Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea Magistrale in Matematica

DENSE OPTICAL FLOW ESTIMATION FOR EVENT CAMERAS: Application to parcel sorting conveyors.

Tesi di Laurea in Applied Inverse Problems in Imaging

Relatore:
Chiar.mo Prof.
ALESSANDRO LANZA

Presentata da:
SABRINA RICCI

Correlatore:
Dott. Ing.
MARTINO ALESSANDRINI

Anno Accademico 2023-2024

Introduction

Event cameras are bio-inspired visor sensors that react to changes in brightness in the scene, providing an output with high dynamic range and minimal blur at high temporal resolution. Thanks to his many advantages, one of the main applications of this camera is the estimation of optical flow. This thesis aims to estimate a dense optical flow map to infer depth.
Many studies are being carried out about this subject, we will refer to the state-of-the-art article [1], applying a few changes. The goal is to provide more specific mathematical reasoning and show the algorithms used. Moreover, this theory will be applied to a particular scenario: the estimation of the height of a parcel moving on a conveyor with fixed and known velocity.

In Chapter 1, we are going to explore the mathematical model of the event camera and its method of work. Then we're going to consider the optical flow estimation problem with a focus on the most used technique of contrast maximization. We will present our applied problem, analyzing the linear transformation to calibrate a camera with respect to the world frame.

In Chapter 2 we are going to present the first method we have tested and the reasons why it hasn't worked. After a brief introduction to variational models we study the multi-reference objective function proposed in [1] and the computation of its gradient.

In Chapter 3 we will analyze the three algorithms used: Steepest Descent, (Accelerated) Proximal Gradient, and Alternating Direction Method of Multipliers combined into a convergence analysis.

In Chapter 4 we will apply the algorithms presented in the previous chap-

ter to some synthetic data at first and then to two datasets of real data.

This project is mostly been developed during the internship at the company Datalogic S.p.A., which has been providing the Python initial codes and the datasets of real data.

Introduzione

Le camere ad eventi sono dei sensori di visione che reagiscono ai cambiamenti della luminoscità nella scena generando un output caratterizzato da un elevato intervallo dinamico e da una sfocatura minima anche con alta risoluzione temporale. Grazie a questi numerosi vantaggi, una delle principali applicazioni della camera ad eventi è il flusso ottico.
L'obiettivo di questa tesi riguarda la stima di un flusso ottico denso dipendente dalla posizione del pixel per poi ricavare la profondità.
Su queste tematiche sono stati svolti numerosi studi, nello specifico in questo progetto faremo riferimento allo stato dell'arte [1], con l'aggiunta di alcune modifiche. Lo scopo è quello di fornire maggiori spiegazioni matematiche e mostrare nello specifico i diversi algoritmi applicati. In particolare testeremo questi risultati a una specifica applicazione che riguarda il passaggio di un pacco su un conveyor con velocità fissata e nota.

Nel Capitolo 1 andremo ad approfondire il modello matematico e il funzionamento di una camera ad eventi. Sucessivamente considereremo la risoluzione del problema riguardante la stima del flusso ottico, concentrandoci sulla tecnica maggiormente usata: il *contrast maximization* degli eventi. Presenteremo poi il nostro specifico caso di applicazione del metodo, analizzando le trasformazioni lineari per calibrare la camera rispetto il ristema di riferimento del conveyor.

Nel Capitolo 2 presenteremo il primo modello testato e i motivi per cui non è risultato corretto. Dopo una veloce introduzione sui metodi variazionali, studieremo la funzione obiettivo *multi-reference focus* proposta da [1],

soffermandoci sul calcolo del suo gradiente.

Nel Capitolo 3 analizzeremo i tre algoritmi proposti: il metodo di discesa gradiente, del gradiente prossimale accelerato e non e dei moltiplicatori con direzione alternata uniti allo studio sulla loro analisi di convergenza.

Nel Capitolo 4 applicheremo gli algoritmi proposti nel precedente capitolo, prima ad alcuni esempi con i dati sintetici e poi a due dataset di dati reali.

Questo progetto nasce durante il percorso di tirocinio svolto presso l'azienda Datalogic S.p.A. che ha reso disponibile il codice Python iniziale e il dataset dei dati reali.

Contents

Chapter 1

Optical flow estimation by event camera

In 1991 C. Mead and M. Mahowald published an article in the Scientific American where they presented the new 'Silicon Retina' [2]. Here appeared also the first image taken with the one that after will be called 'Event Camera'.
This new camera was inspired by the neural architecture of the eye, which defined the start of the field of neuromorphic engineering.
Event cameras had many advantages over conventional frame cameras but have become commercially available only since 2008. Now their potential, in particular for high-speed and high dynamic range scenarios is understood. Still, the difference in operating from frame cameras brings the necessity to develop new methods to process their output.

1.1 Event Camera

The **Event Camera** (or **silicon retina neuromorphic camera** or **dynamic vision sensor**) responds to brightness changes in the scene asynchronously and independently for every pixel.
The camera collects a sequence of events (or spikes), each of them containing

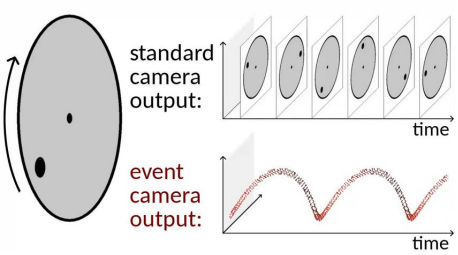

Figure 1.1: Comparison between frame camera and event camera regarding the capture of the scene (Rebecq, Gallego, Mueggler, Scaramuzza [3]).

the pixel location, the time stamp, and the change of brightness.
When an event occurs in a pixel, it memorizes the brightness and keeps monitoring the scene's luminosity waiting for a sufficiently big change. When this happens, another event is generated.
Events cameras are data-driven sensors, that is the output depends on the amount of motion in the scene, then generation per second of the events is proportional to the velocity of the motion. The sensor quickly reacts to the visual stimuli: the events are timestamped with microsecond resolution and sub-millisecond latency.
The incident light at a pixel is the result of scene illumination and surface reflectance. Usually, illumination is assumed to be constant then the changes in the brightness are caused by reflectance changes which are mostly the result of the movements of objects in the scene.

Advantages

The main difference between the event camera and the standard frame camera is the output (Figure 1.1). In the first case, the full image is acquired at a fixed frame rate (commonly expressed in frames per second or fps) while the second one stores just an event in correspondence with brightness changes. This implies saving memory for the event camera. In particular, there are many advantages:

- *High temporal resolution.* The camera can detect very fast motions

thanks to the quick monitor of brightness changes: events are caught and stamped with microsecond resolution. For this reason, it is less sensitive to motion blur.

- *Low latency.* The latency of a system is the delay between the time at which the input is sent and the time at which the output is registered, it measures the velocity of the response. In event cameras each pixel works independently, as a consequence, as soon as the event is detected, it is transmitted. It has a latency from $10\mu s$ to $10^{-3}s$ in the real world.

- *Low power consumption.* Redundant data are removed, then the power can be used only for brightness changes.

- *Hight dynamic range.* The dynamic range is the ratio between the smallest and the largest signal value. It is greater than 120 dB, indeed pixel detection works in very dark as well as very bright environments. This happens because events are detected with a logarithmic scale of intensity.

On the other hand, there are also some disadvantages:

- Events cameras are expensive (about 10k€)

- Events are asynchronous and sparse while images are synchronous and dense, so standard imaging methods cannot be applied to event cameras.

- The detection of events depends on the brightness and also on the relative motion between the scene and the camera.

- Event cameras are noisier than other vision sensors.

Summing up, event cameras in some aspects are already much more convenient than frame ones, moreover, finding methods and algorithms to resolve the previous challenges would improve even more the obtained results.

1.1.1 Data acquisition model

Definition 1.1.1. We define the **brightness** as $L := log(I)$ where I represents the intensity.

The output of an event camera is a collection of **events** $\mathcal{E} = \{e_k\}_{k=1}^{N_e}$ with $e_k := \{\boldsymbol{x}_k, t_k, p_k\}$ $\forall k = 1, ...N_e$. In particular:

- $\boldsymbol{x}_k = (i_k, j_k)$ represents the pixel location,
- t_k is the time at which the event happens,
- $p_k := \text{sgn}\left(\frac{\partial I}{\partial t}(\boldsymbol{x}_k, t_k)\right) \in \{+1, -1\}$ is the polarity which is defined as the sign of the brightness change compared to the brightness of the previous event in the same location.

We consider a temporal threshold $C > 0$, called **contrast sensitivity**, which is fixed, usually between 10 to 50 percent illumination change. The change in brightness at the pixel location $\boldsymbol{x}$ is represented as

$$\Delta L(\boldsymbol{x}_k, t_k) = |L(\boldsymbol{x}_k, t_k) - L(\boldsymbol{x}_k, t_k - \Delta t_k)| > p_k C$$

where Δt_k is the time elapsed since the last event in $\boldsymbol{x}_k$. A new event is generated when $|\Delta L(\boldsymbol{x}_k, t_k)| > C$ (Figure 1.2).

For small Δt_k the brightness increment can be approximated using Taylor's expansion:

$$\begin{aligned} & \Delta L(\boldsymbol{x}_k, t_k) \approx \frac{\partial L}{\partial t}(\boldsymbol{x}_k, t_k)\Delta t_k \\ \Rightarrow \quad & \frac{\partial L}{\partial t}(\boldsymbol{x}_k, t_k) \approx \frac{\Delta L(\boldsymbol{x}_k, t_k)}{\Delta t_k} = \frac{p_k C}{\Delta t_k}. \end{aligned} \tag{1.1}$$

This representation considers the temporal derivative of L and represents an indirect way of measuring brightness.

Proposition 1.1.2. *Assuming constant illumination, events are generated by moving edges.*

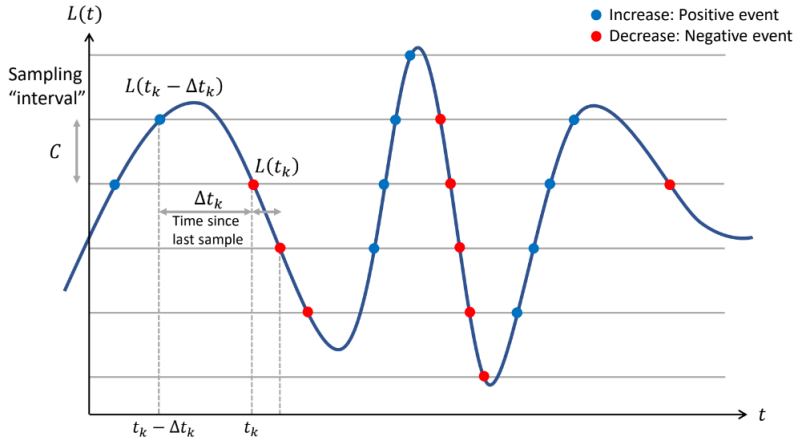

Figure 1.2: Graphical representation of the model of event generation. Every time the brightness changes its value more than a fixed threshold C a new event is generated with positive (if brightness is increasing) or negative (if brightness is decreasing) polarity (G. Gallego [4]).

Proof. Suppose that the pixel location $\boldsymbol{x}$ is also a function of t. By the assumption of constant brightness, we get the so-called *optical flow constraint equation*:

$$\frac{\partial L}{\partial t}(\boldsymbol{x}(t), t) + \nabla_{\boldsymbol{x}} L(\boldsymbol{x}(t), t) \cdot \dot{\boldsymbol{x}}(t) = 0.$$

Now we set $\dot{\boldsymbol{x}} = \mathbf{v}$ and using (1.1) we obtain:

$$\Delta L(\boldsymbol{x}(t), t) \approx -\nabla_{\boldsymbol{x}} L(\boldsymbol{x}(t), t) \cdot \mathbf{v} \Delta t = -\nabla_{\boldsymbol{x}} L(\boldsymbol{x}(t), t) \cdot \Delta \boldsymbol{x}.$$

This is equivalent to saying that, for a small increment in time, the intensity increment is caused by $-\nabla_{\boldsymbol{x}} L(\boldsymbol{x}(t), t)$ moving with velocity $\mathbf{v}$, over a displacement $\Delta \boldsymbol{x}$. □

1.2 Optical flow estimation and related work

Events-based cameras are suitable for estimating optical flow.

Definition 1.2.1. We define **optical flow** as the distribution of the apparent velocities of objects caused by the relative motion between an observer and the scene.

Optical flow estimation is the problem of computing the velocity of objects without having any information about the scene motion.
In standard cameras, it is measured by comparing consecutive frames. In general, the more a moving object is closer to the camera, the more it will display apparent motion.

On the other hand, one single event does not have enough information to estimate optical flow, therefore, they have to be aggregated.
In this case, event cameras are optimal to determine the flow because, as we have seen in Proposition 1.1.2, events represent moving edges, where the flow is less ambiguous. Moreover, their time information allows high-speed flow measurements.

The main problem of optical flow estimation is the expensive computational cost: instead of determining the flow field over the whole space it is preferable to determine it only at certain points, which usually are the events' locations.

Related work

Since optical flow estimation has many advantages in event cameras, much research has been carried out. At first, has been proposed the adaptation of frame-based approaches (block matching [5] and Lucas-Kanade [6]), filter-banks [7], [8], spatio-temporal plane-fitting [9], [10], time surface matching [11], variational optimization on voxelized events [12] and feature-based contrast maximization [13], [14]. In [15] is presented a detailed survey.
The current state-of-the-art approach is artifical neural networks in [16], [17], [18], [19]-[21], mostly inspired by frame-based optical flow architectures [22], [23]. If spiking-based approaches are not used, the input signal has to be adapted and the events have to be converted into a tensor representation. These learning-based methods can be classified into supervised, semi-supervised, and unsupervised. The three most common in terms of architectures are U-Net [18], [24], FireNet [25], and RAFT [19], [26].
Supervised methods train ANNs in simulations and real data: [19], [24], [26]-

[30]. To match the space-time resolution of the event cameras are required GT flow. This technique works well in simulations, while there are some issues with real-world data sets. Indeed because of the large domain gap between training and test data, there is a performance gap when trained models are used to predict flow on real data ([28], [31]).
To partially solve this problem, semi-supervised methods use grayscale images from a colocated camera, for instance, DAVIS camera [32], as a supervisory signal. Images are warped using the flow predicted by the ANN and their photometric consistency is used as a loss function ([18], [20], [21]). This supervisory signal can suffer from the limitations of frame-based cameras, such as motion blur and low dynamic range, and can affect the trained ANNs. EV-FlowNet [18] pioneered these approaches.
Unsupervised methods only use event data. The flow predicted by the artificial neural network is used to compute an event alignment error that represents the loss function ([16], [25], [17], [33]-[35]). Zhu et al. [16] extended EV-FlowNet [18] to the unsupervised setting using a motion-compensation loss inspired by the average timestamp image in [36]. This U-Net-like approach has been improved with recurrent blocks in [25], [17]. A FireFlowNet, a lightweight recurrent ANN with no downsampling, has been proposed in [25]. More recently, in [17] many variants of EV-FlowNet and Fire-FlowNet models are considered. Thanks to the recurrent blocks, sequentially processing short-time event frames replaces the usual voxel-grid input event representation. Finally, recurrent work [35] builds upon [17], proposing iterative event warping at multiple reference times in a multi-timescale fashion, which allows curved motion trajectories.

Contrast Maximization

The main technique used in event cameras to estimate the optical flow is contrast maximization (proposed in [13]).
Consider an image of dimensions $h \times w$ represented by a set of N_p pixels $\{(i,j)\}_{i,j=1}^{h,w}$. Given a specific spatiotemporal region of interest, we select a

set of events $\mathcal{E} = \{e_k\}_{k=1}^{N_e}$ with $e_k = \{\boldsymbol{x}_k, t_k, p_k\}$, $\forall k = 1, ..N_e$.
Under the assumption of events generated by moving edges and flow locally constant, we transform the events geometrically according to a motion model $\mathbf{W}$. Indeed corresponding events are triggered by the same edge, therefore they lie on the same trajectory. After selecting a reference time t_{ref}, we warp the events according to these trajectories.
We obtain another set of events $\mathcal{E}' = \{e'_k\}_{k=1}^{N_e}$ such that

$$e_k = \{\boldsymbol{x}_k, t_k, p_k\} \;\mapsto\; e'_k = \{\boldsymbol{x}'_k, t_{ref}, p_k\}$$

where

$$x'_k = \mathbf{W}(\boldsymbol{x}_k, t_k, \mathbf{v})$$

according to a candidate velocity $\mathbf{v}$.

Definition 1.2.2. After having obtained the set of warped events $\mathcal{E}'$, we can construct a **image of warped events** (**IWE**):

$$H(\boldsymbol{x}; \mathbf{v}) = \sum_{k=1}^{N_e} b_k \delta(\boldsymbol{x} - \boldsymbol{x}'_k)$$

where each pixel $\boldsymbol{x}$ sums the b_k of the warped events $\boldsymbol{x}'_k$ falling within it.

- if $b_k = p_k$ then we sum the polarities of the events along the trajectories
- if $b_k = 1$ then we sum the number of warped events along the trajectories.

We'll be using $b_k = 1$ throughout the whole project.

The Dirac delta δ is not differentiable, then it is approximated by a Gaussian distribution:

$$\delta(\boldsymbol{x} - \boldsymbol{x}'_k) \approx \mathcal{N}(\boldsymbol{x}; \boldsymbol{x}'_k, \sigma^2 \mathbf{I}).$$

Then

$$H(\mathbf{v}; (i,j)) = \sum_{k=1}^{N_e} \mathcal{N}((i,j); \boldsymbol{x}'_k, \sigma^2 \mathbf{I}_2) = \sum_{k=1}^{N_e} \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(i - i'_k)^2 + (j - j'_k)^2}{2\sigma^2}\right).$$

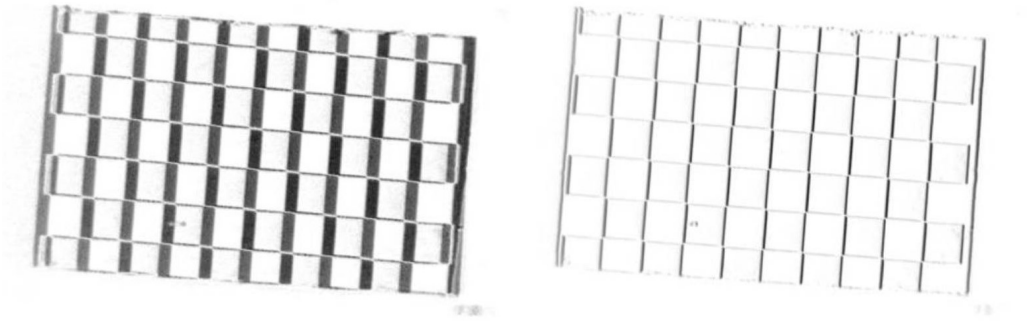

Figure 1.3: Representation of the events generated by the movement of a parcel on a conveyor. On the left the image of H before contrast maximization, and on the right the image of H after contrast maximization.

In this case, we obtain a differentiable function that is strictly related to the Euclidian distance and hence to the distance between events. Therefore it is ideal for optimization but it is very computationally expensive: the Gaussian contribution has to be computed at every pixel location. If we have an image of N_p pixels and N_e events we need N_pN_e operations for every function call.

The goal of the contrast maximization framework is to find a warp maximizing the alignment of the events caused by the same edges.
From Figure 1.3 we can see the difference before (on the left) and after (on the right) contrast maximization of a parcel moving on a conveyor. We can observe that in the first case, events are not aligned: the events generate a shadow in correspondence with the edges of the image. In the second case, to the previous set of events is applied a motion model with the velocity that maximizes the alignment of the events.

Dense optical flow

In particular, we are interested in the estimation of the optical flow that is not constant but depends on the pixel location of the image plane called a **dense optical flow**. In fact:

- the parcel can have a dimension that is not regular, for instance, it can

be higher on one side and lower on another one. Then, since the optical flow depends on the proximity to the camera, we obtain different values depending on the height.

- the camera could be not well calibrated and this implies different values of optical flow in each pixel.

Therefore we have:

$$x'_k = \mathbf{W}(\boldsymbol{x}_k, t_k, \mathbf{v}(\boldsymbol{x}))$$

where $\{\mathbf{v}(\boldsymbol{x}))\}_{\boldsymbol{x}=1,\dots N_p}$ is the flow field on the image plane at t_{ref}.

In our specific case, parcels move on a conveyor with fixed velocity. Then the trajectories followed by the events are straight and the motion model $\mathbf{W}$ can be approximated by translations:

$$\mathbf{W}(\boldsymbol{x}_k, t_k, \mathbf{v}(\boldsymbol{x})) = \boldsymbol{x}_k - (t_k - t_{ref})\mathbf{v}(\boldsymbol{x}) = \boldsymbol{x}_k - \Delta t_k \mathbf{v}(\boldsymbol{x})$$

where $\mathbf{v}(\boldsymbol{x})$ is a vector with two components representing vertical and horizontal direction, i.e. $\mathbf{v}(\boldsymbol{x}) = \begin{pmatrix} v^{(1)}(\boldsymbol{x}) \\ v^{(2)}(\boldsymbol{x}) \end{pmatrix}$.

1.3 Depth estimation for 3D model

1.3.1 Forward imaging model

To switch from a 3D model to a 2D model and vice-versa we consider the so-called **forward imaging model** [37] which is illustrated in Figure 1.4. We start from 3D world frame reference $\hat{x}_w\hat{y}_w\hat{z}_w$ and we consider a point P with coordinates $\mathbf{x}_w = (x_w, y_w, z_w)$.
The camera lies in this world frame and it has its own reference system $\hat{x}_c\hat{y}_c\hat{z}_c$ where the axis $\hat{z}_c$ is aligned with the optical axis of the camera. f is the focal length that is the distance between the image plane and the axis $\hat{y}_c$. If we assume to know the relative position $\mathbf{c}_w$ and the orientation between the camera frame and the world frame then we can obtain the coordinates of P projected into the image plane.

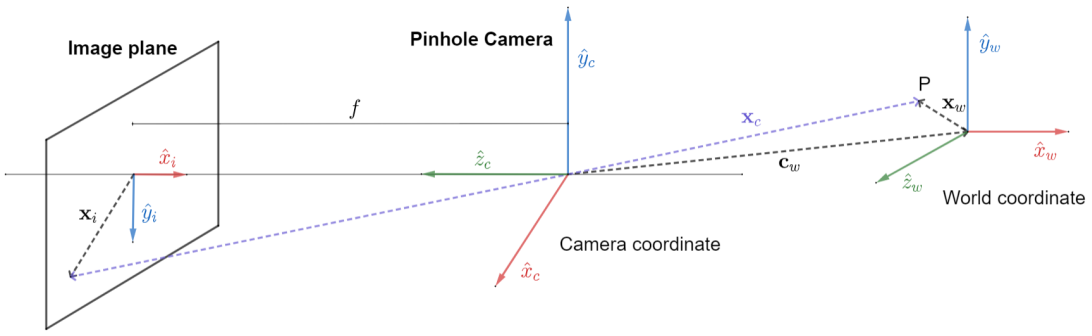

Figure 1.4: Representation of the forward imaging model of a point P in 3D world coordinates to 2D camera's image plane coordinate.

In particular, we can obtain the coordinates of the point P in the camera frame system $\mathbf{x}_c = (x_c, y_c, z_c)$ through a 3D to 3D coordinate transformation. Then we apply perspective projection to obtain the 2D image coordinates $\mathbf{x}_i = (x_i, y_i)$.

We at first assume to know the coordinates of P in the camera system, by the criterion of similar triangles,

$$\frac{x_i}{f} = \frac{x_c}{z_c} \Rightarrow x_i = f\frac{x_c}{z_c} \tag{1.2}$$

$$\frac{y_i}{f} = \frac{y_c}{z_c} \Rightarrow y_i = f\frac{y_c}{z_c}. \tag{1.3}$$

The image plane is defined in pixels, then we have to transform the coordinates from mm to pixels. We set m_x and m_y the pixel densities (i.e. $pixel/mm$) in x and y directions, respectively. Moreover, we don't know the position in which the optical axis pierces the image plane, say the coordinates (o_x, o_y). As a convention, the origin of the image is usually located at one of its corners. Then we obtain

$$x_i^{pix} = m_x x_i + o_x = m_x f\frac{x_c}{z_c} + o_x$$

$$y_i^{pix} = m_y y_i + o_y = m_y f\frac{y_c}{z_c} + o_y.$$

$(m_x f, m_y f, o_x, o_y)$ are said to be the **intrinsic parameters** of the camera.

We have obtained a non-linear system, therefore we transform it into a linear system through the homogeneous coordinates.

We write $\mathbf{x}^{pix} = (x^{pix}, y^{pix})$ in homogeneous coordinates $\tilde{\mathbf{x}}^{pix} = (\tilde{x}^{pix}, \tilde{y}^{pix}, \tilde{z}^{pix})$ such that $x^{pix} = \dfrac{\tilde{x}^{pix}}{\tilde{z}^{pix}}$ and $y^{pix} = \dfrac{\tilde{y}^{pix}}{\tilde{z}^{pix}}$ where $\tilde{z}^{pix} \neq 0$.
We set $\tilde{z}^{pix} = 1$, then

$$\tilde{\mathbf{x}}^{pix} = \begin{pmatrix} x_i^{pix} \\ y_i^{pix} \\ 1 \end{pmatrix} = \begin{pmatrix} \tilde{x}_i^{pix} \\ \tilde{y}_i^{pix} \\ \tilde{z}_i^{pix} \end{pmatrix} = \begin{pmatrix} z_c x_i^{pix} \\ z_c y_i^{pix} \\ z_c \end{pmatrix} = \begin{pmatrix} m_x f x_c + z_c o_x \\ m_y f y_c + z_c o_y \\ z_c \end{pmatrix}$$

$$= \begin{pmatrix} m_x f_x & 0 & o_x & 0 \\ 0 & m_y f_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_c \\ y_c \\ z_c \\ 1 \end{pmatrix} = \mathbf{M}_{int} \tilde{\mathbf{x}}_c. \tag{1.4}$$

The matrix $\mathbf{M}_{int}$ is called **intrinsic matrix** and it can be written as $\mathbf{M}_{int} = (\mathbf{K}|0)$ where $\mathbf{K}$ is the upper triangular **calibration matrix**.

Now, we consider the coordinate transformation from the world frame reference to the camera frame reference. We suppose to know the **extrinsic parameters** that are the position $\mathbf{c}_w$ and the rotation matrix $\mathbf{R}$ of the camera in the world frame and we consider the point P. We can find the vector $\mathbf{x}_c$ in the world coordinate frame as:

$$\mathbf{x}_c = \mathbf{R}(\mathbf{x}_w - \mathbf{c}_w) = \mathbf{R}\mathbf{x}_w + \mathbf{t}$$

where $\mathbf{t} = -\mathbf{R}\mathbf{c}_w$ is the translation vector.
As before we use homogeneous coordinates:

$$\tilde{\mathbf{x}}_c = \begin{pmatrix} x_c \\ y_c \\ z_c \\ 1 \end{pmatrix} = \left(\begin{array}{ccc|c} & \mathbf{R}_{3\times 3} & & \begin{matrix} t_x \\ t_y \\ t_z \end{matrix} \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix} = \mathbf{M}_{ext} \tilde{\mathbf{x}}_w. \tag{1.5}$$

We call the matrix $\mathbf{M}_{ext}$ **extrinsic matrix**.

