (240×180), cropping reduces the amount of information and does not work better.
Usually the maps of *shapes* and *poster* sequences are done after 6 and 10 keyframes, respectively; as the number of keyframes increases, the size of the parameter set also increases (116 parameters in the bundle adjustment phase with 10 keyframes), which makes the problem hard to optimize. For a least square optimizing problem, which is often encountered in visual odometry, the Levenberg-Marquardt algorithm[24] is a good iteration method. However, the cost function eq. (2.3) used in this work is a more general non-linear function, and thus harder to optimize. The Nelder-Mead Simplex algorithm[25] used in bundle adjustment phase is better suitable for problems with small parameter size, and the choice of the initial simplex size is important. In the per-frame optimization and frame-to-model matching phase we make use of the derivatives; both BFGS and conjugate gradient approximate the objective function by a quadratic hypersurface, thus work well only when the evaluation point is close enough to the optimum. For a more reliable tracking over a longer distance, we should consider if better strategies are available, for example optimize only a subset of the keyframes that are most important for the current frame or most recent, or try multiple starting points.
DVS: 130 dB (1 lux**); APS: 55 dB

Bibliography

[1] P. Lichtsteiner, C. Posch, and T. Delbruck, “A 128× 128 120 db 15 μs latency asynchronous temporal contrast vision sensor,” *IEEE journal of solid-state circuits*, vol. 43, no. 2, pp. 566–576, 2008.

[2] C. Brandli, R. Berner, M. Yang, S.-C. Liu, and T. Delbruck, “A 240× 180 130 db 3 μs latency global shutter spatiotemporal vision sensor,” *IEEE Journal of Solid-State Circuits*, vol. 49, no. 10, pp. 2333–2341, 2014.

[3] C. Li, C. Brandli, R. Berner, H. Liu, M. Yang, S.-C. Liu, and T. Delbruck, “Design of an rgbw color vga rolling and global shutter dynamic and active-pixel vision sensor,” in *Circuits and Systems (ISCAS), 2015 IEEE International Symposium on*. IEEE, 2015, pp. 718–721.

[4] D. P. Moeys, F. Corradi, C. Li, S. A. Bamford, L. Longinotti, F. F. Voigt, S. Berry, G. Taverni, F. Helmchen, and T. Delbruck, “A sensitive dynamic and active pixel vision sensor for color or neural imaging applications,” *IEEE transactions on biomedical circuits and systems*, vol. 12, no. 1, pp. 123–136, 2018.

[5] B. Son, Y. Suh, S. Kim, H. Jung, J.-S. Kim, C. Shin, K. Park, K. Lee, J. Park, J. Woo *et al.*, “A 640× 480 dynamic vision sensor with a 9μm pixel and 300meps address-event representation,” in *Solid-State Circuits Conference (ISSCC), 2017 IEEE International*. IEEE, 2017, pp. 66–67.

[6] A. Z. Zhu, N. Atanasov, and K. Daniilidis, “Event-based feature tracking with probabilistic data association,” in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 4465–4470.

[7] C. Harris and M. Stephens, “A combined corner and edge detector.” in *Alvey vision conference*, vol. 15, no. 50. Citeseer, 1988, pp. 10–5244.

[8] D. Tedaldi, G. Gallego, E. Mueggler, and D. Scaramuzza, “Feature detection and tracking with the dynamic and active-pixel vision sensor (davis),” in *Event-based Control, Communication, and Signal Processing (EBCCSP), 2016 Second International Conference on*. IEEE, 2016, pp. 1–7.

[9] E. Mueggler, C. Bartolozzi, and D. Scaramuzza, “Fast event-based corner detection,” in *28th British Machine Vision Conference (BMVC)*, 2017.

[10] H. Rebecq, G. Gallego, and D. Scaramuzza, “Emvs: Event-based multi-view stereo,” in *British Machine Vision Conference (BMVC)*, no. EPFL-CONF-221504, 2016.

[11] G. Gallego, J. E. Lund, E. Mueggler, H. Rebecq, T. Delbruck, and D. Scaramuzza, “Event-based, 6-dof camera tracking from photometric depth maps,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[12] H. Rebecq, T. Horstschaefer, and D. Scaramuzza, "Real-time visualinertial odometry for event cameras using keyframe-based nonlinear optimization," in *British Machine Vis. Conf.(BMVC)*, vol. 3, 2017.