To sum up, by combining equations (1.4) and (1.5) we obtain the following relation that allows us to transform 3D coordinates in the world frame to 2D coordinates in the image frame.

$$\tilde{\mathbf{x}}^{pix} = \mathbf{M}_{int}\mathbf{M}_{ext}\tilde{\mathbf{x}}_w = \mathbf{P}\tilde{\mathbf{x}}_w$$

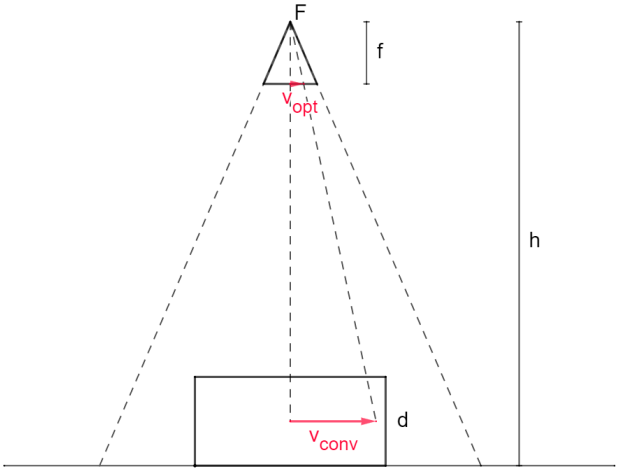

Figure 1.5: Graphical representation of a parcel with height d moving on the conveyor with velocity v_{conv}. On the top is represented the event-based camera with focal point F and focal length f. On the image plane is projected the velocity v_{opt} representing the optical flow. The distance between the camera and the conveyor is h.

where $\mathbf{P}$ is called **projection matrix**.
If we know the image coordinates and we want to recover the world coordinates, then

$$\tilde{\mathbf{x}}_w = \mathbf{P}^{-1}\tilde{\mathbf{x}}^{pix}.$$

1.3.2 3D model of interest

In this project, we have to estimate the depth of a parcel in the scenario illustrated in Figure 1.5. We consider a conveyor that sorts parcels depending on their detected bar code, moving with a fixed known velocity. An event camera is placed stationary over the conveyor. We have the following data:

- $h = 1.4m$ the distance between the focal point F of the camera and the conveyor;
- $f = 5 \cdot 10^{-3}m$ the focal length of the camera;

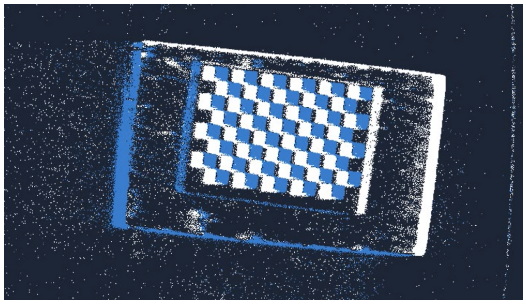

Figure 1.6: Representation of the events collected by the camera during the passage of a parcel at a specific time stamp.

- $\mathbf{v}_{conv} = (v_{conv}^{(1)}, v_{conv}^{(2)}) = (1.5, 0) m/s$ the conveyor velocity;
- $p = 4.86 \cdot 10^{-6} m$ the dimension in m/s of one pixel.

We have to estimate the following unknown quantities:

- d the parcel's height to be estimated;
- $\mathbf{v}_{opt}^{p}$ the optical flow in $pixel/s$.

We assume that the world frame of the conveyor and the camera frame are aligned and that the second component of the velocity is null. In this case, the depth estimation is simplified.
Note that $\mathbf{v}_{opt}$ is expressed in $pixel/s$, then to transform it in meters we multiply it by the pixel dimension:

$$\mathbf{v}_{opt} = \mathbf{v}_{opt}^{p} p$$

By relation (1.2), (1.3) we obtain

$$\frac{\mathbf{v}_{opt}}{\mathbf{v}_{conv}} = \frac{f}{h_{est}} \quad \Longrightarrow \quad h_{est} = \frac{\mathbf{v}_{conv}}{\mathbf{v}_{opt}} f$$

where h_{est} is the unknown distance between the top of the parcel and F.
From here we can easily obtain the following:

$$d = h - h_{est}. \tag{1.6}$$

Chapter 2

The considered estimation model

2.1 First (unsuccessful) attempt

The first idea is to use the pixel location of the events on the image plane at the reference time instant. In this way, we obtain a loss function to be minimized, defined as the variance of the geometric position of the warped pixel locations. Better estimation of $\mathbf{v}$ brings to a less dispersion of the x'_k around its centroid.
We consider the case with constant velocity $\mathbf{v}$.

In particular, given the warped events $\boldsymbol{x}'_k$ we compute the centroid:

$$\begin{aligned}
\mathbf{C}' &:= \frac{1}{N_e}\sum_{k=1}^{N_e}\boldsymbol{x}'_k = \frac{1}{N_e}\sum_{k=1}^{N_e}(\boldsymbol{x}_k - \Delta t_k \mathbf{v}) \\
&= \frac{1}{N_e}\sum_{k=1}^{N_e}\boldsymbol{x}_k - \frac{1}{N_e}\sum_{k=1}^{N_e}\Delta t_k \mathbf{v} \\
&= \mathbf{C} - \frac{\mathbf{v}}{N_e}\sum_{k=1}^{N_e}\Delta t_k \\
&= \begin{pmatrix} C^{(1)} - \dfrac{v^{(1)}}{N_e}\displaystyle\sum_{k=1}^{N_e}\Delta t_k \\ C^{(2)} - \dfrac{v^{(2)}}{N_e}\displaystyle\sum_{k=1}^{N_e}\Delta t_k \end{pmatrix}
\end{aligned}$$

where $\mathbf{C}$ is defined as the centroid of the events $e_k \; \forall k = 1, ..N_e$.

We define the loss as the variance of the geometric positions of the warped events:

$$\begin{aligned}
\mathcal{L}(\mathbf{v}) &= \frac{1}{N_e}\sum_{k=1}^{N_e} ||\boldsymbol{x}'_k - \mathbf{C}'||_2^2 \\
&= \frac{1}{N_e}\sum_{k=1}^{N_e} (\boldsymbol{x}'^{(1)}_k - \mathbf{C}'^{(1)})^2 + \frac{1}{N_e}\sum_{k=1}^{N_e} (\boldsymbol{x}'^{(2)}_k - \mathbf{C}'^{(2)})^2 \\
&= \frac{1}{N_e}\sum_{k=1}^{N_e} \left(\boldsymbol{x}^{(1)}_k - \Delta t_k \mathbf{v}^{(1)} - C^{(1)} + \frac{v^{(1)}}{N_e}\sum_{k=1}^{N_e} \Delta t_k \right)^2 + \\
&\quad + \frac{1}{N_e}\sum_{k=1}^{N_e} \left(\boldsymbol{x}^{(2)}_k - \Delta t_k \mathbf{v}^{(2)} - C^{(2)} + \frac{v^{(2)}}{N_e}\sum_{k=1}^{N_e} \Delta t_k \right)^2 .
\end{aligned}$$

We have to solve:

$$\mathbf{v} = \underset{\mathbf{v}}{\operatorname{argmin}}\, \mathcal{L}(\mathbf{v}).$$

$\mathcal{L}$ is quadratic and strongly convex, then it has a unique global minimizer. To obtain it, we impose first-order optimality conditions:

$$\nabla \mathcal{L}(v) = 0.$$

Let's compute the partial derivative:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial v^{(1)}} &= \frac{2}{N_e}\sum_{k=1}^{N_e} \left[\left(\boldsymbol{x}^{(1)}_k - \Delta t_k \mathbf{v}^{(1)} - C^{(1)} + \frac{v^{(1)}}{N_e}\sum_{k=1}^{N_e} \Delta t_k \right) \left(-\Delta t_k + \frac{1}{N_e}\sum_{k=1}^{N_e} \Delta t_k \right) \right] \\
&= \frac{2}{N_e} \left(v^{(1)} \sum_{k=1}^{N_e} \Delta t_k^2 - \frac{v^{(1)}}{N_e} \left(\sum_{k=1}^{N_e} \Delta t_k \right)^2 - \sum_{k=1}^{N_e} \Delta t_k x^{(1)}_k + \frac{1}{N_e} \sum_{k=1}^{N_e} \Delta t_k \sum_{k=1}^{N_e} x^{(1)}_k \right) .
\end{aligned}$$

With the same calculation, we obtain a similar result for $v^{(2)}$. We set both

equal to 0, obtaining a linear system with 2 equations and 2 unknowns:

$$v^{(1)} = \frac{\sum_{k=1}^{N_e} \Delta t_k x_k^{(1)} - \frac{1}{N_e} \sum_{k=1}^{N_e} \Delta t_k \sum_{k=1}^{N_e} x_k^{(1)}}{\sum_{k=1}^{N_e} \Delta t_k^2 - \frac{1}{N_e} \left(\sum_{k=1}^{N_e} \Delta t_k \right)^2} := \frac{b^{(1)}}{a}$$

$$v^{(2)} = \frac{\sum_{k=1}^{N_e} \Delta t_k x_k^{(2)} - \frac{1}{N_e} \sum_{k=1}^{N_e} \Delta t_k \sum_{k=1}^{N_e} x_k^{(2)}}{\sum_{k=1}^{N_e} \Delta t_k^2 - \frac{1}{N_e} \left(\sum_{k=1}^{N_e} \Delta t_k \right)^2} := \frac{b^{(2)}}{a}.$$

We observe that the denominator of the two velocity components is the same.

To obtain a more accurate result, we divide the events according to their polarity. We compute two different loss functions and we sum them:

$$\mathcal{L}(\mathbf{v}) = \mathcal{L}_{pos}(\mathbf{v}) + \mathcal{L}_{neg}(\mathbf{v}).$$

We obtain a linear system:

$$\begin{pmatrix} a_{pos} + a_{neg} & 0 \\ 0 & a_{pos} + a_{neg} \end{pmatrix} = \begin{pmatrix} v^{(1)} \\ v^{(2)} \end{pmatrix} \begin{pmatrix} b_{pos}^{(1)} + b_{neg}^{(1)} \\ b_{pos}^{(2)} + b_{neg}^{(2)} \end{pmatrix}.$$

The advantages of this method are that it is very easy to solve (it involves just the resolution of a linear system) and that it doesn't depend on the choice of t_{ref}. Moreover, we can add some data to adjust the new velocity.

Unfortunately, this method doesn't work because the outlier events weigh too much on the final result.
We consider an example with few synthetic events. Suppose we have 11 events moving with the same velocity on a straight line with equation $y = 2x + 5$ (first plot of Figure 2.1). Applying the explained method we obtain the correct velocities: as we can see from the second plot of Figure 2.1 all the events are warped to $t_{ref} = t_{min}$, where t_{min} stands for the minimum timestamp of the events. Now we add to the previous events some noise:

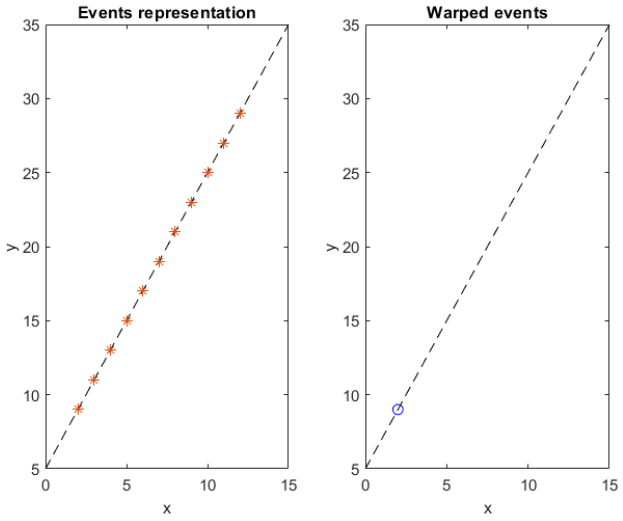

Figure 2.1: Example of synthetic data without noise. On the left are shown the events moving in the straight line with $y = 2x + 5$ with velocities $\mathbf{v} = (v_x, v_y) = (1, 2)$. On the right is represented the same events warped according to the estimated velocity.

a single event moving with different velocity. From Figure 2.2 can be seen that the events are not warped to the same point anymore, therefore the estimated velocity is incorrect.
For this reason, we have decided to leave behind this method to consider the model developed by [1].

2.2 Variational models for optical flow

Before specifically considering the loss function, we are going to make an introduction about the variational methods applied to the inverse problems, in which this model is cast.

Definition 2.2.1. A general system or formation model is defined as:
$y = \mathcal{N}(\Phi(x))$ where:

- $\Phi(x) = \phi(Ax)$ is the deterministic degradation operator where $\phi(\cdot)$ is

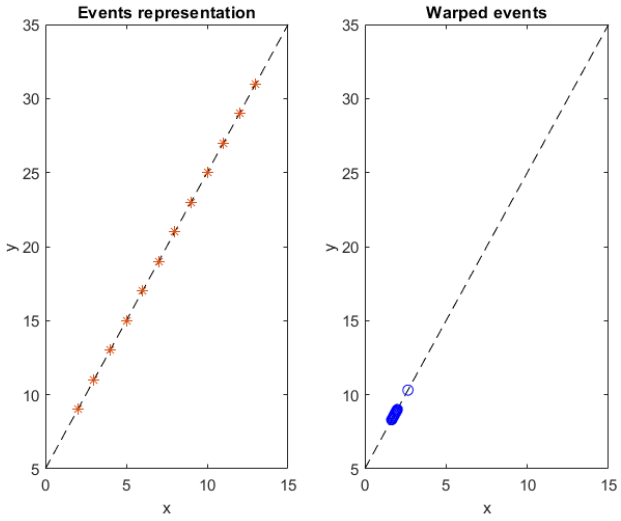

Figure 2.2: Example of synthetic data with noise. On the left, the events are moving on the line $y = 2x + 5$. On the right, we can see that the events are not warped to the same point.

the identity or a nonlinear operator and $A\cdot$ is a linear operator;

- $\mathcal{N}(\cdot)$ randomic noise operator.

We consider two different models:

- A *forward problem* is to compute the output, given the input and the system:

$$\text{input} \Rightarrow \text{system} \Rightarrow \text{output}.$$

- An *inverse problem* is to compute the input given the system and output, which is often noisy:

$$\text{input} \Leftarrow \text{system} \Leftarrow \text{output}.$$

Definition 2.2.2. A model is said to be **well-posed by Hadamard definition** if there exists a unique solution depending continuously on data. If these assumptions are not satisfied, the problem is **ill-posed**.

In particular:

- If noise is not considered, we obtain a deterministic forward model: $y = \Phi(x)$. The corresponding inverse problem is $x = \Phi^{-1}(y)$.
- if we consider noise, we have a probabilistic forward model: $y = \Phi(x)$ with $p(y|x)$. The inverse problem becomes $x = \Phi^{-1}(y)$ with $p(x|y)$.

The posedness or conditioning of the problem depends on the existence and properties of Φ^{-1}.

Definition 2.2.3. A **variational method** consists of the minimization of an energy function:

$$x^* = \underset{x \in \mathcal{D}}{\operatorname{argmin}} \{\mathcal{F}(x; y, A) + \mu \mathcal{R}(x)\} \text{ subject to } y = Ax.$$

$\mathcal{D}$ can be either a vector space $\mathbb{R}^n$ (discrete setting) or a function space (continuous setting). In particular

- $\mathcal{R}$ is the regularization term, gives priori information on the input;
- $\mathcal{F}$ is the fidelity term that gives information on the data acquisition model
- μ is the regularization parameter which sets a trade-off between the regularization and the fidelity.

The most popular class of variational model presents the two terms in the following form:
$\mathcal{R}(x) = \sum_{i=1}^{n} \Phi(g_i(x))$ where $g_i(x) = ||(\nabla x)_i||_2$ and

1. if $\Phi(g_i) = g_i^2$ it is a Tikhonov regularizer,
2. if $\Phi(g_i) = g_i^p$ it is a Total $p-$Variation regularizer.

$\mathcal{F}(x; y, A) = \frac{1}{q}||Ax - y||_q^q$.

To derive the variational models we use just the probabilistic definition. In particular, we consider two different estimations to recover the input x.

Consider the forward model where x is known, the **maximum likelihood estimator** x^*_{ML} is obtained by the maximization of the likelihood probability $p(y|x)$ which is equivalent to minimize the negative $log-$likelihood function, i.e.

$$x^*_{ML} \in \underset{x\in\mathbb{R}^n}{\operatorname{argmax}}\, p(y|x) = \underset{x\in\mathbb{R}^n}{\operatorname{argmin}}\{-ln(p(y|x)\}.$$

In the **maximum a posteriori estimation** (MAP), we consider the inverse model where the output y is given. The prior belief before seeing data $p(y)$ and the posterior distribution $p(x|y)$ which represents the updated belief on x after observing some data are known.
From Bayes' rule we have that

$$p(x|y) = \frac{p(x)p(y|x)}{p(y)}.$$

The maximum posterior estimator x^*_{MAP} is obtained as a maximization of the posterior distribution which is equivalent to the minimization of the negative $log-$posterior function:

$$\begin{aligned} x^*_{MAP} &\in \underset{x\in\mathbb{R}^n}{\operatorname{argmax}}\, p(x|y) = \underset{x\in\mathbb{R}^n}{\operatorname{argmin}}\{-ln(p(x|y))\} \\ &= \underset{x\in\mathbb{R}^n}{\operatorname{argmin}}\{-ln(p(x)) - ln(p(y|x)) + ln(p(y))\}. \end{aligned}$$

We erase the terms not depending on x and we obtain:

$$x^*_{MAP} \in \underset{x\in\mathbb{R}^n}{\operatorname{argmin}}\{-\underbrace{ln(p(x))}_{\mathcal{R}(x)} - \underbrace{ln(p(y|x))}_{\mathcal{F}(x;y,A)}\}.$$

Therefore we have obtained the maximization of the sum of two terms: the logarithm of the evidence $p(x)$ representing the regularization term and the log-likelihood representing the fidelity term. Depending on the features of the probabilistic formation model of these two distributions we obtain different loss functions.

2.3 Multi reference Focus Objective Function

To estimate the optical flow of a given set of events we have to maximize the alignment of the warped events, therefore the loss function has to mea-

sure the goodness of the warp motion applied.
In the majority of the literature, it is applied the so-called **contrast objective**: the loss is defined as the variance of H, that is

$$f_{var}(\mathbf{v}) = \mathrm{Var}[H(\boldsymbol{x}, \mathbf{v})] = \frac{1}{N_p}\sum_{i,j}^{N_p}(H(\boldsymbol{x}; \mathbf{v}) - \mu_H)^2 = \frac{1}{N_p}\sum_{i,j}^{N_p}(H(\boldsymbol{x}; \mathbf{v}) - \mu_H)^2 \tag{2.1}$$

with mean

$$\mu_H = \frac{1}{N_p}\sum_{i,j}^{N_p} H(\boldsymbol{x}, \mathbf{v}).$$

It has been observed that with very complex flow fields, the warp tends to accumulate the events in a few pixels. The result is that the objective function overfits the events. To partly resolve this problem we apply an alternative loss function proposed by Shiba et al. [1].

Definition 2.3.1. Consider three different reference times:

$$\underline{t} = \min_{k=1,..N_e}\{t_k\}; \quad \overline{t} = \max_{k=1,..N_e}\{t_k\}; \quad t_{mid} = \frac{\underline{t} + \overline{t}}{2}.$$

We define

$$G(\mathbf{v}) := ||\nabla H(\mathbf{v}; i, j)||_2^2.$$

The **multi reference focus loss** is the average of G at the three difference reference times, selected before, normalized by the identity warp $4G(\mathbf{0})$ with zero flow:

$$f(\mathbf{v}) := \frac{G(\mathbf{v}; \underline{t}) + 2G(\mathbf{v}; t_{mid}) + G(\mathbf{v}; \overline{t})}{4G(\mathbf{0})}.$$

We use the multi-focus loss function, instead of the contrast objective because:

- among the proposed objective functions it has the best performance and converges more easily;
- it is sensitive to permutations of the pixel values in H whereas the variance is not.

Given an image of dimensions $h \times w$, we have to find the velocity that maximizes the alignment of the events. Thus the problem is reduced to solve the following optimization problem

$$\begin{aligned}\mathbf{v}^* &= \underset{\mathbf{v}\in\mathbb{R}^{h\times w\times 2}}{\operatorname{argmax}}\{f(\mathbf{v})\} = \underset{\mathbf{v}\in\mathbb{R}^{h\times w\times 2}}{\operatorname{argmin}}\{-f(\mathbf{v})\} \\ &= \underset{\mathbf{v}\in\mathbb{R}^{h\times w\times 2}}{\operatorname{argmin}}\left\{-\frac{G(\mathbf{v};\underline{t}) + 2G(\mathbf{v};t_{mid}) + G(\mathbf{v};\overline{t})}{4G(\mathbf{0})}\right\}. \qquad (\mathcal{A})\end{aligned}$$

This model works fine how it is, however, we can add a regularizer to obtain additional smoothness of the flow, even in regions with few events. Therefore we obtain a composite optimization problem:

$$\mathbf{v}^* = \underset{\mathbf{v}\in\mathbb{R}^{h\times w\times 2}}{\operatorname{argmin}}\{F(\mathbf{v}) = -f(\mathbf{v}) + \mu\mathcal{R}(\mathbf{v})\}.$$

where $\lambda \in \mathbb{R}$ is the regularization parameter to be set. In the experiments of this project, we are going to consider the Vectorial Total Variation (Section 3.2.3), therefore the optimization problem becomes

$$\mathbf{v}^* = \underset{\mathbf{v}\in\mathbb{R}^{h\times w\times 2}}{\operatorname{argmin}}\left\{-\frac{G(\mathbf{v};\underline{t}) + 2G(\mathbf{v};t_{mid}) + G(\mathbf{v};\overline{t})}{4G(\mathbf{0})} + \mu\sum_{i=1}^{N_p}||(\mathbf{D}\mathbf{v})_i||_*\right\}. \quad (\mathcal{B})$$

To simplify calculations, in both the minimization problems $(\mathcal{A})$ and $(\mathcal{B})$ we have decided to use $-f(\mathbf{v})$, instead of $\dfrac{1}{f(\mathbf{v})}$ as it is proposed in [1].

In this project, we will try to resolve both models: in Section 3.1 we consider the minimization problem $(\mathcal{A})$ applying the *Steepest Descent* method, in Section 3.2 we consider the composite problem $(\mathcal{B})$ applying *Proximal Gradient Method* and *Accelerated Proximal Gradient Method.*

Convexity of f

Definition 2.3.2. A function $g : \mathbb{R}^n \to \mathbb{R}$ is said to be convex if and only if $\forall\theta \in [0,1],\ \forall\boldsymbol{x},\mathbf{y} \in \mathbb{R}^n$ holds

$$g(\theta\mathbf{x} + (1-\theta)\mathbf{y}) \leq \theta g(\mathbf{x}) + (1-\theta)g(\mathbf{y}). \qquad (2.2)$$

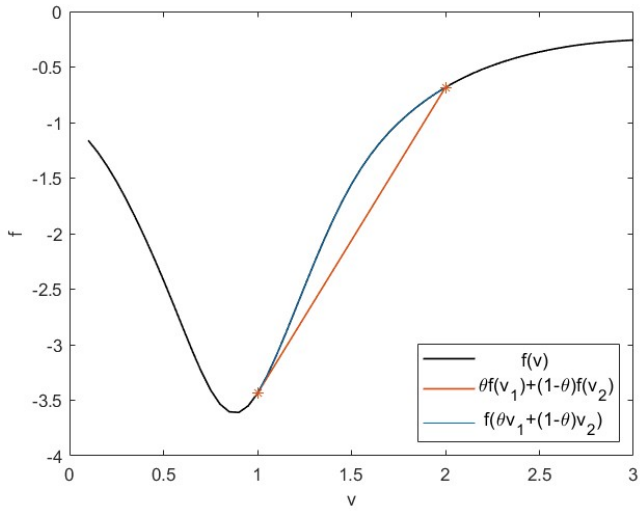

Figure 2.3: Plot of the objective function $f(\mathbf{v})$. The red segment represents the function $\theta f(\mathbf{v}_1) + (1-\theta) f(\mathbf{v}_2)$ by varying θ. The intersection points between the segment and the function are $(\mathbf{v}_1, f(\mathbf{v}_1))$ on the left and $(\mathbf{v}_2, f(\mathbf{v}_2)$ on the right. The blue part of the plot represents the function $f(\theta\mathbf{v}_1 + (1-\theta)\mathbf{v}_2)$. $f(\mathbf{v})$ is non-convex because the red segment is under the blue section of the curve.

We want to show that in our case the relation (2.2) does not hold and therefore, the multi-reference focus objective function f is non-convex. We consider two different constant velocities: $\mathbf{v}_{1,k} = (1,1)$ and $\mathbf{v}_{2,k} = (2,2)$, $\forall$ event $k = 1, ...5$. Then, setting $\theta = 0.5 \in [0,1]$, $\mathbf{v}_1 = (\mathbf{v}_{1,1}, ...\mathbf{v}_{1,5})$ and $\mathbf{v}_2 = (\mathbf{v}_{2,1}, ...\mathbf{v}_{2,5})$ we have

$$f(\theta\mathbf{v}_1 + (1-\theta)\mathbf{v}_2) = -1.55 \nleq -2.06 = \theta f(\mathbf{v}_1) + (1-\theta) f(\mathbf{v}_2).$$

The graphical interpretation is presented in Figure 2.3 where we plotted the objective function $f(\mathbf{v})$. We can observe that the red segment $\theta f(\mathbf{v}_1) + (1-\theta) f(\mathbf{v}_2)$ is under the part of the plot highlighted in blue $f(\theta\mathbf{v}_1 + (1-\theta)\mathbf{v}_2)$, therefore the objective function is non-convex.

The non-convexity of the function implies the non-existence and uniqueness of a global minimum. Indeed we will observe that applying the Steepest Descent and the Proximal Gradient algorithms gives as a result also local

minima, that don't necessarily coincide with the most accurate result.

2.3.1 Computation of the gradient of the loss function

Throughout the project, for model ($\mathcal{A}$) and ($\mathcal{B}$) to apply the Stesspest Descent and the Gradient Descent algorithm, we'll need the gradient of the objective function $f(\mathbf{v})$.
We consider an image plane of dimension $h \times w$ hence H is an image of the same dimension. The magnitude of the gradient of the IWE can be written in the following way:

$$G(\mathbf{v}) = ||\mathbf{D}H(\mathbf{v}; i, j)||_2^2$$

where $\mathbf{D} = \begin{pmatrix} D_h \\ D_v \end{pmatrix} \in \mathbb{R}^{2N_p \times N_p}$ and $D_h, D_v \in \mathbb{R}^{N_p \times N_p}$ are coefficients matrices of linear difference operators discretizing horizontal and vertical partial derivatives of the image H of dimension $w \times h = N_p$.

The function G can be written as composition of three different functions:

$$G(\mathbf{v}) = G_1(G_2(G_3(\mathbf{v}))),$$

where in particular:

- $G_1 : \mathbb{R}^{h \times w} \to \mathbb{R}^+ \quad G_1(\mathbf{u}) = ||\mathbf{D}\mathbf{u}||_2^2 \quad \forall \mathbf{u} \in \mathbb{R}^{h \times w}$
- $G_2 : \mathbb{R}^{N_e \times 2} \to \mathbb{R}^{h \times w} \quad G_{2,i,j}(\mathbf{y}) = \sum_{k=1}^{N_e} \mathcal{N}((i,j); \mathbf{y}_k, \sigma^2 \mathbf{I}_2) \quad \forall \mathbf{y} \in \mathbb{R}^{N_e \times 2}$, $i = 1, ...h, \ j = 1, ..w$
- $G_3 : \mathbb{R}^{h \times w \times 2} \to \mathbb{R}^{N_e \times 2} \quad G_{3,k}(\mathbf{v}) = \boldsymbol{x}_k - \Delta t_k \mathbf{v}(i_k, j_k) \quad \forall \mathbf{v} \in \mathbb{R}^{h \times w \times 2}$, $k = 1, ..N_e$.

Therefore the gradient of the objective function $f(\mathbf{v})$ can be computed as:

$$\nabla f(\mathbf{v}) = \frac{\mathbf{J}_{G(\mathbf{v};t_1)} + 2\mathbf{J}_{G(\mathbf{v};t_{mid})} + \mathbf{J}_{G(\mathbf{v};t_{N_e})}}{4G(\mathbf{0})}.$$

We compute at first $\mathbf{J}_{G(\mathbf{v};t_{ref})}$.
To simplify the representation and avoid tensors we vectorize all the variables. Define:

- $\boldsymbol{x}_{ev} = (i_1, ..., i_{N_e}, j_1, ..., j_{N_e})^T \in \mathbb{R}^{2N_e \times 1}$ the vector containing the coordinates of each event
- $\boldsymbol{x}'_{ev} = (i'_1, ..., i'_{N_e}, j'_1, ..., j'_{N_e})^T \in \mathbb{R}^{2N_e}$ the vector containing the coordinates of each warped event
- $\Delta\mathbf{t}_{ev} = ((t_1 - t_{ref}), ...(t_{N_e} - t_{ref}), (t_1 - t_{ref}), ...(t_{N_e} - t_{ref}))$
  $= (\Delta t_1, ...\Delta t_{ref}, \Delta t_1, ...\Delta t_{ref})^T \in \mathbb{R}^{2N_e}$
- $\mathbf{v} = (\text{vec}(\mathbf{v}^{(1)}), \text{vec}(\mathbf{v}^{(2)}))^T \in \mathbb{R}^{2N_p \times 1}$ is the vectorization of the velocity referred to each pixel location (i, j) $\forall$ $i = 1, ...h, j = 1, ...w$.

We have to compute the gradient of G, i.e.:

$$\begin{aligned}\mathbf{J}_G(\mathbf{v}) &= \mathbf{J}_{G_1 \circ G_2 \circ G_3}(\mathbf{v}) \\ &= \mathbf{J}_{G_1}(G_2(G_3(\mathbf{v})))\mathbf{J}_{G_2}(G_3(\mathbf{v}))\mathbf{J}_{G_3}(\mathbf{v}) \\ &= \mathbf{J}_{G_1}(\mathbf{u})\mathbf{J}_{G_2}(\mathbf{y})\mathbf{J}_{G_3}(\mathbf{v}).\end{aligned}$$

- Computation of $\mathbf{J}_{G_3}(\mathbf{v})$ where $G_3 : \mathbb{R}^{2N_p} \to \mathbb{R}^{2N_e}$ and $G_{3,k}(\mathbf{v}) = \boldsymbol{x}_k - \Delta t_k \mathbf{v}(i_k, j_k)$ $\forall \mathbf{v} \in \mathbb{R}^{h \times w \times 2}, k = 1, ..N_e$.
  The function G_3 can be seen in matrix form in the following way:

$$G_3(\mathbf{v}) = \begin{pmatrix} i'_1 \\ \vdots \\ i'_{N_e} \\ j'_1 \\ \vdots \\ j'_{N_e} \end{pmatrix} = \begin{pmatrix} i_1 \\ \vdots \\ i_{N_e} \\ j_1 \\ \vdots \\ j_{N_e} \end{pmatrix} - \begin{pmatrix} t_1 - t_{ref} \\ \vdots \\ t_{N_e} - t_{ref} \\ t_1 - t_{ref} \\ \vdots \\ t_{N_e} - t_{ref} \end{pmatrix} \odot \left(\begin{array}{c|c} \mathbf{S}^{(1)} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{S}^{(2)} \end{array}\right) \begin{pmatrix} v^{(1)}_{1,1} \\ \vdots \\ \vdots \\ v^{(1)}_{h,w} \\ v^{(2)}_{1,1} \\ \vdots \\ \vdots \\ v^{(2)}_{h,w} \end{pmatrix}$$

  where $\odot$ represents the Hadamard product (or entry-wise product). Then

$$G_3(\mathbf{v}) = \boldsymbol{x}'_{ev} = \boldsymbol{x}_{ev} - \Delta\mathbf{t}_{ev} \odot \mathbf{S}\mathbf{v} = \boldsymbol{x}_{ev} - \text{diag}(\Delta\mathbf{t}_{ev})\mathbf{S}\mathbf{v}.$$

  In particular:

– $\mathbf{S} = \begin{pmatrix} \mathbf{S}^{(1)} & \mathbf{0} \\ \mathbf{0} & \mathbf{S}^{(2)} \end{pmatrix}$ is a block binary selection matrix of dimension $(2N_e \times 2N_p)$ where $\mathbf{S}^{(1)} = \mathbf{S}^{(2)} \in \mathbb{R}^{N_e \times N_p}$ and

$$\mathbf{S}^{(l)}_{n,m} = \begin{cases} 1 & \text{if } \exists \text{ the } n_{th} \text{ event in pixel location } (i_n, j_n) \text{ s.t. } m = (j_n - 1)w + i_n \\ 0 & \text{otherwise} \end{cases}$$

$\forall\ l = 1, 2\ n = 1, ...N_e,\ m = 1, ...N_p.$
In particular, note that for every row of $\mathbf{S}$ there exists at most one entry equal to 1;

- $\text{diag}(\Delta t_{ev})$ is a $(2N_e \times 2N_e)$ matrix whose diagonal is the vector Δt_{ev}.

Then we have:

$$\mathbf{J}_{G_3}(\mathbf{v}) = -\text{diag}(\Delta \mathbf{t}_{ev})\mathbf{S} \in \mathbb{R}^{2N_e \times 2N_p}.$$

- Computation of $\mathbf{J}_{G_2}(\mathbf{y})$ where $G_2 : \mathbb{R}^{2N_e} \to \mathbb{R}^{N_p}$ and $G_{2,i,j}(\mathbf{y}) = \sum_{k=1}^{N_e} \mathcal{N}((i,j); \mathbf{y}_k, \sigma^2 \mathbf{I}_2)\ \forall \mathbf{y} \in \mathbb{R}^{N_e \times 2}$.
Consider $G_{2,i,j}$ as a vector of dimension N_p and define $\mathbf{y} = (\text{vec}(y^{(1)}), \text{vec}(y^{(2)}))^T$, then $\forall i = 1, ...h,\ j = 1, ...w,\ k = 1, ...N_p$

$$\begin{aligned} \frac{\partial G_{2,i,j}}{\partial y_k^{(1)}} &= \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(i - y_k^{(1)})^2 + (j - y_k^{(2)})^2}{2\sigma^2} \right) \left(\frac{2(i - y_k^{(1)})}{2\sigma^2} \right) \\ &= \frac{i - y_k^{(1)}}{2\pi\sigma^4} \exp\left(-\frac{(i - y_k^{(1)})^2 + (j - y_k^{(2)})^2}{2\sigma^2} \right) \\ \frac{\partial G_{2,i,j}}{\partial y_k^{(2)}} &= \frac{j - y_k^{(2)}}{2\pi\sigma^4} \exp\left(-\frac{(i - y_k^{(1)})^2 + (j - y_k^{(2)})^2}{2\sigma^2} \right). \end{aligned}$$

As a consequence, setting $C_{i,j,k} := \frac{1}{2\pi\sigma^4} \exp\left(-\frac{(i-y_k^{(1)})^2 + (j-y_k^{(2)})^2}{2\sigma^2} \right)$, for $i = 1, ...h\ j = 1, ..w$

$$(\mathbf{J}_{G_2})_{i,j} = \begin{pmatrix} (i - y_1^{(1)})C_{i,j,1} & \cdots & (i - y_{N_e}^{(1)})C_{i,j,N_e} & (j - y_1^{(2)})C_{i,j,1} & \cdots & (j - y_{N_e}^{(2)})C_{i,j,Ne} \end{pmatrix}.$$

In particular, we construct:

$$\mathbf{dx} = \begin{pmatrix} 1-y_1^{(1)} & \dots & 1-y_{N_e}^{(1)} \\ 2-y_1^{(1)} & \dots & 2-y_{N_e}^{(1)} \\ \vdots & \ddots & \vdots \\ h-y_1^{(1)} & \dots & h-y_{N_e}^{(1)} \\ \vdots & \ddots & \vdots \\ 1-y_1^{(1)} & \dots & 1-y_{N_e}^{(1)} \\ \vdots & \ddots & \vdots \\ h-y_1^{(1)} & \dots & h-y_{N_e}^{(1)} \end{pmatrix}, \ \mathbf{dy} = \begin{pmatrix} 1-y_1^{(2)} & \dots & 1-y_{N_e}^{(2)} \\ 1-y_1^{(2)} & \dots & 1-y_{N_e}^{(2)} \\ \vdots & \ddots & \vdots \\ 1-y_1^{(2)} & \dots & 1-y_{N_e}^{(2)} \\ \vdots & \ddots & \vdots \\ w-y_1^{(2)} & \dots & w-y_{N_e}^{(2)} \\ \vdots & \ddots & \vdots \\ w-y_1^{(2)} & \dots & w-y_{N_e}^{(2)} \end{pmatrix}$$

and

$$\begin{aligned} \mathbf{C} &= \frac{1}{2\pi\sigma^4} \begin{pmatrix} \exp\left(-\frac{(1-y_1^{(1)})^2+(1-y_1^{(2)})^2}{2\sigma^2}\right) & \dots & \exp\left(-\frac{(1-y_{N_e}^{(1)})^2+(1-y_{N_e}^{(2)})^2}{2\sigma^2}\right) \\ \exp\left(-\frac{(2-y_1^{(1)})^2+(1-y_1^{(2)})^2}{2\sigma^2}\right) & \dots & \exp\left(-\frac{(2-y_{N_e}^{(1)})^2+(1-y_{N_e}^{(2)})^2}{2\sigma^2}\right) \\ \vdots & \ddots & \vdots \\ \exp\left(-\frac{(h-y_1^{(1)})^2+(w-y_1^{(2)})^2}{2\sigma^2}\right) & \dots & \exp\left(-\frac{(h-y_{N_e}^{(1)})^2+(w-y_{N_e}^{(2)})^2}{2\sigma^2}\right) \end{pmatrix} \\ &= \frac{1}{2\pi\sigma^4} \exp\left(-\frac{\mathbf{dx}^{\odot 2}+\mathbf{dy}^{\odot 2}}{2\sigma^2}\right) \end{aligned}$$

where $\cdot^{\odot 2}$ is the element wise exponentiation.

Therefore we obtain the following:

$$\mathbf{J}_{G_2} = \left(\ \mathbf{dx} \odot \mathbf{C} \ \middle| \ \mathbf{dy} \odot \mathbf{C} \ \right) \in \mathbb{R}^{N_p \times 2N_e}.$$

- Computation of $\mathbf{J}_{G_1}(\mathbf{u})$ where $G_1 : \mathbb{R}^{N_p} \to \mathbb{R}^+$ and $G_1(\mathbf{u}) = ||\mathbf{Du}||_2^2$.
The function G_1 can be written as:

$$G_1(\mathbf{u}) = (\mathbf{Du})^T(\mathbf{Du}) = \mathbf{u}^T(D_h^T D_h + D_v^T D_v)\mathbf{u}.$$

Hence:

$$\mathbf{J}_{G_1}(\mathbf{u}) = 2\mathbf{u}^T(D_h^T D_h + D_v^T D_v).$$

Implementation of the loss function

In previous sections, we have seen that H is computationally very expensive. In the implementation, to reduce the cost, we apply a truncated square Gaussian support proposed in [38] that achieves the same accuracy but with a linear time complexity. In a Gaussian distribution, the 98.9% of the data falls in a neighborhood of 3 standard deviation of the mean, therefore the 98.9% influence for $\boldsymbol{x}'_k$ is extended for events $\boldsymbol{x}$ up to 3 standard deviation of $\boldsymbol{x}'_k$.
For this reason, we use the following function:

$$H(\mathbf{v};\boldsymbol{x}) = \sum_{k=1}^{N_e} \mathcal{N}(\boldsymbol{x};\boldsymbol{x}'_k,\sigma^2\mathbf{I}_2) \quad \forall \boldsymbol{x} = \lfloor \boldsymbol{x}'_k \rceil \pm r\sigma$$

where $1 \leq r \leq 3$.
We will see, in the experiments proposed, that the choice of the dimension of the kernel has an important impact on the accuracy of the estimation, that happens because using a truncation means also adding some discontinuities to the function.
The computation of H has linear time complexity $\mathcal{O}(N_e)$ which is independent of the size of the sensor.

This choice reduces also the complexity of the matrices $\mathbf{dx}$, $\mathbf{dy}$, $\mathbf{C}$ and as a consequence $\mathbf{J}_{G_2}$. If we set, for instance, $r = 3$, the derivative of G has not to be computed at every pixel location. Indeed we are interested only in pixels falling in the neighborhood of dimension 7×7 centered at $\boldsymbol{x}'_k,\ \forall k = 1, ...N_e$.
$\mathbf{dx}$ and $\mathbf{dy}$ become a sparse $N_p \times N_e$ matrix where in the k_{th} column we have at most 49 non-zero entries (24 over ad 24 under the k_{th} element equivalent to i'_k and j'_k respectively).
By definition, also $\mathbf{C}$ and $\mathbf{J}_{G_2}$ become sparse matrix.

2.3.2 Boundary conditions

To construct the matrix $\mathbf{D}$ and in particular D_h and D_v that discretize the partial derivative of H, we have to assume some boundary conditions. We consider the ones that are more related to the situation we have to handle. The conveyor moves with constant velocity and at the boundary of the image, there are noisy events caused by the conveyor. The most logical idea seems to consider reflective boundary conditions, also called Neumann homogeneous conditions. In this case, the boundary of the image is reflected preserving the same sign.

Definition 2.3.3. In particular consider a grid with n points $\{x_0, x_1, ..., x_n\}$ and a function f differentiable, the reflective boundary conditions take the following form:

$$f'(x_0) = 0, \quad f'(x_n) = 0.$$

For example, we analyze the one-dimensional case. We want to compute the matrix $\mathbf{d}$ discretizing the approximation of the first order derivative of a vector $\mathbf{x} = (x_1, x_2, ..., x_n)$.
We approximate the first derivative with forward differences:

$$f'(x) = \frac{f(x+h) - f(x)}{h} + \mathcal{O}(h)$$

then we have:

$$\begin{aligned}
&\text{for } k = 1: && y_1 = f'(x_1) = f(x_2) - f(x_1) \\
&\text{for } 1 < k < n: && y_k = f'(x_k) = f(x_{k+1}) - f(x_k) \\
&\text{for } k = n: && y_n = f'(x_n) = f(x_{n+1}) - f(x_n) = f(x_n) - f(x_n) = 0.
\end{aligned}$$

Therefore we obtain a bidiagonal matrix:

$$\mathbf{d} = \begin{pmatrix} -1 & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \\ & & & & 0 \end{pmatrix} \in \mathbb{R}^{n \times n} \quad \text{s.t.} \quad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \mathbf{d} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}. \quad (2.3)$$

In our case, we have to compute the derivative of the matrix of dimension $h \times w$. Indeed for what concerns the construction of the matrices D_v and D_h, we create at first the matrix $\mathbf{d}_h \in \mathbb{R}^{w \times w}$ and $\mathbf{d}_v \in \mathbb{R}^{h \times h}$ with the same bidiagonal form of (2.3). Then:

$$D_h = \mathbf{d}_h \otimes \mathbf{I}_{h \times h}$$

$$D_v = \mathbf{d}_v \otimes \mathbf{I}_{v \times v}$$

where $\otimes$ denotes the Kronecker product. Both of the matrices have a block structure.

2.4 Estimation with constant velocity

We consider now a particular case of the proposed model. We assume that the optical flow is constant in every pixel, i.e.

$$\mathbf{v}(i,j) = \mathbf{v} \quad \forall i = 1, ...h, \ j = 1, ...w.$$

This result will be used also as initialization for the estimation of the dense optical flow.
We consider the situation described in Section 1.3.2, where the parcel is passed under the event camera. We know the parcel used for the experiments is $15cm$ high.
During the acquisition, the camera overall collects more than $1.8 \cdot 10^7$ events. Obviously, this number of events is too high and will cause an out-of-memory problem. Moreover, the events are not only caused by the edge movement of the parcel but also by the noise on the conveyor. Therefore to reduce the computational time and eventually avoid any outliers problem, we consider a spatial and temporal region of interest of events. In particular, we select events with

- time stamp $t_k \in [t_{start}, t_{start} + \Delta t] = [3.68 \cdot 10^6, 3.68 \cdot 10^6 + 3 \cdot 10^4] \mu s \ \forall k = 1, ...N_e$

- pixel coordinates $j_k \in [x_{min}, x_{max}] = [370, 837]$ and $i_k \in [y_{min}, y_{max}] = [250, 544]$ $\forall k = 1, ...N_e$.

This reduces the dataset to $4.4 \cdot 10^5$ events. In this experiment, it is sufficient to use the contrast objective function defined in (2.1), therefore we solve:

$$\mathbf{v}^* = \underset{\mathbf{v} \in \mathbb{R}^{h \times w}}{\operatorname{argmin}} f_{var}(\mathbf{v}) = \underset{\mathbf{v} \in \mathbb{R}^{h \times w}}{\operatorname{argmin}} \operatorname{Var}[H(\boldsymbol{x}, \mathbf{v})]$$

where $h = y_{max} - y_{min} + 1 = 294$ and $w = x_{max} - x_{min} + 1 = 467$. We apply the algorithm L-BFGS-B of the Python library SciPy, providing just the Jacobian function and the initial guess of optical flow $\mathbf{v}_0 = (0, 0)$ [1].
After 13 iterations, the method estimates an optical flow of

$$\mathbf{v}^* = (1250.68164, 20.5165) pixel/s.$$

Figure 2.4 compares the graphical representation of H with the initial guess $\mathbf{v}_0$ with the same representation with estimated optical flow $\mathbf{v}^*$. We can observe that the edges appear to be more aligned. To better understand the accuracy of the results we can directly estimate the associated distance h^* between the parcel and the camera. It has an estimated value of $1.24953m$. Applying equation (1.6) we obtain that the parcel has a height of:

$$h_{parcel} = h_{camera} - h^* = 1.4m - 1.24953m = 0.15047m.$$

We can conclude that the method estimates the correct parcel dimension with an absolute error of $4.7 \cdot 10^{-4}$.

[1] this code has been provided by Datalogic S.p.A..

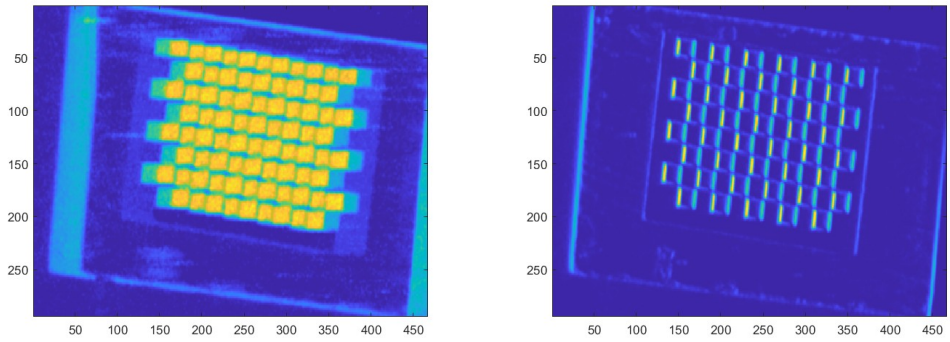

Figure 2.4: Graphical representation of H with the events warped with initial velocities $\mathbf{v} = (0, 0) pixel/s$ (on the left) and estimated velocities $\mathbf{v} = (1250.7, 20.5) pixel/s$ (on the right), plotted on a camera domain of dimension 294×467.

Chapter 3

Optimization Algorithms

In this chapter, we will analyze the two algorithms that will be used for the optical flow estimation. In both cases, after a brief introduction to the related theory, we will examine the algorithms and their convergence analysis.

3.1 Steepest Descent

Firstly we consider the optimization problem described in $(\mathcal{A})$, to solve it we apply the *Steepest Descent Algorithm* [39].

3.1.1 Line search-based algorithm

Optimization problems have the following form:

$$\min_{\boldsymbol{x} \in S} f(\boldsymbol{x})$$

where S is the **feasible region** and $\boldsymbol{x}$ is the vector of variables.

- If $S = \mathbb{R}^n$ it is said to be an **unconstrained problem**;
- if $S \neq \mathbb{R}^n$ it is said to be an **constrained problem**.

Optimization problems generate a sequence of candidate solutions $\boldsymbol{x}_k$ that stop when either it can not make any more progress or when it reaches a solution point with high accuracy.

In order to solve our problem $(\mathcal{A})$ we will focus on unconstrained optimization.
We consider $f : \mathbb{R}^n \to \mathbb{R}$, $f \in C^1$ (i.e. f is continuously differentiable).

The **line search based algorithms** are methods characterized by the following framework:
given the starting point $\boldsymbol{x}_0$, until convergence we do:

1. choose a **descent direction** $\mathbf{p}_k \in \mathbb{R}^n$, i.e. such that $\mathbf{p}_k^T \nabla f(\boldsymbol{x}_k) < 0$ ($\mathbf{p}_k$ makes an angle of strictly less than $\frac{\pi}{2}$ radians with $-\nabla f(\boldsymbol{x}_k)$);
2. determine a **step size** $\alpha_k \in \mathbb{R}$ such that $f(\boldsymbol{x}_k + \alpha_k \mathbf{p}_k) < f(\boldsymbol{x}_k)$;
3. update the iteration $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \mathbf{p}_k$.

We obtain different algorithms depending on the choice of the step size and the search direction.
The algorithm stops when one of the following criteria is satisfied. Given $\varepsilon > 0$ small, we have:

- *Gradient norm*: $||\nabla f(\boldsymbol{x}_k)|| < \varepsilon$, indeed near local minima $\nabla f(\boldsymbol{x}_k) \approx 0$
- *Step size*: $||\boldsymbol{x}_{k+1} - \boldsymbol{x}_k|| < \varepsilon$
- *Relative objective change*: $\dfrac{|f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1})|}{|f(\boldsymbol{x}_k)|} < \varepsilon$.

Steepest descent direction

We select the **steepest descent direction**, i.e. we fix:

$$\mathbf{p}_k = -\nabla f(\boldsymbol{x}_k).$$

The negative gradient maximizes the directional derivatives of $f \in C^1$ at $\boldsymbol{x}_k$ along the direction $\mathbf{p}$, s.t. $||\mathbf{p}|| = 1$.
Indeed by definition of directional derivative

$$\frac{\partial f}{\partial \mathbf{p}}(\boldsymbol{x}_k) = \nabla f(\boldsymbol{x}_k)^T \mathbf{p} = ||\nabla f(\boldsymbol{x}_k)||||\mathbf{p}|| \cos\theta = ||\nabla f(\boldsymbol{x}_k)|| \cos\theta$$

where $\theta \in [0, \pi]$ is the angle between $\nabla f(\boldsymbol{x}_k)$ and $\mathbf{p}$. Then it is minimized for $\theta = \pi$ and $\mathbf{p} = -\dfrac{\nabla f(\boldsymbol{x}_k)}{||\nabla f(\boldsymbol{x}_k)||}$.

Inexact line search

The next step is to choose α_k such that it guarantees a sufficient decrease:

1. constant step size: $\alpha = \alpha_k \ \forall k \geq 0$
2. exact line search: $\alpha_k = \underset{\alpha>0}{\operatorname{argmin}} f(\boldsymbol{x}_k + \alpha_k \mathbf{p}_k)$
3. inexact line search: we compute different candidates for α_k until it does not satisfy the Armijo sufficient decrease condition (3.1).

It is important to choose the best step size, indeed if the step is too large the iterates start to oscillate near the minimum (Figure 3.1), and if it is too short they cannot reach the solution (Figure 3.2). In both cases, the method cannot converge.
In this project, we will focus on the inexact line search because, with a constant step size, the method shows oscillations preventing convergence.

Well-definition

The simple condition $f(\boldsymbol{x}_{k+1}) < f(\boldsymbol{x}_k)$ is not sufficient to guarantee the convergence $||\nabla f(\boldsymbol{x}_k)|| \to 0$. Therefore we have to consider a more complex condition.

Definition 3.1.1. We define the **Armijo sufficient decrease condition** as

$$f(\boldsymbol{x}_k + \alpha_k \mathbf{p}_k) \leq f(\boldsymbol{x}_k) + c_1 \alpha \nabla f(\boldsymbol{x}_k)^T \mathbf{p}_k \tag{3.1}$$

where $c_1 \in]0, 1[$ (usually it is set as 10^{-4}).

Figure 3.3 represents the graphical motivation of Armijo's rule. We set $\phi(\alpha) = f(\boldsymbol{x}_k + \alpha_k \mathbf{p}_k)$, then $\phi'(\alpha) = \nabla f(\boldsymbol{x}_k + \alpha \mathbf{p}_k)^T \mathbf{p}_k$. Consider the plot of

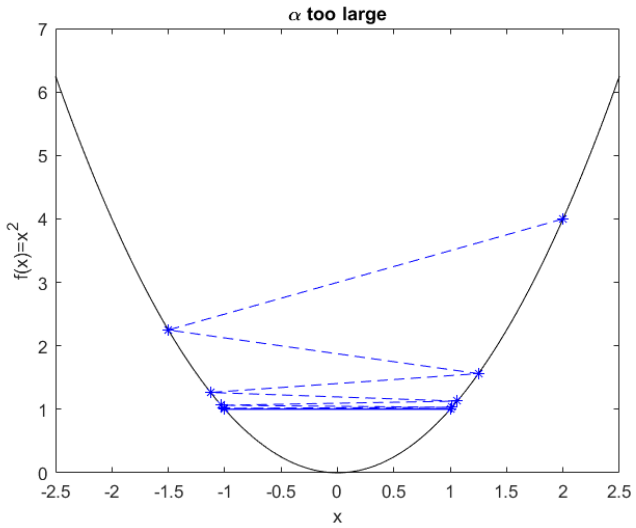

Figure 3.1: Example of a function where the chosen step size is too large. Consider $f(x) = x^2$, we set $x_0 = 2$, $p_k = (-1)^{k+1}$ and $\alpha_k = 2 + \frac{3}{2^{k+1}}$. The method cannot reach the minimum because it oscillates.