[13] H. Kim, S. Leutenegger, and A. J. Davison, "Real-time 3d reconstruction and 6-dof tracking with an event camera," in *European Conference on Computer Vision*. Springer, 2016, pp. 349–364.

[14] H. Rebecq, T. Horstschaefer, G. Gallego, and D. Scaramuzza, "Evo: A geometric approach to event-based 6-dof parallel tracking and mapping in real time," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 593–600, 2017.

[15] G. Orchard, C. Meyer, R. Etienne-Cummings, C. Posch, N. Thakor, and R. Benosman, "Hfirst: a temporal approach to object recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 10, pp. 2028–2040, 2015.

[16] A. I. Maqueda, A. Loquercio, G. Gallego, N. Garcıa, and D. Scaramuzza, "Event-based vision meets deep learning on steering prediction for self-driving cars," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5419–5427.

[17] A. Z. Zhu, L. Yuan, K. Chaney, and K. Daniilidis, "Ev-flownet: Self-supervised optical flow estimation for event-based cameras," *arXiv preprint arXiv:1802.06898*, 2018.

[18] G. Gallego and D. Scaramuzza, "Accurate angular velocity estimation with an event camera," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 632–639, 2017.

[19] 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 Int. Conf. Comput. Vis. Pattern Recog.(CVPR)*, vol. 1, 2018.

[20] A. A. Michelson, *Studies in optics*. Courier Corporation, 1995.

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

[22] B. Gough, *GNU scientific library reference manual*. Network Theory Ltd., 2009.

[23] R. Fletcher, *Practical methods of optimization*. John Wiley & Sons, 2013.

[24] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, "Numerical recipes in c," *Cambridge University Press*, vol. 1, p. 3, 1988.

[25] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The computer journal*, vol. 7, no. 4, pp. 308–313, 1965.

Appendix A

Derivative of the Contrast Metric

Since we use bilinear voting to evaluate the *dirac delta*, the derivatives of the cost function eq. (2.3) can be analytically computed as

Let

$$\rho\left(\boldsymbol{x};\boldsymbol{\theta}\right)=\mathcal{I}\left(\boldsymbol{x};\boldsymbol{\theta}\right)-\mu\left(\mathcal{I}\left(\boldsymbol{x};\boldsymbol{\theta}\right)\right) \tag{A.1}$$

Then

$$\frac{\partial}{\partial\boldsymbol{\theta}}\operatorname{Var}\left(\mathcal{I}\left(\boldsymbol{x};\boldsymbol{\theta}\right)\right)=\frac{2}{\mid\Omega\mid}\int_{\Omega}\rho\left(\boldsymbol{x};\boldsymbol{\theta}\right)\frac{\partial\rho\left(\boldsymbol{x};\boldsymbol{\theta}\right)}{\partial\boldsymbol{\theta}}d\boldsymbol{x} \tag{A.2}$$

The derivatives of the warped image are

$$\frac{\partial\mathcal{I}\left(\boldsymbol{x};\boldsymbol{\theta}\right)}{\partial\boldsymbol{\theta}}=-\sum_{k=1}^{N}\pm_k\nabla\delta\left(\boldsymbol{x}-\boldsymbol{x}_k'\left(\boldsymbol{\theta}\right)\right)\frac{\partial\boldsymbol{x}_k'\left(\boldsymbol{\theta}\right)}{\partial\boldsymbol{\theta}}, \tag{A.3}$$

In the case of planar homography the events warping is computed as

$$\begin{aligned}
\boldsymbol{\theta}&=\left(\boldsymbol{\omega}^\top,\boldsymbol{v}^\top,\varphi,\psi\right)^\top\in\mathbb{R}^8 && \text{(A.4)}\\
\boldsymbol{x}_k'\left(\boldsymbol{\theta}\right)&=\begin{bmatrix}x_{im}' & y_{im}' & 1\end{bmatrix}^\top=\begin{bmatrix}x'/z' & y'/z' & 1\end{bmatrix}^\top && \text{(A.5)}\\
\bar{\boldsymbol{x}}_k'\left(\boldsymbol{\theta}\right)&=\begin{bmatrix}x' & y' & z'\end{bmatrix}^\top\\
&=\boldsymbol{P}\left(\boldsymbol{R}(t)^\top\left(\boldsymbol{I}+\boldsymbol{T}(t)\boldsymbol{n}^\top/d\right)\right)^{-1}\boldsymbol{x}_k\\
&=\boldsymbol{P}\left(\boldsymbol{I}+\boldsymbol{v}t\boldsymbol{n}^\top\right)^{-1}\boldsymbol{R}(t)\boldsymbol{x}_k, && \text{(A.6)}
\end{aligned}$$