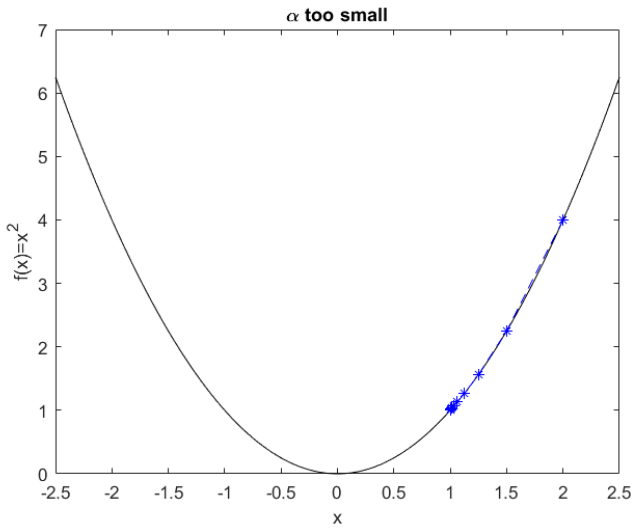

Figure 3.2: Example of a function where the chosen step size is too small. Consider $f(x) = x^2$, we set $x_0 = 2$, $p_k = -1$ and $\alpha_k = \frac{1}{2^{k+1}}$. The method cannot reach the minimum because it stops before.

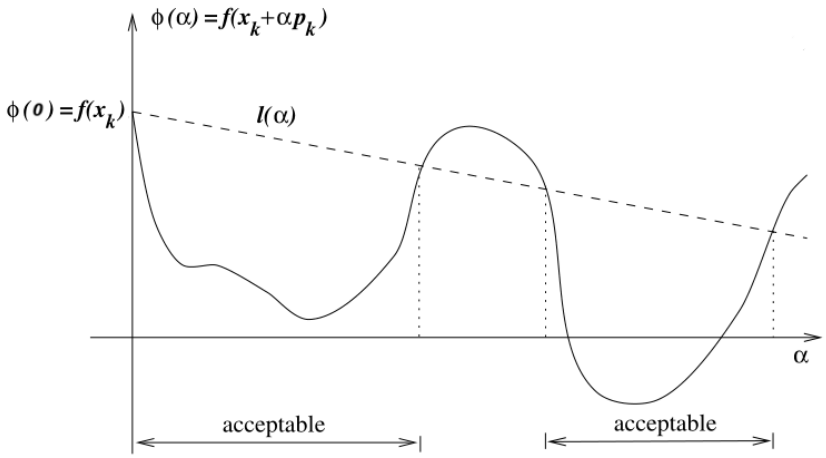

Figure 3.3: Graphical interpretation of the Armijo rule. The dotted line represents $l(\alpha) = f(\boldsymbol{x}_k) + c_1\alpha\nabla f(\boldsymbol{x}_k)^T\mathbf{p}_k$. The acceptable α's values are highlighted down the graph (Nocedal, Wright [39]).

$\phi(\alpha)$.
We trace the tangent line to $\phi(\alpha)$ at the point $(0, \phi(0))$, it has equation:

$$y(\alpha) = \phi(0) + \alpha\phi'(0) = f(\boldsymbol{x}_k) + \alpha\nabla f(\boldsymbol{x}_k)^T p_k.$$

We set $l(\alpha) = f(\boldsymbol{x}_k) + c_1\alpha\nabla f(\boldsymbol{x}_k)^T\mathbf{p}_k$. The slope of $l(\alpha)$ is:

$$c_1\phi'(0) = \underbrace{c_1}_{>0}\underbrace{\nabla f(\boldsymbol{x}_k)^T\mathbf{p}_k}_{<0} > \nabla f(\mathbf{x}_k)^T\mathbf{p}_k = \phi'(0).$$

To conclude, we select the values of step size that decrease the function but such that α is not too large.

Definition 3.1.2. Define the **Wolfe condition** as:

$$\nabla f(\boldsymbol{x}_k + \alpha_k\mathbf{p}_k)^T\mathbf{p}_k \geq c_2\nabla f(\boldsymbol{x}_k)^T\mathbf{p}_k$$

where $c_2 \in]c_1, 1[$.

Figure 3.4 shows the graphical interpretation of Wolfe's condition.
$\phi'(\alpha_k)$ is the slope of the line tangent to $\phi(\alpha)$ at $(\alpha_k, \phi(\alpha_k))$.

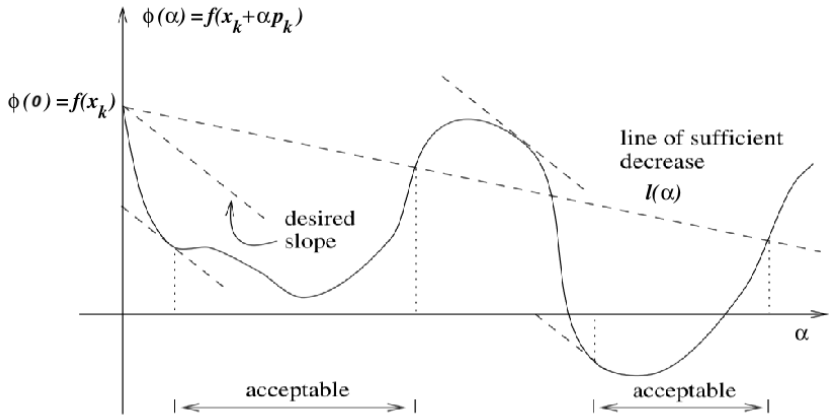

Figure 3.4: Graphical interpretation of the Wolfe's rule. The dotted line represents $l(\alpha) = f(\boldsymbol{x}_k) + c_1\alpha\nabla f(\boldsymbol{x}_k)^T\mathbf{p}_k$. The acceptable α's values are highlighted down the graph (Nocedal, Wright [39]).

The desired slope is $c_2\nabla f(\boldsymbol{x}_k)^T\mathbf{p}_k$.
By definition $c_1 < c_2 < 0$ and $\nabla f(\boldsymbol{x}_k)^T\mathbf{p}_k$ because it is a descent direction, then

$$c_1\nabla f(\boldsymbol{x}_k)^T\mathbf{p}_k > c_2\nabla f(\boldsymbol{x}_k)^T\mathbf{p}_k > \nabla f(\boldsymbol{x}_k)^T\mathbf{p}_k.$$

Therefore we have that the desired slope is between the slope of $l(\alpha)$ and the slope of $y(\alpha)$.

Theorem 3.1.3 (Wolfe's Lemma). *Consider $f : \mathbb{R}^n \to \mathbb{R}$, $f \in C^1$ and bounded below along $\{\boldsymbol{x}_k + \alpha\boldsymbol{p}_k | \alpha > 0\}$. If $0 < c_1 < c_2 < 1$, then exist a set $I \neq \emptyset$, $I \subset]0, +\infty[$ such that $\alpha \in I$ satisfy both Armijo and Wolfe condition.*

The previous lemma assures that if the Armijo and the Wolfe conditions are satisfied the algorithm is well defined. Remain to prove the convergence.

Convergence

Definition 3.1.4. A function $f : \mathbb{R}^n \to]-\infty, +\infty]$ is **L smooth** over a set $D \subset \mathbb{R}^n$ ($f \in C_L^{1,1}(D)$) with $L \geq 0$ if it is continuously differentiable over D

with Lipshitz continuous gradient, i.e.

$$||\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})|| \leq L||\mathbf{x} - \mathbf{y}|| \quad \forall \mathbf{x},\ \mathbf{y} \in D.$$

Theorem 3.1.5 (Zoutendijk's Theorem). *Let $\Omega = \{\boldsymbol{x} \in \mathbb{R}^n \mid f(\boldsymbol{x}) \leq f(\boldsymbol{x}_0)\}$, with $f \in C_L^{1,1}(\Omega)$, bounded below on Ω. Consider the sequence $\{\boldsymbol{x}_k\}$ generated by Steepest Descent with inexact line search. Then*

$$\lim_{k \to +\infty} \nabla f(\boldsymbol{x}_k) = 0.$$

Moreover, given $\varepsilon > 0$ and $\overline{k}_\varepsilon$ the first iteration such that $||\nabla f(\boldsymbol{x}_{\overline{k}_\varepsilon})|| \leq \varepsilon$, then

$$\overline{k}_\varepsilon \leq \left\lceil \frac{1}{\varepsilon^2} \frac{f(\boldsymbol{x}_0) - f_{low}}{c} \right\rceil$$

is the iteration complexity, where f_{low} is such that $f(\boldsymbol{x}) > f_{low}$ $\forall x \in \Omega$ and $c = \frac{c_1(1-c_2)}{L}$.

Remark 3.1.6. This theorem says that every limit point of $\boldsymbol{x}_k$ if exists, is a stationary point, therefore the steepest descent is a globalization strategy. In addition $||\nabla f(\boldsymbol{x}_k)||$ decreases at a rate $\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$.

Compared to other choices of descent direction, such as Newton or Quasi-Newton descent, the Steepest descent has the advantage of having a low computational cost because it requires only the computation of the gradient (it is a first-order algorithm).
However, it has a low convergence, particularly for problems with high-condition numbers.

Backtracking procedure

The Armijo's rule is usually combined with the **backtracking procedure**:
Starting with a candidate α, until the Armijo's rule is not satisfied we reduce α by a decaying factor $\rho \in]0, 1[$, i.e.

$$\text{While } f(\boldsymbol{x}_k + \alpha_k p_k) > f(\boldsymbol{x}_k) + c_1 \alpha \nabla f(\boldsymbol{x}_k)^T \mathbf{p}_k$$
$$\alpha = \rho\alpha.$$

If the backtracking step fails, it means that the $\overline{\alpha}$ we are looking for is too small, indeed the α_k satisfying the Armijo condition yields stagnation. By construction, the backtracking technique avoids small step size, so we do not need to verify also the Wolfe condition.
As initial stepsize, we se

$$\alpha_0 = \frac{\mathbf{s}_k^T \mathbf{s}_k}{\mathbf{s}_k^T \mathbf{y}_k} = \frac{(\boldsymbol{x}_k - \boldsymbol{x}_{k-1})^T (\boldsymbol{x}_k - \boldsymbol{x}_{k-1})}{(\boldsymbol{x}_k - \boldsymbol{x}_{k-1})^T (\nabla f(\boldsymbol{x}_k) - \nabla f(\boldsymbol{x}_{k-1}))}$$

To sum up, the final Steepest Descent algorithm with inexact line search is the following:

Algorithm 1 Steepest Descent algorithm

Require: $\boldsymbol{x}, f, \nabla f, k_{max}, tol > 0, c_1 > 0, \rho > 0$
Ensure: $\boldsymbol{x}^*$ *s.t.* $\boldsymbol{x}^* = \operatorname{argmin} f(\boldsymbol{x})$
  for $k = 0, 1, ...k_{max}$ **do**
    $\mathbf{p} = -\nabla f(\boldsymbol{x})$
    $\mathbf{x}_{try} = \boldsymbol{x} + \alpha \mathbf{p}$
    while $f(\boldsymbol{x}_{try}) > f(\boldsymbol{x}) + c_1 \alpha \nabla f(\boldsymbol{x})^T \mathbf{p}$ **do** ▷ Armijo condition
      $\alpha = \rho\alpha$ ▷ Backtraking procedure
      $\boldsymbol{x}_{try} = \boldsymbol{x} + \alpha \mathbf{p}$
    end while
    $\boldsymbol{x} = \boldsymbol{x}_{try}$
    if $||\nabla f(\boldsymbol{x})|| \leq tol$ **then**
      return $\boldsymbol{x}$
    end if
  end for

Remark 3.1.7. The Steepest Descent method doesn't require the convexity of the objective function to converge to a stationary point, however, in the convex case, this corresponds to a global minimum, while in the nonconvex case, it can be also a local minimum.
Unfortunately, our multi-reference objective function is nonconvex, therefore we are not able to avoid convergence to local minima.

3.2 Proximal Gradient Method

We now consider the composite optimization problem shown in $(\mathcal{B})$. For these kind of models, the steepest descent method is not well-suitable for the following reasons:

- it requires differentiability of the function, in composite optimization problems often one term is non-smooth;
- convergence is slow and it can have zigzagging movements;
- tends to stop in local minima, especially for non-convex problems;
- it requires a lot of iterations therefore it becomes inefficient for large-scale problems.

For these reasons, we will introduce another class of algorithms, that are called *Proximal Gradient* [40].

3.2.1 Preliminary theory

Singular values decomposition

Definition 3.2.1. Given $X \in \mathbb{C}^{m\times n}$, $q = \min\{m, n\}$. Then exist a matrix $\Sigma = \begin{pmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{pmatrix} \in \mathbb{R}^{m\times n}$ such that $\Sigma_1 = Diag(\sigma_1, ...\sigma_q)$ with $\sigma_1 \geq \cdots \geq \sigma_q \geq 0$ and two unitary matrices $U \in \mathbb{C}^{m\times m}$, $V \in \mathbb{C}^{n\times n}$ such that

$$X = U\Sigma V^*.$$

This is called **singular value decomposition** of X and σ_i for $i = 1, ..q$ are called **singular values**.

Remark 3.2.2. Some important properties of the singular values decomposition.

- the columns of U and V are left and right singular vectors;

- the singular values and the eigenvalues of a matrix are strictly related: $\sigma_i = \sqrt{\lambda_i(XX^*)}$;
- $||X||_F^2 = \sum_i \sigma_i^2$, $||X||_2 = \sigma_1$, $k(X) = \frac{\sigma_1}{\sigma_q}$;
- If X is real, then U and V are orthogonal.

Proximal operator

Definition 3.2.3. A function $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is said to be **proper** if $\exists \boldsymbol{x} \in \mathbb{R}^n$ such that $f(\boldsymbol{x}) \neq +\infty$.

Definition 3.2.4. The **effective domain** of the function f is defined:

$$dom(f) = \{\boldsymbol{x} \in \mathbb{R}^n \mid f(\boldsymbol{x}) < +\infty\}.$$

Definition 3.2.5. If $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is closed, proper, convex, extended value function is equivalent to say that its epigraph

$$epi(f) = \{(\boldsymbol{x}, t) \in \mathbb{R}^n \times \mathbb{R} \mid f(\boldsymbol{x}) \leq t\}$$

is a nonempty, closed, and convex set.

Definition 3.2.6. Consider $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ a closed, proper, convex, extended value function. We define the **proximal operator** of f at $\boldsymbol{x}$ with parameter $\lambda > 0$ as the map $prox_{\lambda f} : \mathbb{R}^n \to \mathbb{R}$ such that

$$prox_{\lambda f}(\boldsymbol{x}) := \underset{\mathbf{y} \in \mathbb{R}^n}{\operatorname{argmin}} \left\{ f(\mathbf{y}) + \frac{1}{2\lambda} ||\mathbf{y} - \boldsymbol{x}||_2^2 \right\}.$$

$prox_{\lambda f(\boldsymbol{x})}$ is a point that compromises between minimizing f and being near to $\boldsymbol{x}$. λ can be interpreted as a trade-off between these two terms.
For this reason the $prox_{\lambda f(\boldsymbol{x})}$ is said to be a proximal point of $\boldsymbol{x}$ with respect to f.

In Figure 3.5 we can see the graphical interpretation of this formula. Given a convex function f, the thick black line represents the boundary of its domain, while the thin lines are the level curves. Evaluating the proximal map at the blue points, move them to the red ones. In particular:

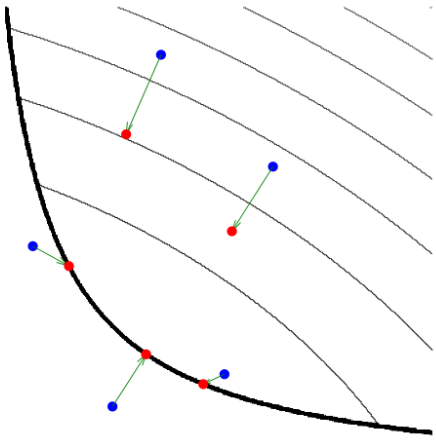

Figure 3.5: Graphical interpretation of the evaluation of a proximal map at various points (Parikh, Boyd [40]).

- if the blue point is inside the domain, it remains there and moves toward the minimum of f;
- if the point is outside the domain, it moves to the boundary of the domain and toward the minimum of f.

The constant λ controls the extent to which the proximal operator maps points to the minimum.

Remark 3.2.7. The proximal operator can be interpreted as a gradient step for f: for λ small and f differentiable we can say that $prox_{\lambda g}(\mathbf{v}) \approx \mathbf{v} - \lambda \nabla g(\mathbf{v})$. This is an important property because it suggests a close relation between proximal maps and gradient methods with λ having the same role of step size in a gradient method.

Theorem 3.2.8. *If $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ a closed, proper, and convex function, then the proximal map of f at $\boldsymbol{x} \in \mathbb{R}^n$ exists and it is unique. Moreover*

$$prox_{\lambda f}(\boldsymbol{x}^*) = x^* \;\Leftrightarrow\; \boldsymbol{x}^* \text{ minimizes } f.$$

Proposition 3.2.9. *Assume f is block separable, i.e. it can be written as*

$f(\boldsymbol{x}, \boldsymbol{y}) = f_1(\boldsymbol{x}) + f_2(\boldsymbol{y})\ \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$. *Then for* $\lambda > 0$

$$prox_{\lambda f}(\boldsymbol{v}, \boldsymbol{w}) = (prox_{\lambda f_1}(\boldsymbol{v}), prox_{\lambda f_2}(\boldsymbol{w})),\ \forall \boldsymbol{v}, \boldsymbol{w} \in \mathbb{R}^n.$$

As a consequence, it holds:

$$\text{if } f(\boldsymbol{x}) = \sum_{i=1}^{n} f_i(\boldsymbol{x}_i)\ \Rightarrow\ (prox_f(\boldsymbol{v}))_i = prox_{f_i}(\boldsymbol{v}_i).$$

This proposition says that the proximal operator of a separable function is equivalent to the proximal operator of each separable part, independently. This property is very important because it allows to simplify calculations.

Example 3.2.10 (Vector Shrinkage Operator: $f(\boldsymbol{x}) = ||\boldsymbol{x}||_1$)**.** We want to compute the proximal map of the ℓ_1 norm. By definition:

$$f(\boldsymbol{x}) = ||\boldsymbol{x}||_1 = \sum_{i=1}^{n} |x_i|.$$

Then, by the separable property, we compute the proximal map of the absolute value, that is:

$$prox_{\lambda|x_i|}(\boldsymbol{x}) = \begin{cases} \boldsymbol{x} - \lambda & \text{if } x_i > \lambda \\ \boldsymbol{x} + \lambda & \text{if } x_i < -\lambda \\ 0 & \text{if } |x_i| \leq \lambda \end{cases} := soft_{[-\lambda,\lambda]}(x_i).$$

This operator is called **soft thresholding** or **shrinkage operator**.
In conclusion:

$$prox_{\lambda||\boldsymbol{x}||_1} = (soft_{[-\lambda,\lambda]}(x_i))_{i=1,\dots n} = \max\{|\boldsymbol{x}| - \lambda, 0\} sgn(\boldsymbol{x})$$

where all operations are component-wise.

Example 3.2.11 (Matrix Shrinkage Operator: $f(X) = ||X||_*$)**.** Consider the singular values decomposition of X: $X = UDiag(\boldsymbol{\sigma})V^T$, where $\sigma_i > 0$
By definition, the nuclear norm can be written as:

$$f(X) = ||X||_* = ||\boldsymbol{\sigma}||_1.$$

Then, by the proximal map of the ℓ_1 norm we have:

$$prox_f(X) = UDiag(soft_{[-\lambda,\lambda]}(\boldsymbol{\sigma}))V^T := soft_{[-\lambda,\lambda]}(X)$$

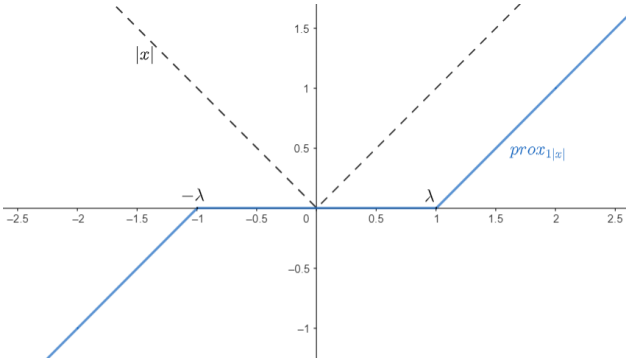

Figure 3.6: Plot of the proximal map of the absolute value.

3.2.2 Composite Optimization

An optimization problem is said to be composite if it has the following form:

$$\min_{x \in \mathbb{R}^n} F(\boldsymbol{x}) = f(\boldsymbol{x}) + g(\boldsymbol{x}) \tag{3.2}$$

where

(A) $g : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ which is closed, proper and convex

(B) $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ which is closed, proper, L smooth over the convex set $dom(f)$ and $dom(g) \subseteq int(dom(f))$.

Proximal Gradient Method

Approximate the function $f(\boldsymbol{x})$ by the following quadratic model:

$$q_\alpha(\boldsymbol{x}, \mathbf{y}) = f(\mathbf{y}) + \langle \boldsymbol{x} - \mathbf{y}, \nabla f(\mathbf{y}) \rangle + \frac{1}{2\alpha} ||\boldsymbol{x} - \mathbf{y}||_2^2.$$

The first part represents a linear approximation of f at some point $\mathbf{y}$, the second part is a quadratic proximal term measuring the local error in the approximation. This is equivalent to:

$$q_\alpha(\boldsymbol{x}, \mathbf{y}) = f(\mathbf{y}) + \frac{1}{2\alpha} ||\boldsymbol{x} - (\mathbf{y} - \alpha \nabla f(\mathbf{y}))||_2^2 - \frac{\alpha}{2} ||\nabla f(\mathbf{y})||_2^2.$$

Now, we consider the general composite model. The function $F(\boldsymbol{x}) = f(\boldsymbol{x}) + g(\boldsymbol{x})$ can be approximated by:

$$Q_\alpha(\boldsymbol{x}, \mathbf{y}) = f(\mathbf{y}) + \frac{1}{2\alpha} ||\boldsymbol{x} - (\mathbf{y} - \alpha \nabla f(\mathbf{y}))||_2^2 - \frac{\alpha}{2} ||\nabla f(\mathbf{y})||_2^2 + g(\boldsymbol{x}).$$

We set $\mathbf{y} = \boldsymbol{x}_{k-1}$, the composite problem (3.2) has the unique solution:

$$\begin{aligned}\boldsymbol{x}_k &= \underset{\boldsymbol{x}\in\mathbb{R}^n}{\operatorname{argmin}}\{Q_\alpha(\boldsymbol{x}, \boldsymbol{x}_{k-1})\} \\ &= \underset{\boldsymbol{x}\in\mathbb{R}^n}{\operatorname{argmin}} \left\{ f(\boldsymbol{x}_{k-1}) + \frac{1}{2\alpha}||\boldsymbol{x} - (\boldsymbol{x}_{k-1} - \alpha\nabla f(\boldsymbol{x}_{k-1}))||_2^2 - \frac{\alpha}{2}||\nabla f(\boldsymbol{x}_{k-1})||_2^2 + g(\boldsymbol{x}) \right\}.\end{aligned}$$

But $f(\boldsymbol{x}_{k-1})$ is constant, then the problem reduces to:

$$\boldsymbol{x}_k = \underset{\boldsymbol{x}\in\mathbb{R}^n}{\operatorname{argmin}} \left\{ g(\boldsymbol{x}) + \frac{1}{2\alpha}||\boldsymbol{x} - (\boldsymbol{x}_{k-1} - \alpha\nabla f(\boldsymbol{x}_{k-1}))||_2^2 \right\}.$$

Therefore, by definition of proximal map, we obtain.

$$\boldsymbol{x}_k = prox_{\alpha g}(\boldsymbol{x} - (\boldsymbol{x}_{k-1} - \alpha\nabla f(\boldsymbol{x}_{k-1})).$$

The **Proximal gradient method** has the following iterations.
Given the step size $\alpha > 0$,

$$\begin{aligned}\mathbf{z}_{k+1} &= \boldsymbol{x}_k - \alpha\nabla f(\boldsymbol{x}_k) && \text{(forward step)} \\ \boldsymbol{x}_{k+1} &= prox_{\alpha g}(\mathbf{z}_{k+1}) && \text{(backward step)}\end{aligned}$$

Remark 3.2.12. We can observe that the backward step doesn't require the gradient of g, indeed only the function f has to be differentiable.

Convergence analysis for the non-convex case.

For this section, for every result, we assume that f and g satisfy the properties (A) and (B) of (3.2). Moreover, we set

$$\overline{\boldsymbol{x}} := prox_{\alpha g}(\boldsymbol{x} - \alpha\nabla f(\boldsymbol{x})).$$

Definition 3.2.13. We define the **gradient mapping** as the operator $G_\alpha : int(dom(f)) \to \mathbb{R}^n$ that is,

$$G_\alpha(\boldsymbol{x}) = \frac{1}{\alpha}(\boldsymbol{x} - \overline{\boldsymbol{x}}), \quad \forall \boldsymbol{x} \in int(dom(f)).$$

With the definition of the gradient mapping, the iteration of the proximal gradient method becomes:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha G_{\alpha_k}(\boldsymbol{x}_k).$$

Lemma 3.2.14 (Descent Lemma). $\forall \boldsymbol{x} \in int(dom(f))$ *and* $\alpha \leq \frac{2}{L}$*:*

$$F(\overline{\boldsymbol{x}}) \leq F(\boldsymbol{x}) + \alpha^2 \left(\frac{1}{\alpha} - \frac{L}{2}\right) ||G_\alpha(\boldsymbol{x})||_2^2.$$

The gradient mapping operation can be seen as a generalization of the gradient operator, indeed we have the following result.

Theorem 3.2.15. *Let* $\frac{1}{\alpha} > 0$*, then:*

1. *if* $g \equiv 0$ *then* $G_\alpha(\boldsymbol{x}) = \nabla f(\boldsymbol{x})$ $\forall \boldsymbol{x} \in int(dom(f))$*,*
2. *for* $\boldsymbol{x}^* \in int(dom(f))$*,* $G_\alpha(\boldsymbol{x}^*) = 0$ *if and only if* $\boldsymbol{x}^*$ *is a stationary point of the composite problem (3.2).*

The quantity $||G_\alpha(\boldsymbol{x})||_2$ can be seen as a **criticality measure** because it is nonnegative and equal to zero if and only if $\boldsymbol{x}$ is a stationary point.

As for the steepest descent algorithm, we can make two choices for the step size. If the Lipshitz constant of ∇f is easy to compute we can use constant step size: $\alpha_k = \alpha < \frac{2}{L}$ $\forall k$.
Otherwise, we apply a backtracking procedure: given $\rho \in]0, 1[$ and $c_1 \in]0, 1[$, starting with a candidate $\alpha > 0$,

$$\begin{aligned} &\text{While } F(\overline{\boldsymbol{x}}_k) > F(\boldsymbol{x}_k) - c_1\alpha_k ||G_{\alpha_k}(\boldsymbol{x}_k)||_2^2, \\ &\quad \text{Set } \alpha = \rho\alpha. \end{aligned} \tag{3.3}$$

Equation (3.3) is called **Armijo-like condition**.
In fact if $g = 0$, we have that $\overline{\boldsymbol{x}} = \boldsymbol{x} - \alpha\nabla f(\boldsymbol{x})$ and $G_\alpha(\boldsymbol{x}) = \nabla f(\boldsymbol{x})$ by the previous Theorem (3.2.15). Substituting in (3.3) we obtain:

$$f(\boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k)) > f(\boldsymbol{x}_k) - c_1\alpha_k ||\nabla f(\boldsymbol{x}_k)||_2^2$$

which is equivalent to the Armijo condition defined in (3.1).

Remark 3.2.16. We note that under the assumption of 3.2 the Descent Lemma implies that the backtracking procedure is finite. In particular if

$$\alpha < \frac{2(1 - c_1)}{L} \quad \Rightarrow \quad c_1 \leq \frac{1 - \alpha L}{2},$$

then by the Descent Lemma, the backtracking procedure ends when

$$\alpha_k \leq \frac{2(1-c_1)}{L}.$$

Theorem 3.2.17 (Convergence). *Consider the sequence $\{\boldsymbol{x}_k\}$ generated by the Proximal Gradient method either with constant step size or chosen by the backtracking procedure. Then the following hold:*

1. *$\{F(\boldsymbol{x}_k)\}_{k\geq 0}$ is nonincreasing;*
2. *$\lim_{k\to\infty} G_{\tilde{\alpha}} = 0$ where $\tilde{\alpha}$ is the chosen step size;*
3. *$\min_{n=0,\dots k} ||G_{\tilde{\alpha}}(\boldsymbol{x}_n)||_2 \leq \sqrt{\frac{F(\boldsymbol{x}_0) - F_{opt}}{M(k+1)}}$ for some $M > 0$;*
4. *all limit points of $\{\boldsymbol{x}_k\}$ are stationary points of the composite problem (3.2).*

The third point of the theorem implies that

$$||G_{\tilde{\alpha}}(\boldsymbol{x}_k)||_2 \approx \mathcal{O}\left(\frac{1}{\sqrt{k}}\right).$$

If we compare the Proximal Gradient method for the nonconvex case to the Steepest Descent method, we can observe that they have the same rate of convergence.

Convergence analysis for the convex case.

To the assumptions of (3.2), we add now the requirement of the convexity for the function f. The following theory is for this convex case only.

The main change is applied to the choice of the step size.
For the constant step size, we apply $\alpha = L$. Otherwise, we apply a backtracking procedure: given $\rho \in]0,1[$ and $c_1 \in]0,1[$, starting with a candidate $\alpha > 0$,

$$\text{While } f(\overline{\boldsymbol{x}_k} > f(\boldsymbol{x}_k) + \nabla f(\boldsymbol{x}_k)^T(\overline{\boldsymbol{x}_k} - \boldsymbol{x}_k) + \frac{1}{2\alpha}||\overline{\boldsymbol{x}_k} - \boldsymbol{x}_k||_2^2$$

$$\text{Set } \alpha = \rho\alpha.$$

Theorem 3.2.18. *Let $\{\boldsymbol{x}_k\}$ be the sequence generated by the Proximal Gradient method either with a constant step size or with a stepsize chosen by the backtracking procedure, then the method has a sublinear rate of convergence. If $\alpha \in]0, \frac{1}{L}]$, $\forall k \geq 1$:*

$$F(\boldsymbol{x}_k) - F(\boldsymbol{x}^*) \leq \frac{\sigma ||\boldsymbol{x}_0 - \boldsymbol{x}^*||_2^2}{2\alpha k} = \mathcal{O}\left(\frac{1}{k}\right),$$

where $\sigma = 1$ if the stepsize is constant.
Moreover

- *$\forall$ optimal solution $\boldsymbol{x}^*$ and $\forall k \geq 1$ we obtain the **Fejer monotonicity**, i.e.:*

$$||\boldsymbol{x}_k - \boldsymbol{x}^*||_2 \leq ||\boldsymbol{x}_{k-1} - \boldsymbol{x}^*||_2;$$

- *the sequence converges to an optimal solution.*

Remark 3.2.19. In particular to obtain an ε-optimal solution, i.e. $F(\boldsymbol{x}_k) - F(\boldsymbol{x}^*) \leq \varepsilon$, are required at most $\dfrac{C}{\varepsilon}$ iterations where

$$C = \frac{\sigma L ||\boldsymbol{x}_0 - \boldsymbol{x}^*||_2^2}{2}.$$

The convergence rate of the proximal gradient method in the convex case is improved with respect to the steepest descent method and to the proximal gradient in the nonconvex case: the rate was $\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$ for both of them. This is not particularly surprising, also because we add other assumptions on the objective function.

Accelerated proximal gradient

The proximal gradient method can be accelerated to obtain a $\mathcal{O}(\frac{1}{k^2})$ convergence rate. This alternative algorithm is called **accelerated proximal gradient method** and consists of the adding of a momentum term to the computation of the new iterate.

Set $t_1 = 1$, then $\forall k \geq 1$

$$\begin{aligned} t_{k+1} &= \frac{1 + \sqrt{1 + 4t_k^2}}{2} \\ \mathbf{y}_k &= \boldsymbol{x}_k + \frac{t_k - 1}{t_{k+1}}(\boldsymbol{x}_k - \boldsymbol{x}_{k-1}) \\ \mathbf{z}_{k+1} &= \mathbf{y}_k - \alpha \nabla f(\mathbf{y}_k) \\ \boldsymbol{x}_{k+1} &= prox_{\alpha g}(\mathbf{z}_{k+1}). \end{aligned}$$

We observe that at the first iteration, $\mathbf{y}_1 = \boldsymbol{x}_0$, therefore it is equivalent to applying the proximal gradient method.
During the first iterations, the current iteration mostly dictates the direction of the next step. As we are reaching the optimum, i.e. the gradient of f is near zero, the momentum pushes more in the direction we are going, represented by the difference of the two successive iterations $\boldsymbol{x}_k$ and $\boldsymbol{x}_{k-1}$. This allows us to speed up the method.

The computational complexity of the algorithm remains the same because we have to compute the gradient of f and the proximal map as before, with the main difference that they are computed on $\mathbf{y}_k$, instead of $\boldsymbol{x}_k$. The two additional computations of t_k and $\mathbf{y}_k$ have a marginal cost.

The backtracking technique for searching the stepsize is the same as the proximal gradient method, but we substitute $\boldsymbol{x}_k$ with $\mathbf{y}_k$.

$$\text{While } f(\overline{\mathbf{y}_k}) > f(\mathbf{y}_k) + \nabla f(\mathbf{y}_k)^T(\overline{\mathbf{y}_k} - \mathbf{y}_k) + \frac{1}{2\alpha}||\overline{\mathbf{y}_k} - \mathbf{y}_k||_2^2$$

$$\text{Set } \alpha = \rho\alpha.$$

Convergence Analysis

Theorem 3.2.20. *Let $\{\boldsymbol{x}_k\}$ be the sequence generated by the accelerated proximal gradient method either with a constant step size or with a stepsize chosen by the backtracking procedure. Then if $\alpha \in]0, \frac{1}{L}]$, $\forall k \geq 1$:*

$$F(\boldsymbol{x}_k) - F(\boldsymbol{x}^*) \leq \frac{2\sigma||\boldsymbol{x}_0 - \boldsymbol{x}^*||_2^2}{\alpha(k+1)^2} = \mathcal{O}\left(\frac{1}{k^2}\right),$$

where $\sigma = 1$ if the stepsize is constant.

Remark 3.2.21. To obtain an ε-optimal solution are required at most $\frac{C}{\sqrt{\varepsilon}} - 1$ iterations with

$$C = \sqrt{2\sigma L||\boldsymbol{x}_0 - \boldsymbol{x}^*||_2^2}.$$

This is only one possibility to define the iteration of t_{k+1}, other alternatives can be used.

Alternating Method of Multipliers

Another algorithm that is often used to solve composite optimization models is the **Alternating Method of Multipliers** (**ADMM**).
Consider a two blocks optimization problem:

$$\begin{aligned} &\text{minimize} \quad f(\mathbf{x}) + g(\mathbf{y}) \\ &\text{subject to} \quad A\mathbf{x} + B\mathbf{y} = \mathbf{c} \end{aligned}$$

where $\mathbf{x} \in \mathbb{R}^n,\ \mathbf{y} \in \mathbb{R}^m,\ A \in \mathbb{R}^{p\times n}\ B \in \mathbb{R}^{p\times m}\ \mathbf{c} \in \mathbb{R}^p$.
Assume $f : \mathbb{R}^n \to \mathbb{R} \cap \{+\infty\}$ and $g : \mathbb{R}^m \to \mathbb{R} \cap \{+\infty\}$ are both closed, proper and convex.
Consider the augmented Lagrangian function.

$$\mathcal{L}_\beta(\mathbf{x}, \mathbf{y}; \boldsymbol{\lambda}) = f(\mathbf{x}) + g(\mathbf{y}) + \langle \boldsymbol{\lambda}, A\boldsymbol{x} + B\mathbf{y} - \mathbf{c}\rangle + \frac{\beta}{2}||A\boldsymbol{x} + B\mathbf{y} - \mathbf{c}||_2^2$$

where $\boldsymbol{\lambda} \in \mathbb{R}^p$ is the lagrangian vector and $\beta \in \mathbb{R}^+$ is the penalty parameter.
Resolving the minimization problem is equivalent to searching for the saddle points of $\mathcal{L}_\beta$:

find $\{\mathbf{x}^*, \mathbf{y}^*, \boldsymbol{\lambda}^*\}$ s.t. $\mathcal{L}_\beta(\mathbf{x}^*, \mathbf{y}^*, \boldsymbol{\lambda}) \leq \mathcal{L}_\beta(\mathbf{x}^*, \mathbf{y}^*, \boldsymbol{\lambda}^*) \leq \mathcal{L}_\beta(\mathbf{x}, \mathbf{y}, \boldsymbol{\lambda}^*) \quad \forall(\mathbf{x}, \mathbf{y}, \boldsymbol{\lambda})$.

ADMM consists of the following iterations:

$$\begin{aligned} \mathbf{x}_{k+1} &= \operatorname*{argmin}_{\mathbf{x}\in\mathbb{R}^n} \mathcal{L}_\beta(\mathbf{x}, \mathbf{y}_k; \boldsymbol{\lambda}_k) \\ \mathbf{y}_{k+1} &= \operatorname*{argmin}_{\mathbf{y}\in\mathbb{R}^m} \mathcal{L}_\beta(\mathbf{x}_{k+1}, \mathbf{y}; \boldsymbol{\lambda}_k) \\ \boldsymbol{\lambda}_{k+1} &= \boldsymbol{\lambda}_k + \beta(A\mathbf{x}_{k+1} + B\mathbf{y}_{k+1} - \mathbf{c}). \end{aligned}$$

It consists of two minimization steps called *primal descent* and an update of $\boldsymbol{\lambda}$ called *dual ascent* which uses a step size equal to the penalty parameter β.

3.2.3 Vectorial Total Variation regularizer

We have to solve the following model:

$$\mathbf{v}^* = \underset{\mathbf{v}\in\mathbb{R}^{h\times w\times 2}}{\operatorname{argmin}} \{-f(\mathbf{v}) + \mu\mathcal{R}(\mathbf{v})\}. \tag{3.4}$$

In this project, we consider a Vectorial Total Variation Regularization. This regularizer is well-defined also for images with sharp discontinuities and we think it can be the most suitable for our scenario.

Definition 3.2.22. Given a matrix A, we define the **p-Schatten norm** of A as

$$||A||_{\mathcal{S}_p} = \left(\sum_{n=1}^{r+1} \sigma_n^p\right)^{\frac{1}{p}}$$

with $\sigma_1, ..\sigma_{r+1}$ singular values of A and $r = rank(A)$.

Remark 3.2.23. In particular:

- if $p = 1$ this is equivalent to the nuclear norm,
- if $p = 2$ to the Frobenius norm.

Definition 3.2.24. Consider $\mathbf{v} = (v^{(1)}, v^{(2)}) \in \mathbb{R}^{h\times w\times 2}$, $\mathbf{D} = (D_h, D_v) : \mathbb{R}^{h\times w\times 2} \to \mathbb{R}^{h\times w\times 4}$ a finite differences operator.
The **Vectorial Total Variation regularizer** is defined as:

$$\mathcal{R}(\mathbf{v}) = \sum_{i=1}^{N_p=wh} ||(\mathbf{Dv})_i||_{\mathcal{S}_p} = \sum_{i=1}^{N_p} \left\| \begin{pmatrix} (D_h\mathbf{v}^{(1)})_i & (D_v\mathbf{v}^{(1)})_i \\ (D_h\mathbf{v}^{(2)})_i & (D_v\mathbf{v}^{(2)})_i \end{pmatrix} \right\|_{\mathcal{S}_p}, \quad \forall p = 1, 2.$$

In our case, we set $p = 1$.
The problem (3.4) becomes:

$$\mathbf{v}^* = \underset{\mathbf{v}\in\mathbb{R}^{h\times w\times 2}}{\operatorname{argmin}} \{-f(\mathbf{v}) + \mu\mathcal{R}(\mathbf{v})\} = \underset{\mathbf{v}\in\mathbb{R}^{h\times w\times 2}}{\operatorname{argmin}} \left\{ ||\mathbf{D}H||_2^2 + \mu \sum_{i=1}^{N_p} ||(\mathbf{Dv})_i||_* \right\}.$$

We apply the Proximal Gradient Method:

$$\mathbf{w}_{k+1} = v_k - \alpha \nabla f(\mathbf{v}_k)$$
$$\mathbf{v}_{k+1} = prox_{\mu\alpha||\mathbf{D}\cdot||_*}(\mathbf{w}_{k+1}).$$

In this case, the proximal operator has no closed-form solution, because of the presence of $\mathbf{D}$. Indeed, since $\mathbf{D}\mathbf{D}^* \neq c\mathbf{I}$, with $c > 0$, we cannot apply the property of the proximal function, therefore we solve the second iteration with the ADMM algorithm.

By definition, we have:

$$prox_{\mu\alpha||\mathbf{D}\cdot||_*}(\mathbf{w}_{k+1}) = \underset{\mathbf{v}_k}{\operatorname{argmin}} \left\{ \sum_{i=1}^{N_p} ||(\mathbf{D}\mathbf{v}_k)_i||_* + \frac{1}{2\mu\alpha}||\mathbf{v}_k - \mathbf{w}_{k+1}||_2^2 \right\}.$$

We set $\mathbf{v} = \mathbf{v}_k$, $\mathbf{w} = \mathbf{w}_{k+1}$ and we consider the variable splitting

$$\mathbf{t} = \begin{pmatrix} t_{h,1} & t_{v,1} \\ t_{h,2} & t_{v,2} \end{pmatrix} = \begin{pmatrix} D_h\mathbf{v}^{(1)} & D_v\mathbf{v}^{(1)} \\ D_h\mathbf{v}^{(2)} & D_v\mathbf{v}^{(2)} \end{pmatrix} = \mathbf{D}\mathbf{v},$$

then the problem reduces to:

$$\mathbf{v}_{k+1} = \mathbf{v}^* = \underset{\mathbf{v}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{N_p} ||\mathbf{t}_i||_* + \frac{1}{2\mu\alpha}||\mathbf{v} - \mathbf{w}||_2^2 \right\} \text{ s.t. } \mathbf{t} = \mathbf{D}\mathbf{v}.$$

It corresponds to a standard two blocks problem where $f(\mathbf{v}) = ||\mathbf{v} - \mathbf{w}||_2^2$ and $g(\mathbf{t}) = \sum_{i=1}^{N_p} ||\mathbf{t}_i||_*$ are closed, proper and convex.

We consider the associated augmented Lagrangian function with $\boldsymbol{\lambda} = \begin{pmatrix} \boldsymbol{\lambda}_h & \boldsymbol{\lambda}_v \end{pmatrix} \in \mathbb{R}^{2N_p \times 2}$ and $\beta \in \mathbb{R}^+$ penalty parameter.

$$\mathcal{L}(\mathbf{v}, \mathbf{t}, \boldsymbol{\lambda}) = \sum_{i=1}^{N_p} ||\mathbf{t}_i||_* + \frac{1}{2\mu\alpha}||\mathbf{v} - \mathbf{w}||_2^2 - \langle \boldsymbol{\lambda}, \mathbf{t} - \mathbf{D}\mathbf{v} \rangle + \frac{\beta}{2}||\mathbf{t} - \mathbf{D}\mathbf{v}||_2^2.$$

Finding a solution to the minimization problem is equivalent to seeking for the saddle points of $\mathcal{L}$, i.e.

$$\text{find } \{\mathbf{v}^*, \mathbf{t}^*, \boldsymbol{\lambda}^*\} \text{ s.t. } \mathcal{L}(\mathbf{v}^*, \mathbf{t}^*, \boldsymbol{\lambda}) \leq \mathcal{L}(\mathbf{v}^*, \mathbf{t}^*, \boldsymbol{\lambda}^*) \leq \mathcal{L}(\mathbf{v}, \mathbf{t}, \boldsymbol{\lambda}^*) \quad \forall (\mathbf{v}, \mathbf{t}, \boldsymbol{\lambda}).$$

Subproblem for the primal variable $\mathbf{v}$.

We drop the terms not depending on $\mathbf{v}$:

$$\mathbf{v}^{(k+1)} = \underset{\mathbf{v}}{\operatorname{argmin}} \left\{ Z(\mathbf{v}) = \frac{1}{2\mu\alpha}||\mathbf{v} - \mathbf{w}||_2^2 + \langle \boldsymbol{\lambda}^{(k)}, \mathbf{D}\mathbf{v} \rangle + \frac{\beta}{2}||\mathbf{t}^{(k)} - \mathbf{D}\mathbf{v}||_2^2 \right\}.$$

It is quadratic with respect to $\mathbf{v}$, the global minimizers are to be sought among its stationary point, then we impose first order optimality condition:

$$\mathbf{v}^{(k+1)} \in \{\mathbf{v} : \nabla Z(\mathbf{v}) = 0\}$$

i.e.

$$\left(\mathbf{D}^T\mathbf{D} + \frac{1}{\beta\mu\alpha}\mathbf{I}\right)\mathbf{v}^{(k+1)} = \mathbf{D}^T\left(\mathbf{t}^{(k)} - \frac{\lambda^{(k)}}{\beta}\right) + \frac{\mathbf{w}}{\beta\mu\alpha}.$$

The coefficient matrix is symmetric, positive definite, and with full rank, then the solution of the linear system exists and it is unique.

During the implementation, we assume reflexive boundary condition (see Section 2.3.2) therefore the system is simply solved with backslash.

Subproblem for the primal variable $\mathbf{t}$.

As before, we do not consider the terms which don't depend on $\mathbf{t}$:

$$\begin{aligned}\mathbf{t}^{(k+1)} &= \underset{\mathbf{t}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{N_p} ||\mathbf{t}_i||_* - \langle \boldsymbol{\lambda}^{(k)}, \mathbf{t} - \mathbf{D}\mathbf{v}^{(k+1)} \rangle + \frac{\beta}{2}||\mathbf{t} - \mathbf{D}\mathbf{v}^{(k)}||_2^2 \right\} \\ &= \underset{\mathbf{t}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{N_p} ||\mathbf{t}_i||_* + \frac{\beta}{2}||\mathbf{t} - \mathbf{q}^{(k)}||_2^2 \right\}\end{aligned}$$

with $\mathbf{q}^{(k)} = \mathbf{D}\mathbf{v}^{(k+1)} + \dfrac{\boldsymbol{\lambda}^{(k)}}{\beta}$.

The problem reduces to N_p independent minimization problems:

$$\mathbf{t}^{(k+1)} = \underset{\mathbf{t}}{\operatorname{argmin}} \sum_{i=1}^{N_p} \left\{ ||\mathbf{t}_i||_* + \frac{\beta}{2}||\mathbf{t}_i - \mathbf{q}_i^{(k)}||_2^2 \right\}.$$

By the separable property of the proximal map, each independent problem admits a closed-form solution.

So we apply the definition of the proximal map of the nuclear norm (Example 3.2.10). Consider the svd decomposition of $\mathbf{q}^{(k)} = \mathbf{U}\text{Diag}(\mathbf{s})\mathbf{V}^T$, then

$$\mathbf{t}^{(k+1)} = \mathbf{U}\text{Diag}(\mathbf{d})\mathbf{V}^T,$$

where $d = prox_{\frac{1}{\beta}||\cdot||_1}(\mathbf{s})$.

λ's update.

$$\boldsymbol{\lambda}^{(k+1)} = \boldsymbol{\lambda}^{(k)} - \beta(\mathbf{t}^{(k+1)} - \mathbf{D}\mathbf{v}^{(k+1)}).$$

β's update.
Usually, the value of β is set at the start of the algorithm and is kept constant. In our case is difficult to choose the best value because the ADMM algorithm is run at every iteration of the proximal gradient method. Therefore we use the following procedure proposed in [45].
Given $0 < \beta_{\min} < \beta_{\max} < +\infty$

$$\beta^{(k+1)} = (1-\omega)\beta^{(k)} + \omega \text{proj}_{[\beta_{\min},\beta_{\max}]}\left(\frac{||\boldsymbol{\lambda}^{(k+1)}||_2}{||-\mathbf{I}_{2N_p}\mathbf{t}^{(k+1)}||_2}\right) \quad \omega \in [0,1].$$

Therefore the ADMM scheme becomes:

$$\begin{cases} \mathbf{v}^{(k+1)} = \underset{\mathbf{v}}{\text{argmin}} \left\{ Z(\mathbf{v}) = \frac{1}{2\mu\alpha}||\mathbf{v}-\mathbf{w}||_2^2 + \langle \boldsymbol{\lambda}^{(k)}, \mathbf{D}\mathbf{v}\rangle + \frac{\beta}{2}||\mathbf{t}^{(k)} - \mathbf{D}\mathbf{v}||_2^2 \right\} \\ \mathbf{t}^{(k+1)} = \underset{\mathbf{t}}{\text{argmin}} \left\{ \sum_{i=1}^{2N_p} ||\mathbf{t}_i||_* - \langle \boldsymbol{\lambda}^{(k)}, \mathbf{t} - \mathbf{D}\mathbf{v}^{(k+1)}\rangle + \frac{\beta}{2}||\mathbf{t} - \mathbf{D}\mathbf{v}^{(k+1)}||_2^2 \right\} \\ \boldsymbol{\lambda}^{(k+1)} = \boldsymbol{\lambda}^{(k)} - \beta(\mathbf{t}^{(k+1)} - \mathbf{D}\mathbf{v}^{(k+1)}) \\ \beta^{(k+1)} = (1-\omega)\beta^{(k)} + \omega \text{proj}_{[\beta_{\min},\beta_{\max}]}\left(\frac{||\boldsymbol{\lambda}^{(k+1)}||_2}{||-\mathbf{I}_{2N_p}\mathbf{t}^{(k+1)}||_2}\right) \quad \omega \in [0,1]. \end{cases}$$

To sum up, the final algorithm to solve (3.4) is the following.

Algorithm 2 Accelerated Proximal Gradient algorithm + ADMM iteration

Require: $\mathbf{x}, f, \nabla f, k_{max}, tol > 0, \rho > 0$

Ensure: $\mathbf{x}^*$ *s.t.* $\mathbf{x}^* = \text{argmin}\{f(\mathbf{x}) + \mu\mathcal{R}(\mathbf{x})\}$

for $k = 0, 1, ...k_{max}$ **do**

$t_{k+1} = \dfrac{1 + \sqrt{1 + 4t_k^2}}{2}$

$\mathbf{y} = x + \dfrac{t_k - 1}{t_{k+1}}(\mathbf{x}_k - \mathbf{x}_{k-1})$

$\mathbf{z} = \mathbf{y} - \alpha\nabla f(\mathbf{y})$

$\mathbf{x} \to ADMM\ step$

while $f(\mathbf{x}_{try}) > f(\mathbf{y}) + \nabla f(\mathbf{y})^T(\mathbf{x}_{try} - \mathbf{y}) + \dfrac{1}{2\alpha}||\mathbf{x}_{try} - \mathbf{y}||^2$ **do**

$\alpha = \rho\alpha$ ▷ Backtraking procedure

$\mathbf{z} = \mathbf{y} - \alpha\nabla f(\mathbf{y})$

$\mathbf{x}_{try} \to ADMM\ step$

end while

if $\dfrac{||F(\mathbf{x}_{try}) - F(\mathbf{x})||}{||F(\mathbf{x})||} \leq tol$ **then**

return $\mathbf{x}_{try}$

end if

$\mathbf{x} = \mathbf{x}_{try}$

end for

Where the ADMM iteration is:

Algorithm 3 ADMM algorithm

Require: $\mathbf{t}, \mathbf{w}, \beta, k_{max}, tol > 0$

Ensure: $\mathbf{x}^*$ *s.t.* $\mathbf{x}^* = \text{argmin}\{f(\mathbf{x}) + \mu\mathcal{R}(\mathbf{x})\}$

for $k = 0, 1, ...k_{max}$ **do**

$$\textbf{Solve } \left(\mathbf{D}^T\mathbf{D} + \frac{1}{\beta\mu\alpha}\mathbf{I}\right)\mathbf{x}_{try} = \mathbf{D}^T\left(\mathbf{t} - \frac{\lambda}{\beta}\right) + \frac{\mathbf{w}}{\beta\mu\alpha}$$

$$\mathbf{t} = \mathbf{U}\text{Diag}(\mathbf{d})\mathbf{V}^T \text{ with } \mathbf{d} = prox_{\frac{1}{\beta}||\cdot||_1}(\mathbf{s})$$

$$\boldsymbol{\lambda} = \boldsymbol{\lambda} - \beta(\mathbf{t} - \mathbf{D}\mathbf{v}_{try}))$$

$$\beta = (1-\omega)\beta + \omega\text{proj}_{[\beta_{\min},\beta_{\max}]}\left(\frac{||\boldsymbol{\lambda}||_2}{||-\mathbf{I}_{2N_p}\mathbf{t}||_2}\right)$$

if $\dfrac{||\mathbf{x}_{try} - \mathbf{x}||}{||\mathbf{x}_{try}||} \leq tol$ **then**

return $\mathbf{x}_{try}$

end if

$\mathbf{x} = \mathbf{x}_{try}$

end for

Remark 3.2.25. The overall method becomes quite computationally expensive. Indeed at every iteration and, if needed, at every backtracking iteration we apply the ADMM method. Moreover, we need to construct matrices of big dimensions, such as $Np \times N_e$. In order to reduce the cost we apply some strategies:

1. all the quantities not depending on $\mathbf{v}$ and which remain constant during the iterations are computed only one time outside the algorithms (e.g. the matrix $\mathbf{D}$ and $\mathbf{D}\mathbf{D}^T$ and the selection matrix $\mathbf{S}$ for the gradient of f);

2. all the matrices of big dimension, which mostly involve the computation of ∇f, are stored as sparse matrices, if possible, or vectorized;

3. in the ADMM algorithm we consider warm start initialization: the variables $\mathbf{t}$, β, $\boldsymbol{\lambda}$ are not initialized to zero at the start of every iteration. Every time ADMM runs, the results are memorized to be used in the

next iteration. This technique allows us to reduce the cost because the values of the variables we are using are nearer to the minimum;

4. during the iterations of the (Accelerated) Proximal Gradient algorithm we decrease the relative change tolerance for the stopping criteria of the ADMM. Indeed it is not necessary to find the optimum solution of the ADMM in the first iterations of the Proximal Gradient, this strategy permits the reduction of the computational complexity of the algorithm. At the start, the tolerance has a value of 10^{-3}, at every iteration it decreases such that it reaches a value of 10^{-7} near the achievement of the minimum.

Chapter 4

Experimental Results

In this chapter, we are going to test the two models ($\mathcal{A}$) and ($\mathcal{B}$) by applying the Steepest Descent and the Accelerated Proximal Gradient methods, respectively. We will use at first 3 different synthetic datasets and then 2 different datasets of real data.