where $\boldsymbol{P}$ is the projection matrix from the camera frame to the world frame or the map, which does not have to correspond with the camera matrix provided by the dataset. Also, for simplicity of notation we assume $d=1$, and denote $\boldsymbol{P}\left(\boldsymbol{I}+\boldsymbol{v}t\boldsymbol{n}^\top\right)^{-1}$ as $\boldsymbol{P}_{\boldsymbol{v}}$. Since t usually spans a small temporal window, we can simplify the derivative with respect to angular velocity as

$$\begin{aligned}
\boldsymbol{R}(t)&=\exp(\boldsymbol{\omega}^\wedge t)\approx\boldsymbol{I}+\boldsymbol{\omega}^\wedge t && \text{(A.7)}\\
\frac{\partial\bar{\boldsymbol{x}}_k'}{\partial\boldsymbol{\omega}}&=-\boldsymbol{P}_{\boldsymbol{v}}t\boldsymbol{x}_k^\wedge. && \text{(A.8)}
\end{aligned}$$

The above equation makes use of the equivalence $\boldsymbol{\omega} \times \boldsymbol{x}_k = -\boldsymbol{x}_k \times \boldsymbol{\omega}$.
The derivative with respect to the linear velocity is

$$\frac{\partial \bar{\boldsymbol{x}}'_k}{\partial \boldsymbol{v}} = -\frac{t\boldsymbol{n}^\top \boldsymbol{R}(t)\boldsymbol{x}_k}{\boldsymbol{n}^\top \boldsymbol{v}t + 1}\boldsymbol{P}_{\boldsymbol{v}} \tag{A.9}$$

$$eu \tag{A.10}$$

Similarly we have

$$\frac{\partial \bar{\boldsymbol{x}}'_k}{\partial \boldsymbol{n}} = -\frac{t\boldsymbol{v}^\top \boldsymbol{R}(t)\boldsymbol{x}_k}{\boldsymbol{n}^\top \boldsymbol{v}t + 1}\boldsymbol{P}_{\boldsymbol{v}} \tag{A.11}$$

and

$$\frac{\partial \bar{\boldsymbol{x}}'_k}{\partial (\varphi;\psi)} = \frac{\partial \bar{\boldsymbol{x}}'_k}{\partial \boldsymbol{n}}\frac{\partial \boldsymbol{n}}{\partial (\varphi;\psi)}, \tag{A.12}$$

with

$$\frac{\partial \boldsymbol{n}}{\partial (\varphi;\psi)} = \begin{bmatrix} -\sin\varphi\sin\psi & \cos\varphi\cos\psi \\ \cos\varphi\sin\psi & \sin\varphi\cos\psi \\ 0 & -\sin\psi \end{bmatrix} \tag{A.13}$$

With the above quantities

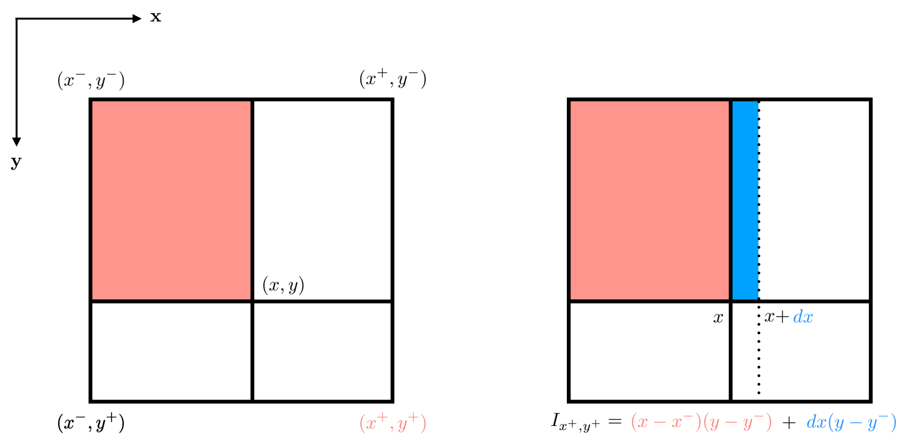
$$\frac{\partial \bar{\boldsymbol{x}}'_k}{\partial \boldsymbol{\theta}} = \begin{bmatrix} \frac{\partial \bar{\boldsymbol{x}}'_k}{\partial \boldsymbol{\omega}} & \frac{\partial \bar{\boldsymbol{x}}'_k}{\partial \boldsymbol{v}} & \frac{\partial \bar{\boldsymbol{x}}'_k}{\partial \varphi} & \frac{\partial \bar{\boldsymbol{x}}'_k}{\partial \psi} \end{bmatrix} \in \mathbb{R}^{3\times 8}$$

we can compute the gradient $\frac{\partial \boldsymbol{x}'_k}{\partial \boldsymbol{\theta}}$ as

$$\frac{\partial \boldsymbol{x}'_k}{\partial \boldsymbol{\theta}} = \frac{\partial \bar{\boldsymbol{x}}'_k}{\partial \boldsymbol{\theta}}\frac{1}{z'} - \bar{\boldsymbol{x}}'_k\frac{\partial z'}{\partial \boldsymbol{\theta}}\frac{1}{z'^2} \tag{A.14}$$

Theoretically, to evaluate the derivative of the cost function, the dirac delta $\nabla\delta(\boldsymbol{x})$ should be evaluated at each pixel location (240×180 for DAVIS). However, since we are using bilinear voting, a infinitesimal change $d\boldsymbol{x}$ at an event location $\boldsymbol{x}$ will only affect the derivative evaluated at the four neighboring pixels of $\boldsymbol{x}$. An illustration is shown in fig. A.1
The detailed algorithm for the evaluation of the Dirac delta as well as its derivative is shown in algorithm. 1. The behavior of the bilinear voting for derivatives is undefined when $\boldsymbol{x}'_k$ locates exactly at a pixel location. However, the possibility of this to happen is zero except when $t = 0$, and can be solved by simply discarding the event or disturbing with a small random noise. When projecting back to the map, The author uses a project matrix with different focal lengths than the ones in the camera calibration matrix, so that there won't be event at integer pixel locations after warping.

By linearity of the mean and the derivative, gaussian smoothing
However, a general nonlinear optimization is usually very hard, especially at bundle adjustment stage he jacobian matrix only measures the local grad



(a) The intensity at a pixel location is the area of the rectangle spanned by the opposite pixel and the event location

(b) Intensity change after an infinitesimal movement of the event

Figure A.1: Bilear voting

Algorithm 1: Bilinear Voting

```
Input  : A warped event $e'_k = \{x'_k, y'_k, t_k\}$
Output: The Dirac delta $\delta(\boldsymbol{x} - \boldsymbol{x}'_k)$ as well as its derivative $\frac{\partial\delta(\boldsymbol{x}-\boldsymbol{x}'_k)}{\partial\boldsymbol{x}'_k}$

Let
$x^- = \lfloor x'_k \rfloor, x^+ = \lceil x'_k \rceil, y^- = \lfloor y'_k \rfloor, y^+ = \lceil y'_k \rceil$
Then
$\delta(x^- - x'_k, y^- - y'_k) = (x - x^+)(y - y^+)$
$\delta(x^- - x'_k, y^+ - y'_k) = -(x - x^+)(y - y^-)$
$\delta(x^+ - x'_k, y^- - y'_k) = -(x - x^-)(y - y^+)$
$\delta(x^+ - x'_k, y^+ - y'_k) = (x - x^-)(y - y^-)$
$\frac{\partial\delta(x^- - x'_k, y^- - y'_k)}{\partial x'_k} = y - y^+, \frac{\partial\delta(x^- - x'_k, y^- - y'_k)}{\partial y'_k} = x - x^+$
$\frac{\partial\delta(x^- - x'_k, y^+ - y'_k)}{\partial x'_k} = -(y - y^-), \frac{\partial\delta(x^- - x'_k, y^+ - y'_k)}{\partial y'_k} = -(x - x^+)$
$\frac{\partial\delta(x^+ - x'_k, y^- - y'_k)}{\partial x'_k} = -(y - y^+), \frac{\partial\delta(x^+ - x'_k, y^- - y'_k)}{\partial y'_k} = -(x - x^-)$
$\frac{\partial\delta(x^+ - x'_k, y^+ - y'_k)}{\partial x'_k} = y - y^-, \frac{\partial\delta(x^+ - x'_k, y^+ - y'_k)}{\partial y'_k} = x - x^-$
```