4.1 Main contributions

Before analyzing in detail the results obtained in the following experiments, we summarize the main contribution given by this project depending on the initial guess.
In the examples with synthetic datasets, we observe that changing the starting guess of the algorithm affects a lot the estimations obtained. Indeed in particular we have that

1. for the starting guess equal to the null velocity, we obtain good results for experiment 1 using a kernel dimension support equal to 13, standard deviation equal to 2, and a camera domain enlargement of 6 pixels. In the second experiment, we obtain worse results, in particular with the application of the Accelerated Proximal Gradient Descent, however, they are better than other starting guesses;

2. for the starting guess with the true value of the velocity at each pixel, as predictable, we obtain the best results for every experiment, indeed in a few iterations we obtain a minimum;

3. for the starting guess equal to the value of one of the two velocities involved and null everywhere else, in general, we obtain estimations that are worse than the previous two cases. In particular, for the second experiment, the algorithm tends to uniform the results to the same velocity;

4. for the starting guess with random velocities values, we obtain the worst results for every experiment and it never converges to a global minimum.

To summarize, we can say that the best results are obtained with an initial guess equal to the correct values of the velocity. For this reason, we will run the algorithm on the real data, setting as a starting guess the result obtained for the case with constant velocity, illustrated in Section 2.4.

4.2 Experiment 1 on synthetic data.

We consider two points moving with constant velocity on the not intersecting straight lines $y = x$ and $y = 2x + 12$, respectively. This generates 15 equispaced events on each of them. Figure 4.1 shows the ground truth of the velocity.
To this set of synthetic data, we apply the Steepest Descent algorithm with inexact line search to the multi-reference focus objective function, reading

$$f(\mathbf{v}) = -\frac{G(\mathbf{v}; t_{min}) + 2G(\mathbf{v}; t_{mid}) + G(\mathbf{v}; t_{max})}{4G(\mathbf{0})}.$$

To highlight the importance of the choice of the starting guess, we run the algorithm with 5 different values of velocities:

1. null velocities at all pixels

2. true value of the velocity at each pixel
3. true value of the velocity on the line $y = x$, null velocities elsewhere
4. true value of the velocity on the line $y = 2x$, null velocities elsewhere
5. velocities component values drawn from the uniform distribution in the interval $[0, 2]$.

We note that the values are imposed only on the pixel location in which there is at least one event since without a regularization term the method cannot change the other values of the velocity.
We evaluate the accuracy of the estimated velocity by the positive scalar metric relative error defined as:

$$E_{rel} = \frac{||\widehat{V} - \overline{V}||_2}{||\widehat{V}||_2} \tag{4.1}$$

where $\widehat{V}$ is the estimated value of the velocity vector field and $\overline{V}$ stands for its true value.
As stopping criterium we use the relative change of the objective function f with tolerance 10^{-16}, i.e.

$$\frac{|f(\mathbf{v}_{k+1}) - f(\mathbf{v}_k)|}{|f(\mathbf{v}_k)|} \leq 10^{-16} \quad \forall \text{ iteration } k = 1, ...k_{max}.$$

In Figure 4.2 we show the plots of the estimated velocities with standard deviation $\sigma = 2$. For what concern the size of the kernel support, for efficiency purposes, as suggested in [38], we adopted a truncated square support with side length 13 pixels (13×13 support).
With the starting guesses equal to 0, to the true velocity, and to the true velocity on the line $y = x$ (i.e. the first three plots), we can see that the method has estimated the correct value with a relative error of $1.58 \cdot 10^{-4}$. This seems to imply that they all converge to the same minimum point (global?).
For the last two plots with a starting point equal to the true velocity on

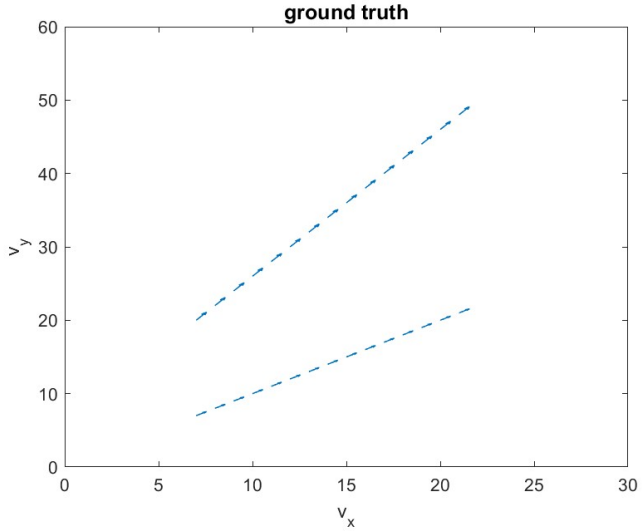

Figure 4.1: Plot of the ground truth velocities of two ideals points 1 and 2 moving with ideal velocities $\mathbf{v}^{(1)} = (v_x, v_y) = (1,1)pixel/s$ and $\mathbf{v}^{(2)} = (v_x, v_y) = (1,2)pixel/s$ along the straight lines $y = x$ and $y = 2x$, plotted on the domain of the event camera of size 60×30.

$y = 2x$ and a random velocity, the error is higher: $3.5 \cdot 10^{-1}$ and $3.3 \cdot 10^{-1}$, respectively. Therefore, we can assume that the algorithm has converged to a local minimum (the objective function is non-convex). As we can see from Figure 4.3 the objectives function decreases monotonically with the number of iterations until the Steepest Descent algorithm converges.

For another visual representation of the results, Figure 4.4, 4.5, 4.6 show the images of the events warped at the reference times t_{min}, t_{mid}, t_{max}, respectively associated to the estimated velocity fields for each starting point, compared with the ground truth. In particular, we used

$$t_{min} = \underline{t} + 0.1(\overline{t} - \underline{t}), \quad t_{mid} = \frac{\underline{t} + \overline{t}}{2}, \quad t_{max} = \overline{t} - 0.1(\overline{t} - \underline{t}).$$

To sum up, in Table 4.1 are reported the values of the relative error, the function G associated with the reference times t_{min}, t_{mid}, t_{max}, of the objective function f, and of the number of iterations varying the starting guesses. We can observe that for the first 3 cases, the value of $f(v)$ is the

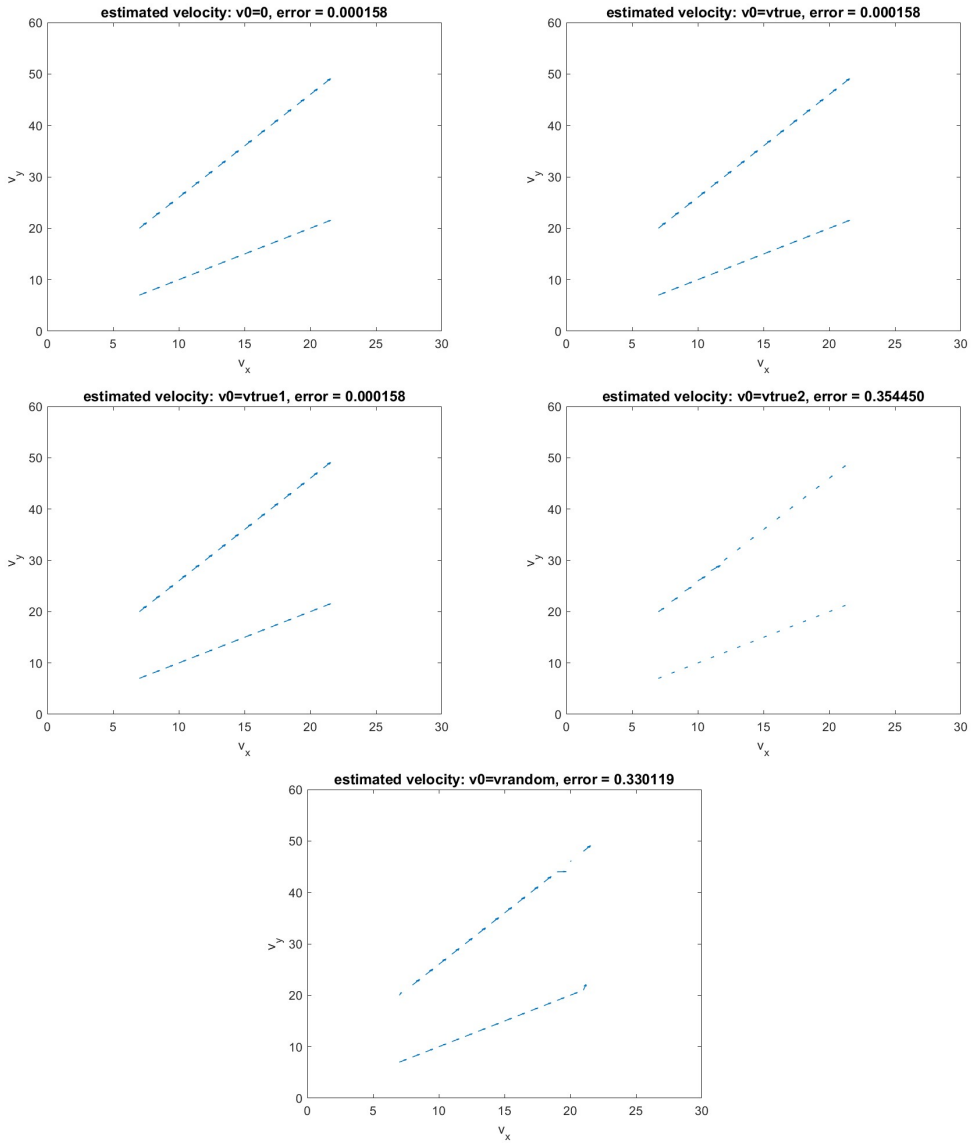

Figure 4.2: Plots of estimated velocity with respect to different choices of starting guesses and associated relative error reported in Table 4.1; with kernel dimension 13×13, standard deviation 2 and camera domain 60×30.

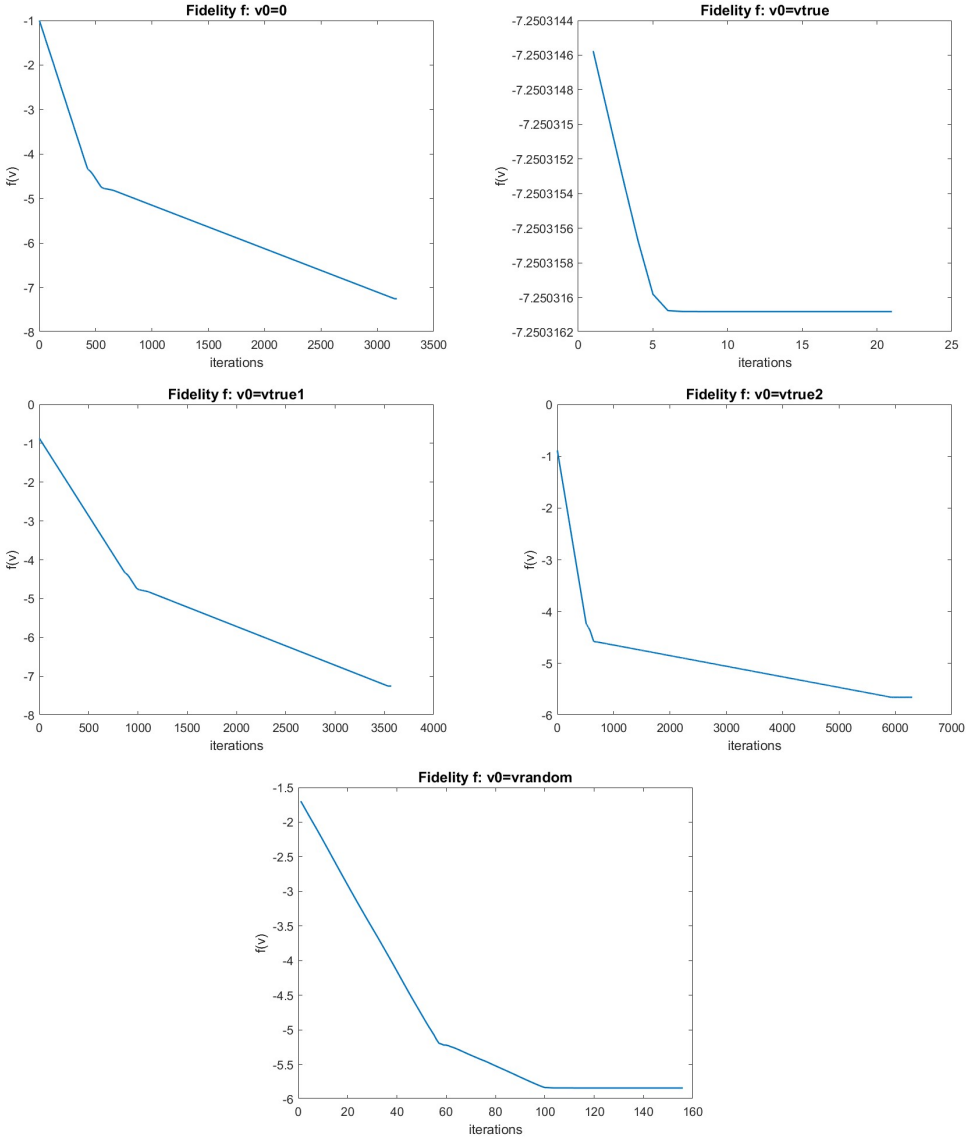

Figure 4.3: Plots of the multi-reference focus function values along the iterations of the Steepest Descent numerical optimization algorithm for different choices of starting guess, with kernel dimension 13×13 and standard deviation 2.

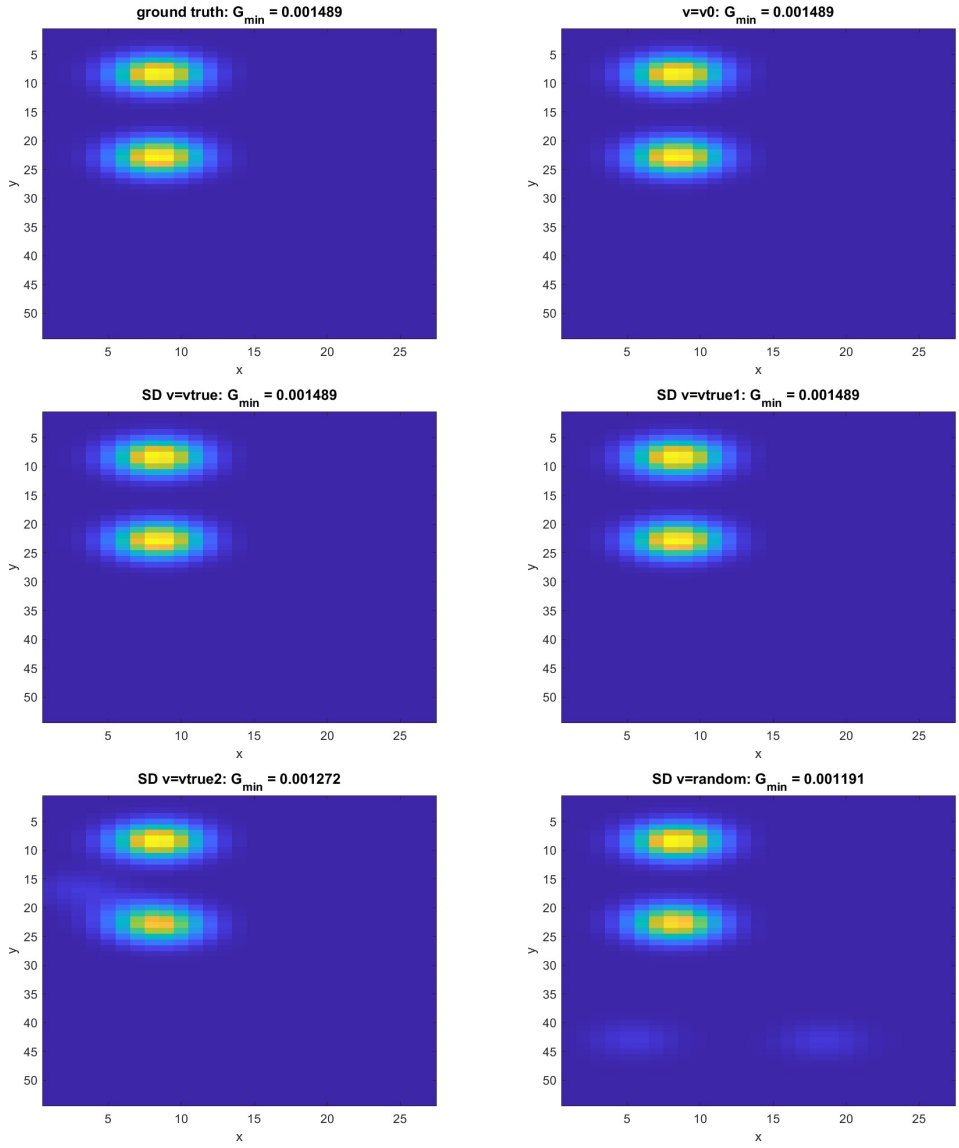

Figure 4.4: Representation of the image $H(i, j)$ of the events warped to the reference time t_{min} by varying the starting guesses and the associated value of the function G reported in Table 4.1, compared to the ground truth. We consider a kernel dimension 13×13 and a standard deviation 2.

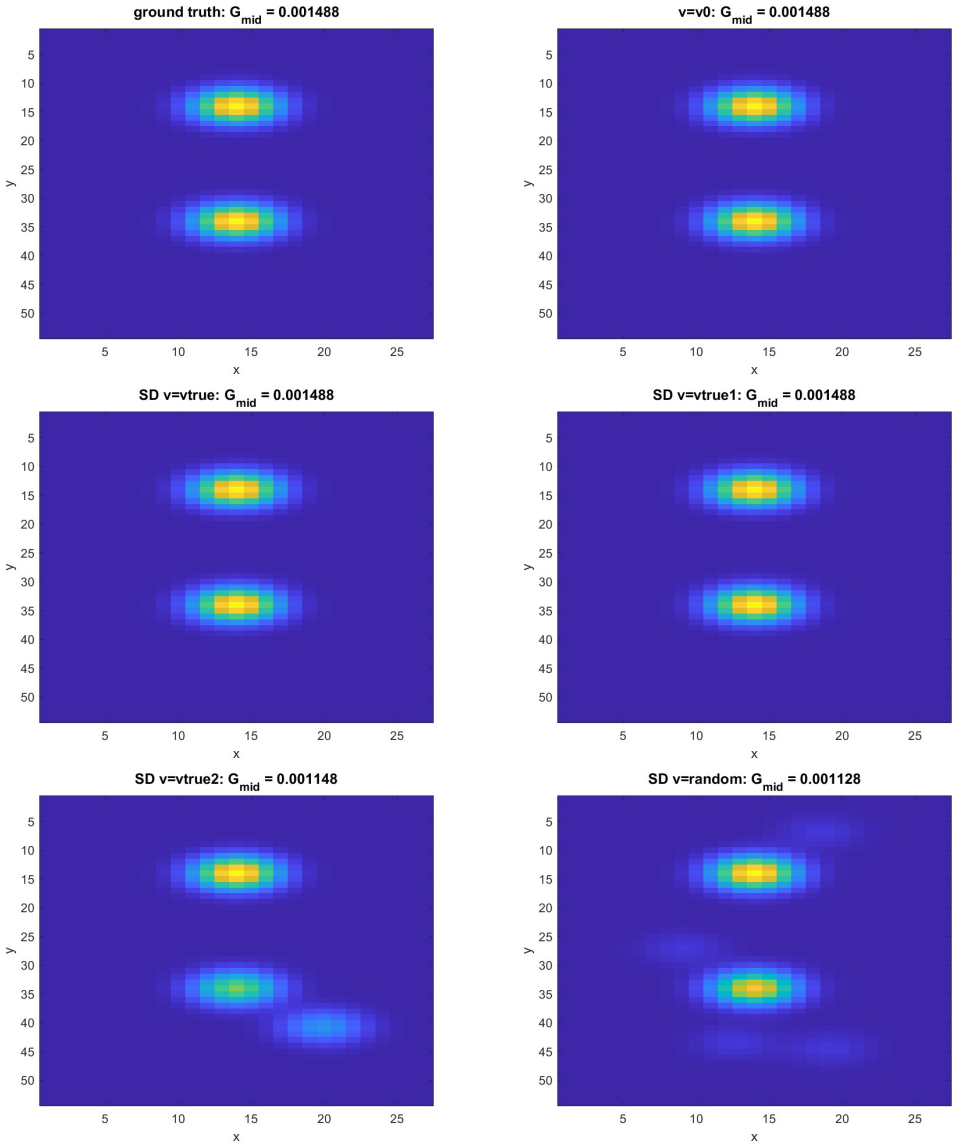

Figure 4.5: Representation of the image $H(i, j)$ of the events warped to the reference time t_{mid} by varying the starting guesses and the associated value of the function G reported in Table 4.1, compared to the ground truth. We consider a kernel dimension 13×13 and a standard deviation 2.

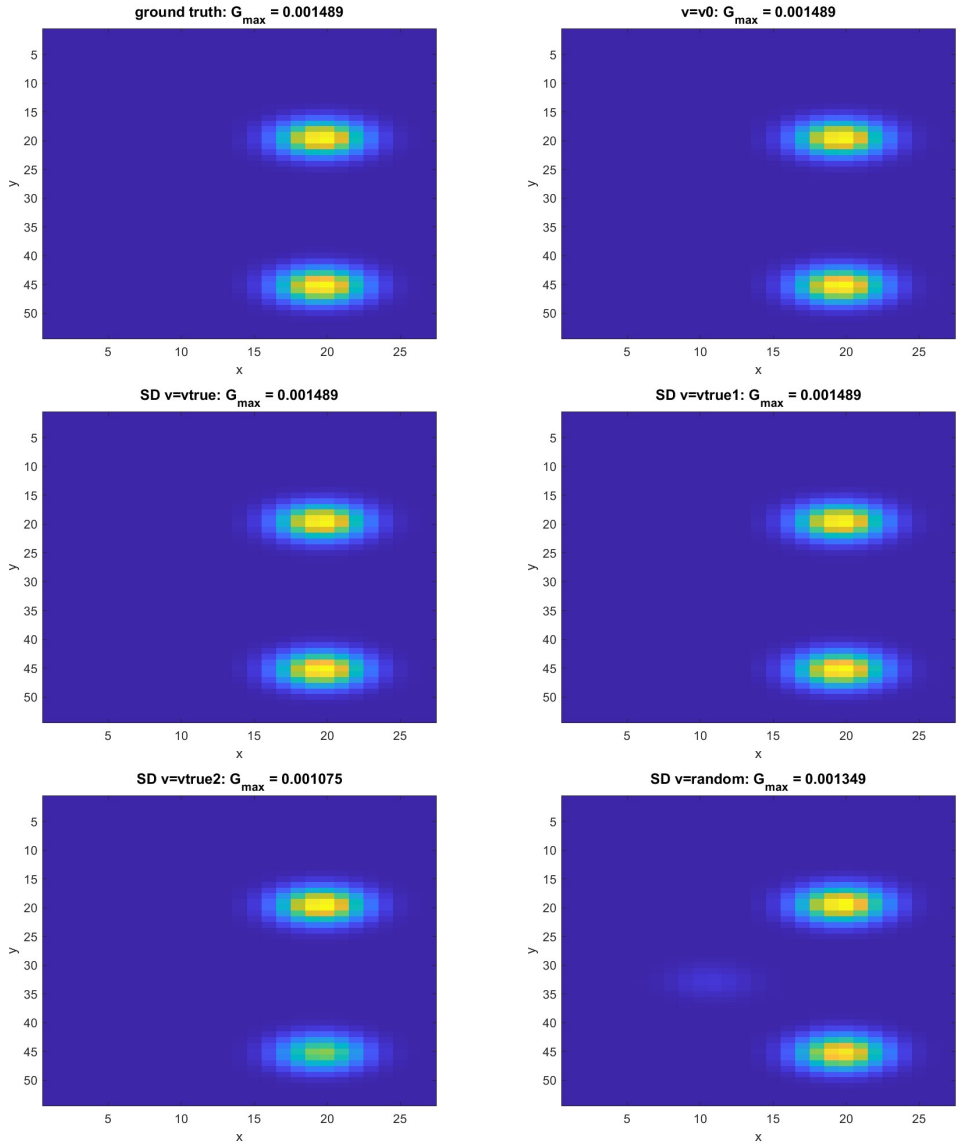

Figure 4.6: Representation of the image $H(i, j)$ of the events warped to the reference time t_{max} by varying the starting guesses and the associated value of the function G reported in Table 4.1, compared to the ground truth. We consider a kernel dimension 13×13 and a standard deviation 2.

same: this confirms that they converged to the same minimum point, while for the other 2 cases, the value is higher. Moreover, it is interesting to note that the case that is slower to converge is the one with a starting guess equal to the correct velocity of $y = 2x$. The cases with an initial guess equal to 0 and to the correct velocity of $y = x$ have more or less the same number of iterations. Not surprisingly, the case with the correct initial velocity is the fastest to converge.

	$\mathbf{v}_0$	$\mathbf{v}_{true}$	$\mathbf{v}_{true1}$	$\mathbf{v}_{true2}$	$\mathbf{v}_{rand}$
E_{rel}	$1.581 \cdot 10^{-4}$	$1.581 \cdot 10^{-4}$	$1.581 \cdot 10^{-4}$	$3.545 \cdot 10^{-1}$	$3.301 \cdot 10^{-1}$
G_{min}	$1.489 \cdot 10^{-3}$	$1.489 \cdot 10^{-3}$	$1.489 \cdot 10^{-3}$	$1.272 \cdot 10^{-3}$	$1.190 \cdot 10^{-3}$
G_{mid}	$1.488 \cdot 10^{-3}$	$1.488 \cdot 10^{-3}$	$1.488 \cdot 10^{-3}$	$1.148 \cdot 10^{-3}$	$1.128 \cdot 10^{-3}$
G_{max}	$1.489 \cdot 10^{-3}$	$1.489 \cdot 10^{-3}$	$1.489 \cdot 10^{-3}$	$1.075 \cdot 10^{-3}$	$1.349 \cdot 10^{-3}$
$f(v)$	-7.2503	-7.2503	-7.2503	-5.6539	-5.8405
$iter$	3127	20	3568	6298	155

Table 4.1: Experiment with synthetic data 1: values of the relative error, of the function G associated with the reference times t_{min}, t_{mid}, t_{max}, of the objective function f and of the number of iterations with kernel size equal to 13 and standard deviation equal to 2 with different starting guesses.

The multi-reference loss function is not a convex function, for this reason, we cannot avoid the convergence of the algorithm to a local minimum instead of a global one. This represents an intrinsic problem of the model. However, we expect that changing the parameters of the Gaussian kernel in the implementation of the image of warped events holds the potential for reducing the local minima issue as well as for improving the accuracy of the estimation, hence in the following section, we experimentally analyze how the results change when the Gaussian kernel is varied.

Kernel variation

In the previous example, we used a kernel of dimension 13×13 and standard deviation $\sigma = 2$. With this setting, we obtain the best results in

terms of accuracy.
We now try to change the kernel size and standard deviation values to see how much they affect the results. In particular, to avoid boundary effects, in the following examples, if necessarily the event camera domain is enlarged such that the kernel support is entirely contained in the image even when it is centered on the spatially most extreme events.

In Figure 4.7 we reduce the dimension of the kernel to 7×7 and the standard deviation to 1. The results are not as good as before: except for the case with the starting point equal to the ground truth, the relative error is higher than before, in particular for the starting guess equal to zero. Indeed, as we can see from Table 4.2, $f(v_{true}) = -13.722$ is much smaller than in the other cases.

	$\mathbf{v}_0$	$\mathbf{v}_{true}$	$\mathbf{v}_{true1}$	$\mathbf{v}_{true2}$	$\mathbf{v}_{rand}$
E_{rel}	$8.337 \cdot 10^{-1}$	$1.883 \cdot 10^{-4}$	$1.303 \cdot 10^{0}$	$1.312 \cdot 10^{-1}$	$4.085 \cdot 10^{-1}$
G_{min}	$1.893 \cdot 10^{-2}$	$3.142 \cdot 10^{-2}$	$1.016 \cdot 10^{-2}$	$2.732 \cdot 10^{-2}$	$1.521 \cdot 10^{-2}$
G_{mid}	$1.898 \cdot 10^{-2}$	$3.146 \cdot 10^{-2}$	$1.050 \cdot 10^{-2}$	$2.454 \cdot 10^{-2}$	$1.644 \cdot 10^{-2}$
G_{max}	$2.050 \cdot 10^{-2}$	$3.142 \cdot 10^{-2}$	$1.550 \cdot 10^{-2}$	$2.703 \cdot 10^{-2}$	$1.906 \cdot 10^{-2}$
$f(v)$	-8.446	-13.722	-5.094	-11.287	-7.329
iter	37158	30	1058	19468	1732

Table 4.2: Experiment with synthetic data 1: values of the relative error, of the function G associated with the reference times t_{min}, t_{mid}, t_{max}, of the objective function f and of the number of iterations with kernel size equal to 7 and standard deviation equal to 1, with different starting guesses.

We now try to increase the kernel's dimension to 19×19 and of σ to 3. In Figure 4.8 and in Table 4.3 we can see that the estimated velocity is better than the previous experiment for the initial velocity value equal to 0 and to the true value of $y = x$. For the starting guess equal to the true value on $y = 2x$ we obtain the same result. In the other two plots, the estimation is worse. In addition, we can observe that as an average all the plots take a smaller time to converge than the previous two experiments. From these

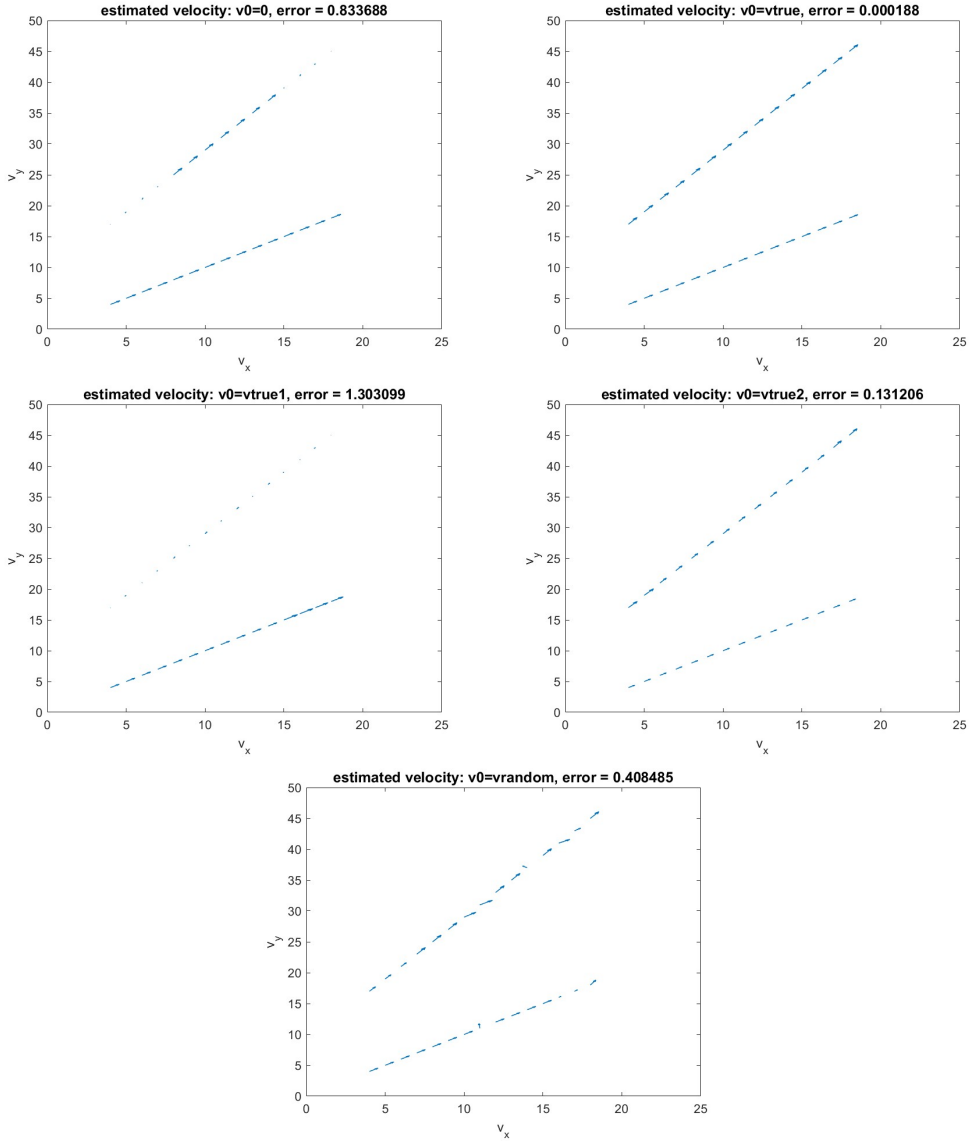

Figure 4.7: Plots of estimated velocity with kernel reduction of 7×7 and $\sigma = 1$ and camera domain of size 50×25, with different choices of starting guesses and associated relative error reported in Table 4.2.

experiments, we can presume that increasing too much the influence of the Gaussian distribution isn't necessarily equivalent to an improvement in the accuracy of the solution.

	$\mathbf{v}_0$	$\mathbf{v}_{true}$	$\mathbf{v}_{true1}$	$\mathbf{v}_{true2}$	$\mathbf{v}_{rand}$
E_{rel}	$6.028 \cdot 10^{-3}$	$6.028 \cdot 10^{-3}$	$6.028 \cdot 10^{-3}$	$4.560 \cdot 10^{-1}$	$4.243 \cdot 10^{-1}$
G_{min}	$2.171 \cdot 10^{-4}$	$2.171 \cdot 10^{-4}$	$2.171 \cdot 10^{-4}$	$3.123 \cdot 10^{-4}$	$1.886 \cdot 10^{-4}$
G_{mid}	$2.202 \cdot 10^{-4}$	$2.202 \cdot 10^{-4}$	$2.202 \cdot 10^{-4}$	$1.578 \cdot 10^{-4}$	$1.819 \cdot 10^{-4}$
G_{max}	$2.203 \cdot 10^{-4}$	$2.203 \cdot 10^{-4}$	$2.203 \cdot 10^{-4}$	$1.443 \cdot 10^{-4}$	$2.038 \cdot 10^{-4}$
$f(v)$	-5.327	-5.327	-5.327	-4.686	-4.589
$iter$	732	42	1102	1120	192

Table 4.3: Experiment with synthetic data 1: values of the relative error, of the function G associated with the reference times t_{min}, t_{mid}, t_{max}, of the objective function f and of the number of iterations with kernel size equal to 19 and standard deviation equal to 3 for different starting guesses.

Boundary issues

We now analyze how boundary issues can affect (negatively) the estimated velocities. In particular, starting from the best-performing kernel, revealed by previous experiments (13×13 with $\sigma = 2$), we shrink the event camera domain by 3 and then 6 pixels, both horizontally and vertically. In this way, we have two new experiments where the support of the kernel when centered at the position of spatially extreme events falls outside the image domain of 3 and 6 pixels, respectively.

Concerning the shrinkage-3, the results are shown in Figure 4.9 and Table 4.4. For all the starting guesses, we obtain an estimation that is worse than the first case. However, in the first 3 plots, we achieve sufficiently good results with a relative error of $3.2 \cdot 10^{-3}$.

For the shrinkage-6 case, the results are worse than the previous example for all the initial velocities, except for the third plot where the relative error

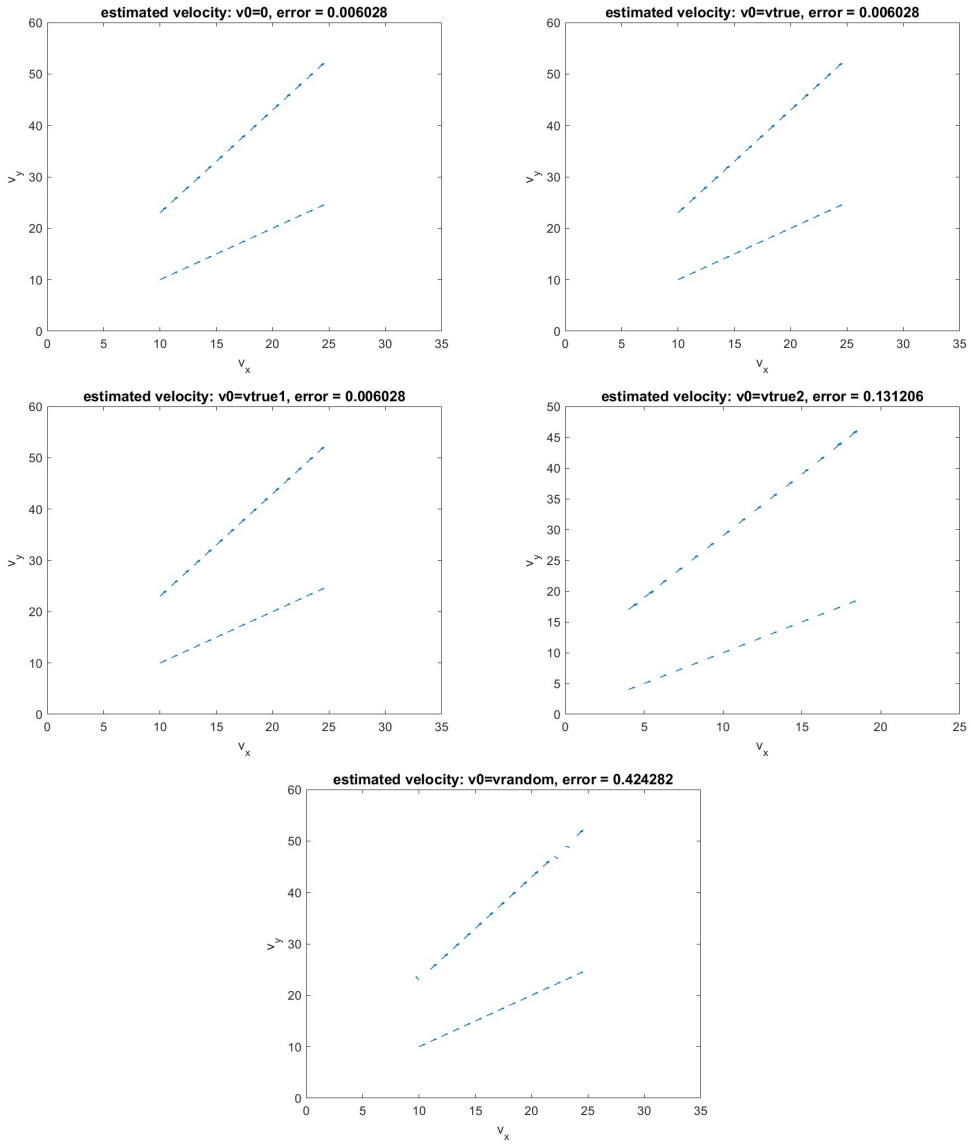

Figure 4.8: Plots of estimated velocity with kernel increment of 19×19 and $\sigma = 3$ and camera domain of size 60×35, with different choices of starting guesses and associated relative error reported in Table 4.3.

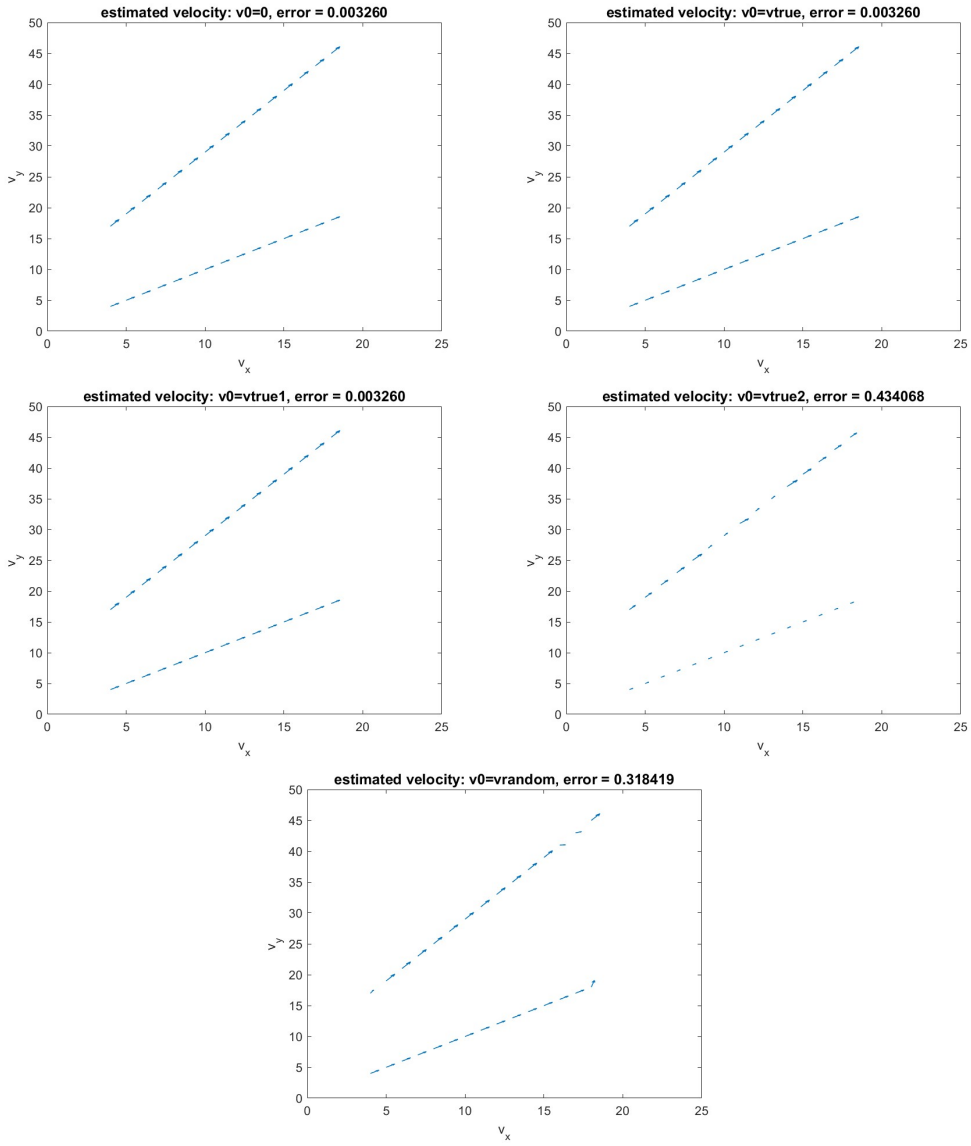

Figure 4.9: Plots of estimated velocity with shrinkage-3 pixels and camera domain of size 50×25, with different choices of starting guesses and associated relative error reported in Table 4.4.

	$\mathbf{v}_0$	$\mathbf{v}_{true}$	$\mathbf{v}_{true1}$	$\mathbf{v}_{true2}$	$\mathbf{v}_{rand}$
E_{rel}	$3.260 \cdot 10^{-3}$	$3.260 \cdot 10^{-3}$	$3.260 \cdot 10^{-3}$	$4.341 \cdot 10^{-1}$	$3.184 \cdot 10^{-1}$
G_{min}	$2.137 \cdot 10^{-3}$	$2.137 \cdot 10^{-3}$	$2.137 \cdot 10^{-3}$	$1.272 \cdot 10^{-3}$	$1.717 \cdot 10^{-3}$
G_{mid}	$2.153 \cdot 10^{-3}$	$2.153 \cdot 10^{-3}$	$2.153 \cdot 10^{-3}$	$1.421 \cdot 10^{-3}$	$1.640 \cdot 10^{-3}$
G_{max}	$2.142 \cdot 10^{-3}$	$2.142 \cdot 10^{-3}$	$2.142 \cdot 10^{-3}$	$1.278 \cdot 10^{-3}$	$1.942 \cdot 10^{-3}$
$f(v)$	-7.314	-7.314	-7.314	-4.595	-5.912
$iter$	2702	26	3298	946	354

Table 4.4: Experiment with synthetic data 1: values of the relative error, of the function G associated with the reference times t_{min}, t_{mid}, t_{max}, of the objective function f and of the number of iterations with kernel size equal to 13 and standard deviation equal to 3 with shrinkage-3.

is a bit lower (Figure 4.10 and Table 4.5). Yet, they all are worse than the first experiment.

	$\mathbf{v}_0$	$\mathbf{v}_{true}$	$\mathbf{v}_{true1}$	$\mathbf{v}_{true2}$	$\mathbf{v}_{rand}$
E_{rel}	$4.895 \cdot 10^{-2}$	$4.895 \cdot 10^{-2}$	$1.815 \cdot 10^{-1}$	$2.281 \cdot 10^{-1}$	$4.019 \cdot 10^{-1}$
G_{min}	$2.395 \cdot 10^{-3}$	$2.395 \cdot 10^{-3}$	$2.252 \cdot 10^{-3}$	$1.212 \cdot 10^{-3}$	$1.781 \cdot 10^{-3}$
G_{mid}	$3.400 \cdot 10^{-3}$	$3.400 \cdot 10^{-3}$	$3.404 \cdot 10^{-3}$	$3.404 \cdot 10^{-3}$	$2.582 \cdot 10^{-3}$
G_{max}	$2.629 \cdot 10^{-3}$	$2.629 \cdot 10^{-3}$	$2.448 \cdot 10^{-3}$	$2.458 \cdot 10^{-3}$	$2.294 \cdot 10^{-3}$
$f(v)$	-7.833	-7.833	-7.623	-7.603	-6.120
$iter$	1591	53	2521	3060	470

Table 4.5: Experiment with synthetic data 1: values of the relative error, of the function G associated with the reference times t_{min}, t_{mid}, t_{max}, and of the objective function f and of the number of iterations with kernel size equal to 13 and standard deviation equal to 3 with shrinkage-6, by varying the starting guesses.

Finally, we carry out another experiment where, instead of shrinking we enlarge the camera domain. We expect that, since boundary issues will not be present as in the first experiment, the obtained result will be very similar. The results shown in Figure 4.11 confirm that for the first 3 and the last

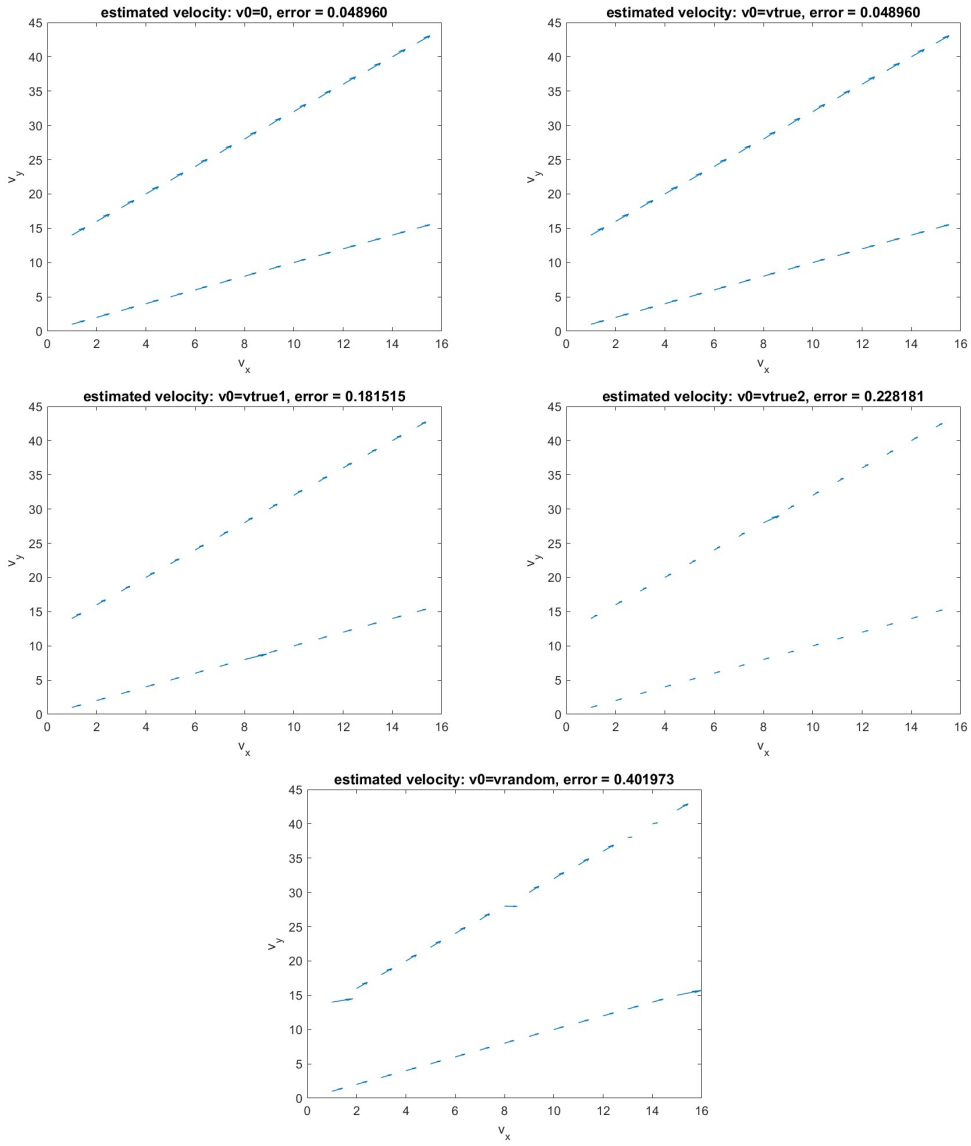

Figure 4.10: Plots of estimated velocity with shrinkage-6 pixels and camera domain of size 45×16, with different choices of starting guesses and associated relative error reported in Table 4.5.

starting guess cases, for which we obtain the same results. Only for the fourth starting guess, the result is very slightly better. Also, the number of iterations taken to converge is similar to the first case.

	$\mathbf{v}_0$	$\mathbf{v}_{true}$	$\mathbf{v}_{true1}$	$\mathbf{v}_{true2}$	$\mathbf{v}_{rand}$
E_{rel}	$1.581 \cdot 10^{-4}$	$1.581 \cdot 10^{-4}$	$1.581 \cdot 10^{-4}$	$3.311 \cdot 10^{-1}$	$3.297 \cdot 10^{-1}$
G_{min}	$1.096 \cdot 10^{-3}$	$1.096 \cdot 10^{-3}$	$1.096 \cdot 10^{-3}$	$9.459 \cdot 10^{-4}$	$8.781 \cdot 10^{-4}$
G_{mid}	$1.096 \cdot 10^{-3}$	$1.096 \cdot 10^{-3}$	$1.096 \cdot 10^{-3}$	$8.394 \cdot 10^{-4}$	$8.301 \cdot 10^{-4}$
G_{max}	$1.096 \cdot 10^{-3}$	$1.096 \cdot 10^{-3}$	$1.096 \cdot 10^{-3}$	$7.979 \cdot 10^{-4}$	$9.946 \cdot 10^{-4}$
$f(v)$	-7.250	-7.250	-7.250	-5.660	-5.842
$iter$	3190	22	2831	1146	186

Table 4.6: Experiment with synthetic data 1: values of the relative error, of the function G associated with the reference times t_{min}, t_{mid}, t_{max}, of the objective function f and of the number of iterations with kernel size equal to 13 and standard deviation equal to 3 with an enlarged domain.

4.2.1 A variation of Experiment 1

We think it is relevant to show another example that is a slight variation of the previous one: we consider the same two points moving on the two straight lines that are disposed such that they intersect. Figure 4.12 shows the ground truth velocities.

This example is interesting because in the intersection point of the two lines we have two events generated by different velocities, therefore the hypothesis of the model [1] is not satisfied anymore. Even for us, it is not simple to predict for sure the velocities this pixel should have. In the cases where we give as a starting guess the correct velocities, we set this value equal to:

$$\mathbf{v}_{int} = \frac{\mathbf{v}^{(1)} + \mathbf{v}^{(2)}}{2} = \left(\frac{v_x^{(1)} + v_x^{(1)}}{2}, \frac{v_x^{(2)} + v_x^{(2)}}{2} \right) = (1, 1.5)$$

where $\mathbf{v}^{(1)} = (v_x^{(1)}, v_y^{(1)})$ and $\mathbf{v}^{(2)} = (v_x^{(1)}, v_y^{(1)})$ are the velocities of the two points. This seems to be the most reasonable estimation.

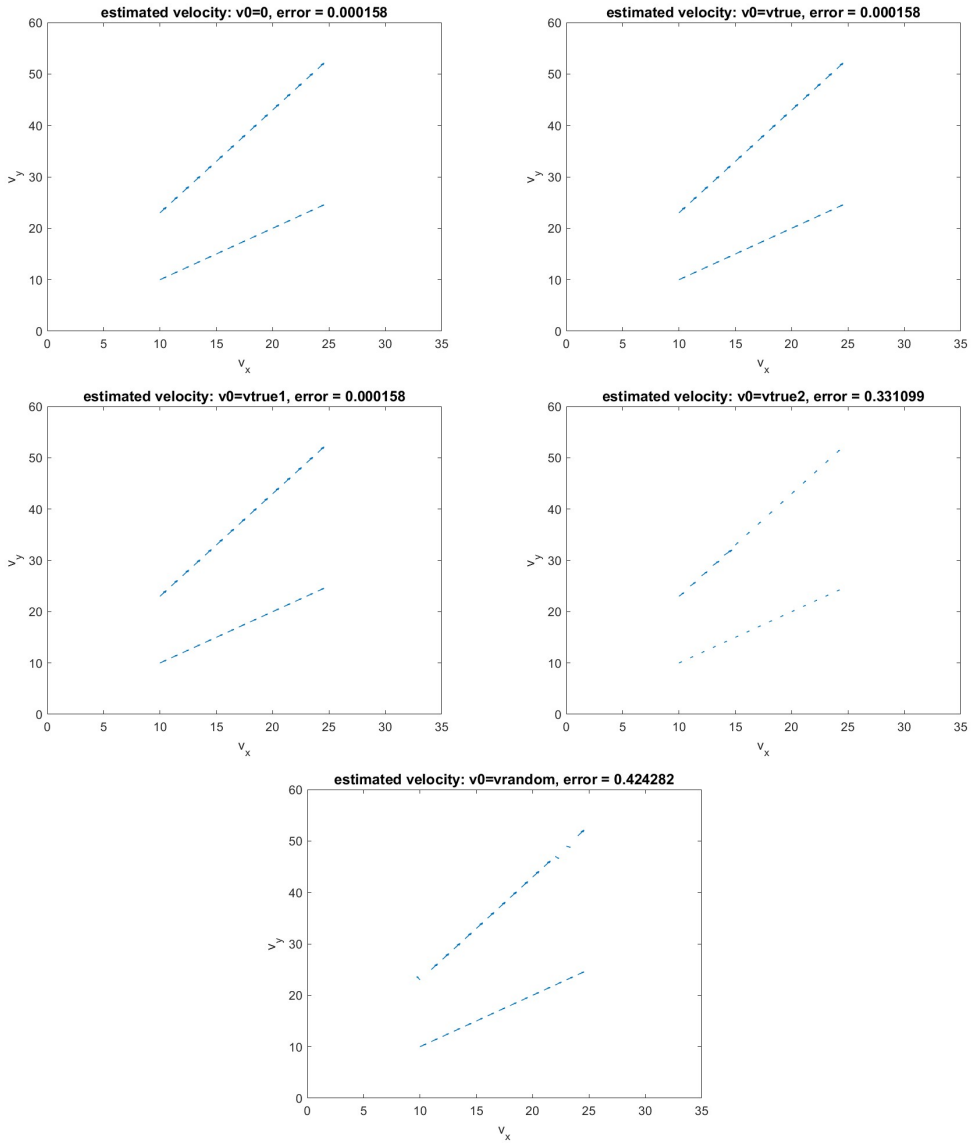

Figure 4.11: Plots of estimated velocity with enlarged camera domain of 3 pixels of size 60×35 with different choices of starting guesses and associated relative error reported in Table 4.6.

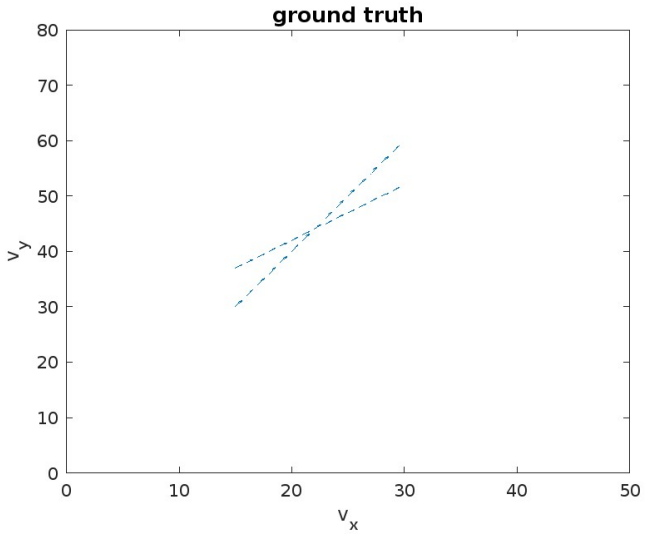

Figure 4.12: Plot of the ground truth velocities of two ideals points 1 and 2 moving with ideal velocities $\mathbf{v}^{(1)} = (v_x, v_y) = (1,1)pixel/s$ and $\mathbf{v}^{(2)} = (v_x, v_y) = (1,2)pixel/s$ along the straight intersecting lines $y = x + 8$ and $y = 2x$, plotted on the domain of the event camera of size 60×30.

We select the dimension of the kernel equal to 19 and the standard deviation σ equal to 3.
In Figure 4.13 are presented the different plots, varying the starting guesses as before. First of all, we observe that in the first 3 cases, the algorithm reaches the same minimum point of f (Table 4.7). The results are not as good as before however in the intersection pixel the algorithm estimates a velocity of $\mathbf{v} = (v_x, v_y) = (0.999177, 1.50028)$ for the the first 3 plots. It is significant that even if it starts with null velocity values, the method estimates with high accuracy the velocities we predicted.

4.3 Experiment 2 on synthetic data

We now apply the same analysis to a different example: consider 3 points moving in parallel along the line $y = x$ with velocities $v^{(1)} = (v_x, v_y) = (1,1)pixel/s$. After $5s$, they half their velocity to $v^{(2)} = (v_x, v_y) = \left(\frac{1}{2}, \frac{1}{2}\right) pixel/s$.

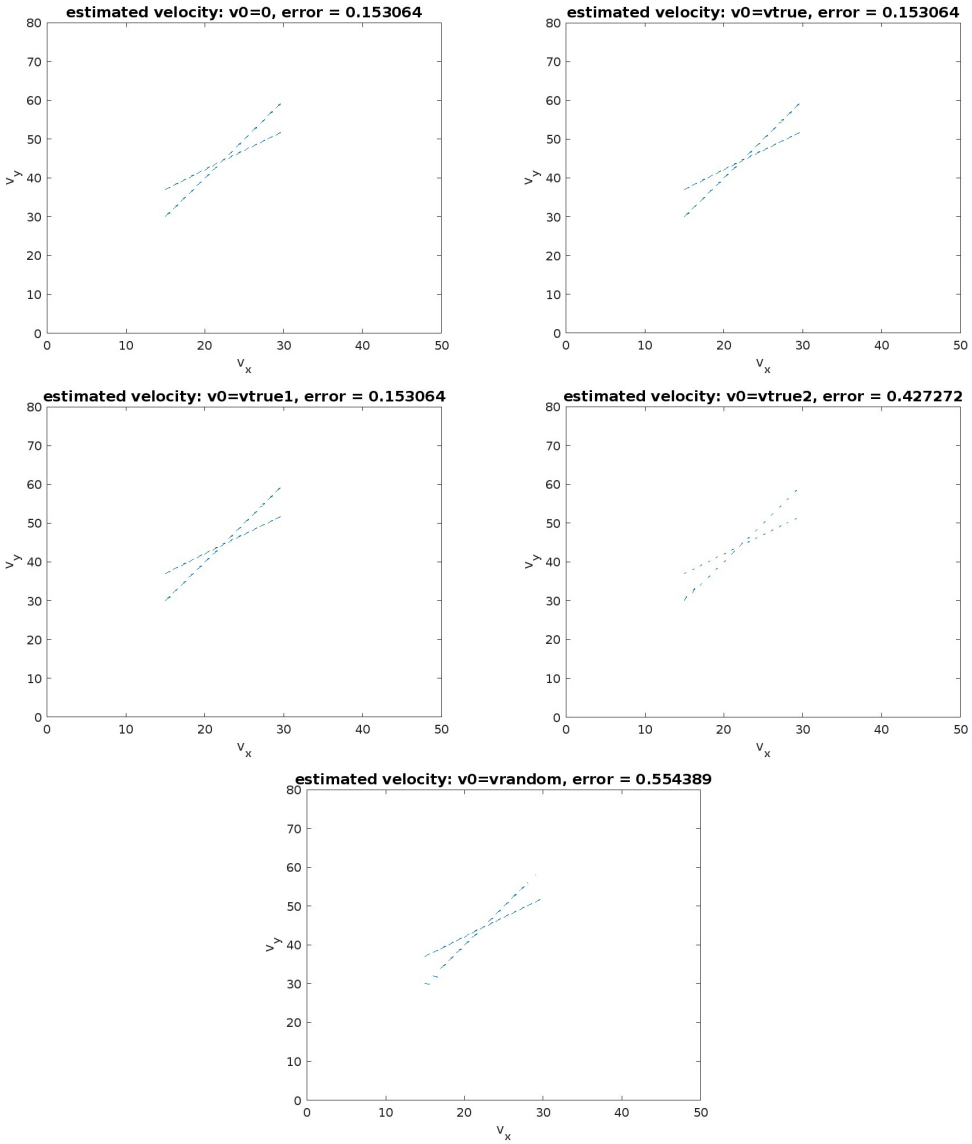

Figure 4.13: Plots of estimated velocity with enlarged camera domain of 14 pixels of size 80×50 with different choices of starting guesses and associated relative error reported in Table 4.7.

	$\mathbf{v}_0$	$\mathbf{v}_{true}$	$\mathbf{v}_{true1}$	$\mathbf{v}_{true2}$	$\mathbf{v}_{rand}$
E_{rel}	$1.530 \cdot 10^{-1}$	$1.530 \cdot 10^{-1}$	$1.530 \cdot 10^{-1}$	$4.272 \cdot 10^{-1}$	$5.543 \cdot 10^{-1}$
G_{min}	$2.161 \cdot 10^{-4}$	$2.161 \cdot 10^{-4}$	$2.161 \cdot 10^{-4}$	$1.800 \cdot 10^{-4}$	$1.787 \cdot 10^{-4}$
G_{mid}	$2.642 \cdot 10^{-4}$	$2.642 \cdot 10^{-4}$	$2.642 \cdot 10^{-4}$	$2.304 \cdot 10^{-4}$	$2.00 \cdot 10^{-4}$
G_{max}	$2.161 \cdot 10^{-4}$	$2.161 \cdot 10^{-4}$	$2.161 \cdot 10^{-4}$	$1.461 \cdot 10^{-4}$	$1.791 \cdot 10^{-4}$
$f(v)$	-5.178	-5.178	-5.178	-4.241	-4.090
$iter$	657	52	1241	1957	243

Table 4.7: Experiment with synthetic data 1-variation: values of the relative error, of the function G associated with the reference times t_{min}, t_{mid}, t_{max}, of the objective function f and of the number of iterations with kernel size equal to 19 and standard deviation equal to 3 for different starting guesses.

These two points generate a strip of 15 events arranged in 3 lines. The ground truth of the velocity is presented in Figure 4.14.
As before, we apply the Steepest Descent method with inexact line search to the multi-focus objective function.
We also consider the same starting guesses as before except for the third and the fourth case where we take the velocities of the first part of the strip $\mathbf{v}^{(1)}$ and of the second part $\mathbf{v}^{(2)}$, respectively.

Figure 4.15 shows the results. In this case, we have selected a kernel support with dimension 7×7 and enlarged the camera domain of 12 more pixels such that Gaussian support is entirely contained in the image. The best result is obtained with a starting guess equal to the correct value. The other cases, except for the random starting guess, have similar relative errors. We also observe in Tab 4.8 that the lower value of the multi-focus function is -12.15 for the starting guess equal to the correct velocity of the second part of the strip. However, this is not the best result in terms of accuracy. Moreover, the first 3 plots present a similar value of f but they have a significantly different relative error.
In this experiment, the image $H(i, j)$ of events warped at the reference times t_{min}, t_{mid}, t_{max} are not centered on the same peak, this happens because we

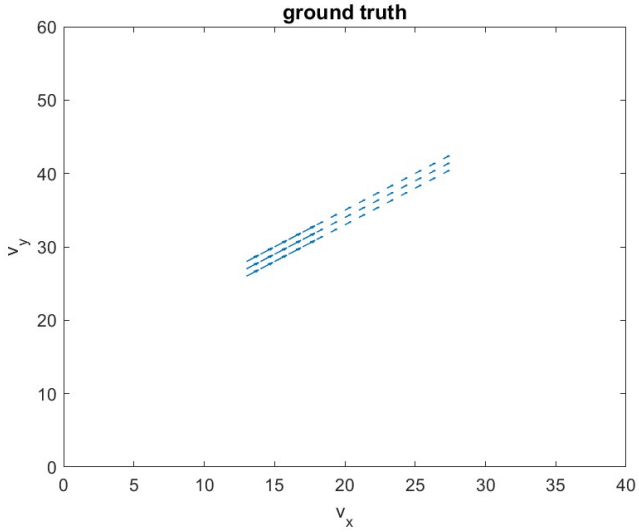

Figure 4.14: Plot of the ground truth velocities of three ideal points moving along the straight lines $y = x + k$ for $k = 0, 1, 2$. They move at first with velocities $v^{(1)} = (v_x, v_y) = (1, 1) pixel/s$, then after $5s$ they half their velocities to $v^{(2)} = (v_x, v_y) = \left(\frac{1}{2}, \frac{1}{2}\right) pixel/s$. The size domain is 60×40.

have two different velocities starting from different positions. In Figure 4.16 the two peaks of the ground truth velocity almost overlap, in 4.17 they are disjointed and 4.18 they increase their distance. For this reason, we need to enlarge the camera domain of more pixels than in the previous example.

We try now to increase the dimension of the kernel support to 31×31 and of the standard deviation to 5. Figure 4.19 shows that the results are worse than before for all the plots, Tab 4.9 confirms that the first 4 plots reach the same minimum of f. From Figure 4.20, 4.21, 4.22 we can see that considering a larger kernel dimension, move closer the two peaks due to the two velocities. Probably this is the reason why in the previous example we have that the function reaches the minimum for velocities that don't coincide with the best estimation, while here this problem doesn't hold. The presence of two disjoint peaks increases the value of $f(\mathbf{v})$.

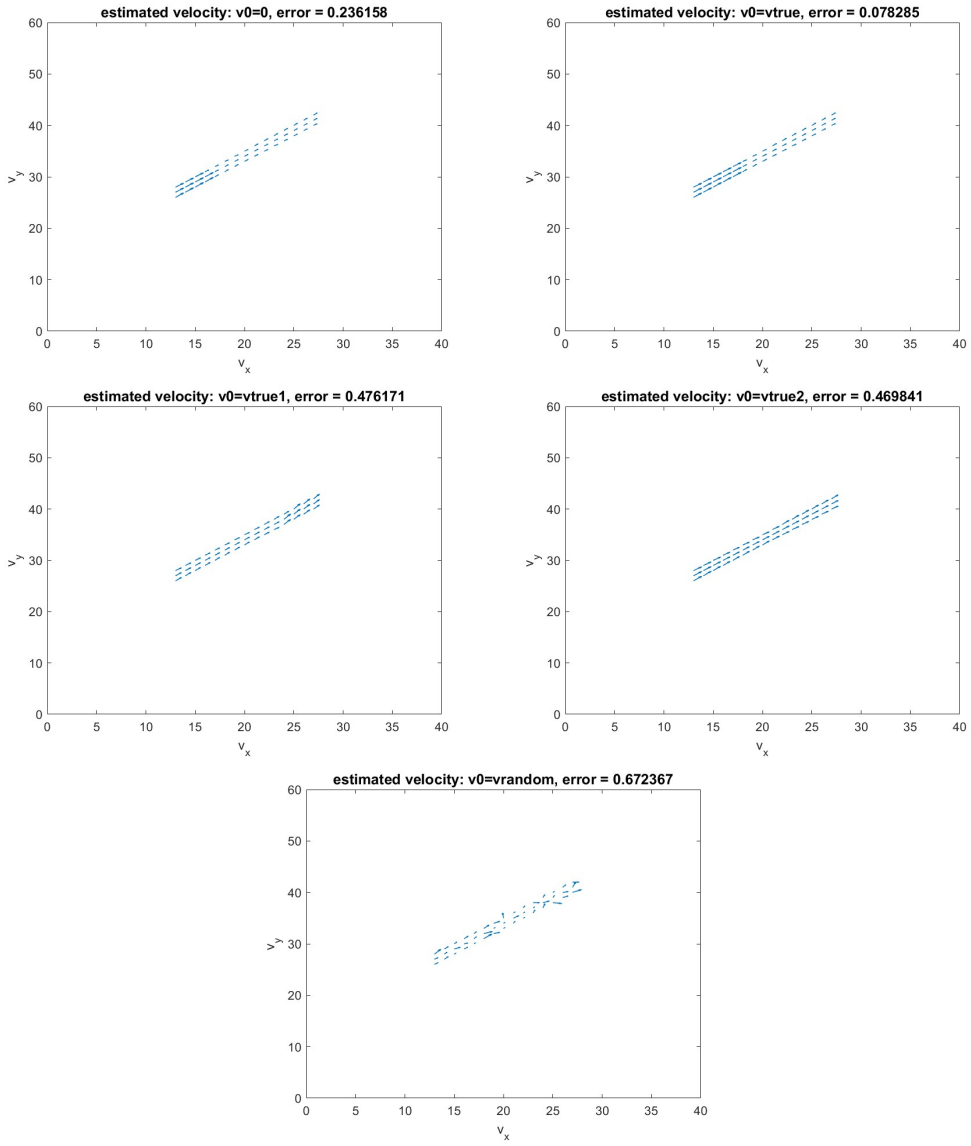

Figure 4.15: Plots of estimated velocity with respect to different choices of starting guesses and associated relative error reported in Table 4.8. The camera domain has a size of 60×40.

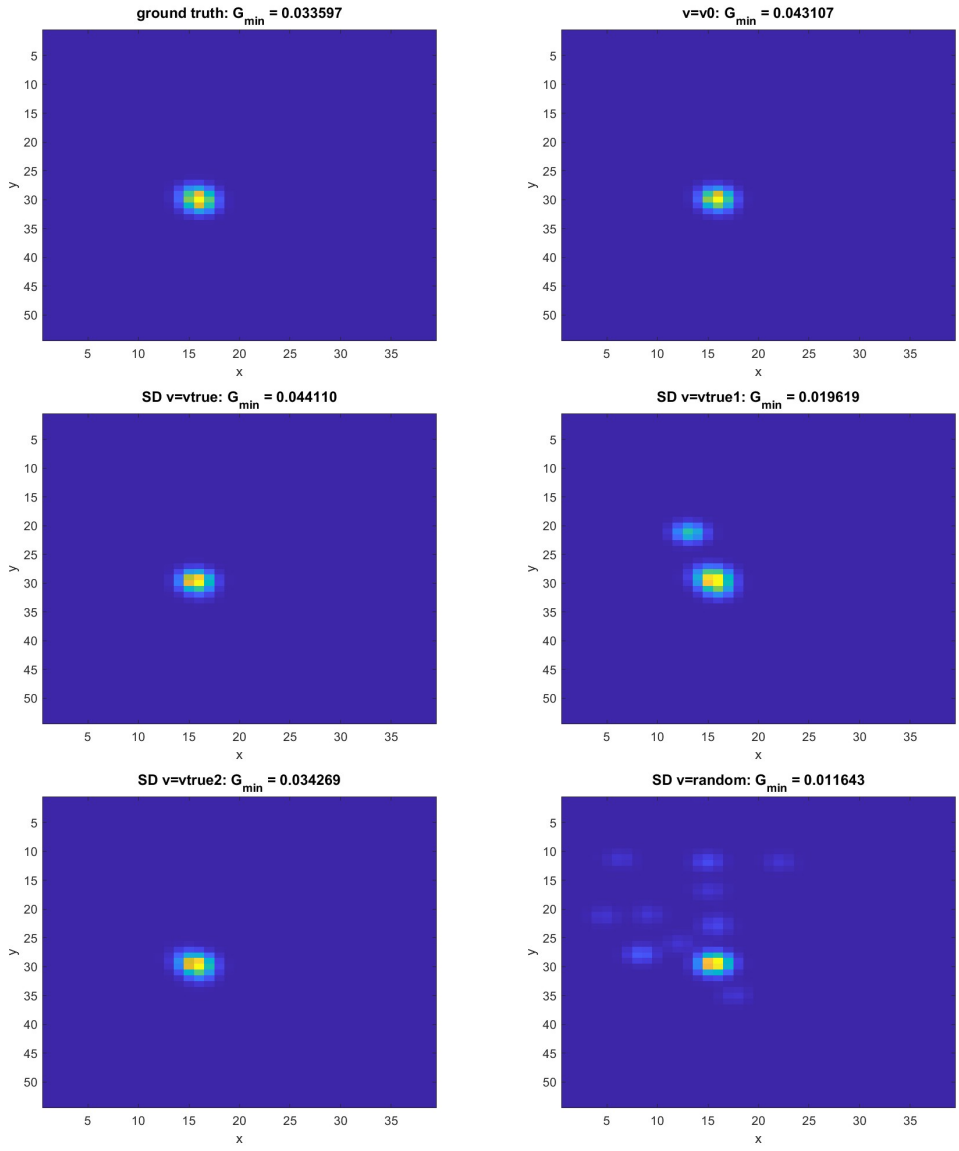

Figure 4.16: Representation of the image $H(i, j)$ of the events warped to the reference time t_{min} by varying the starting guesses and the associated value of the function G reported in Table 4.8, compared to the ground truth.

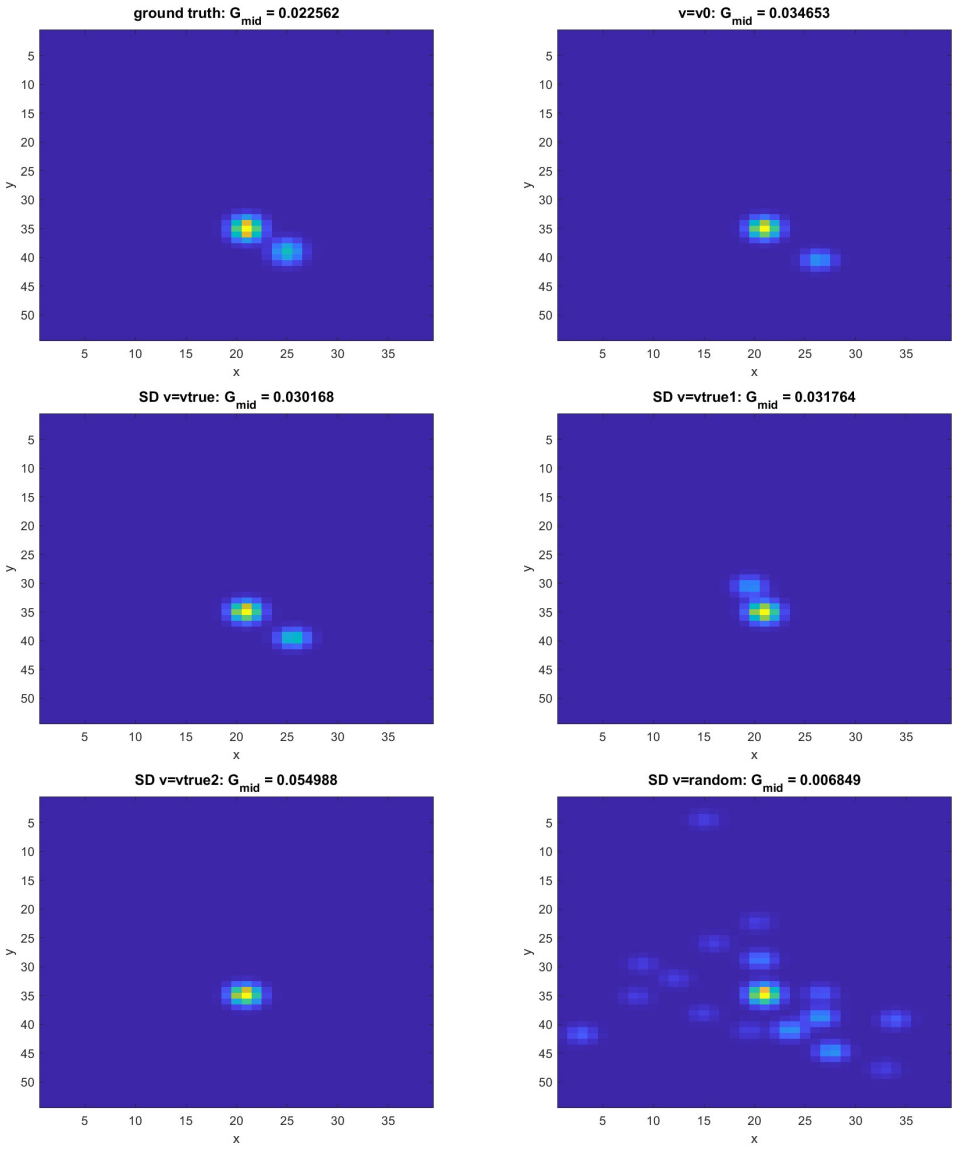

Figure 4.17: Representation of the image $H(i, j)$ of the events warped to the reference time t_{mid} by varying the starting guesses and the associated value of the function G reported in Table 4.8, compared to the ground truth.

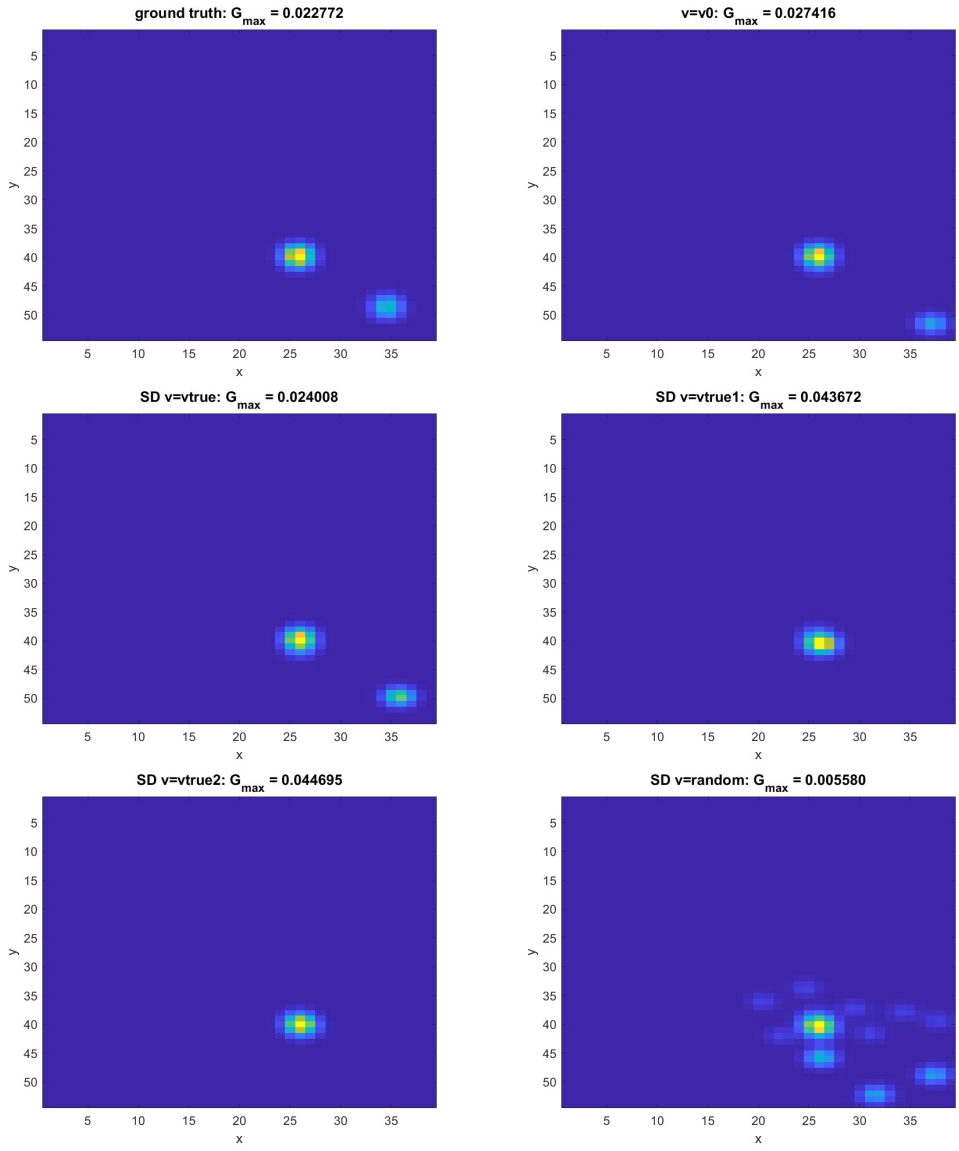

Figure 4.18: Representation of the image $H(i, j)$ of the events warped to the reference time t_{max} by varying the starting guesses and the associated value of the function G reported in Table 4.8, compared to the ground truth.

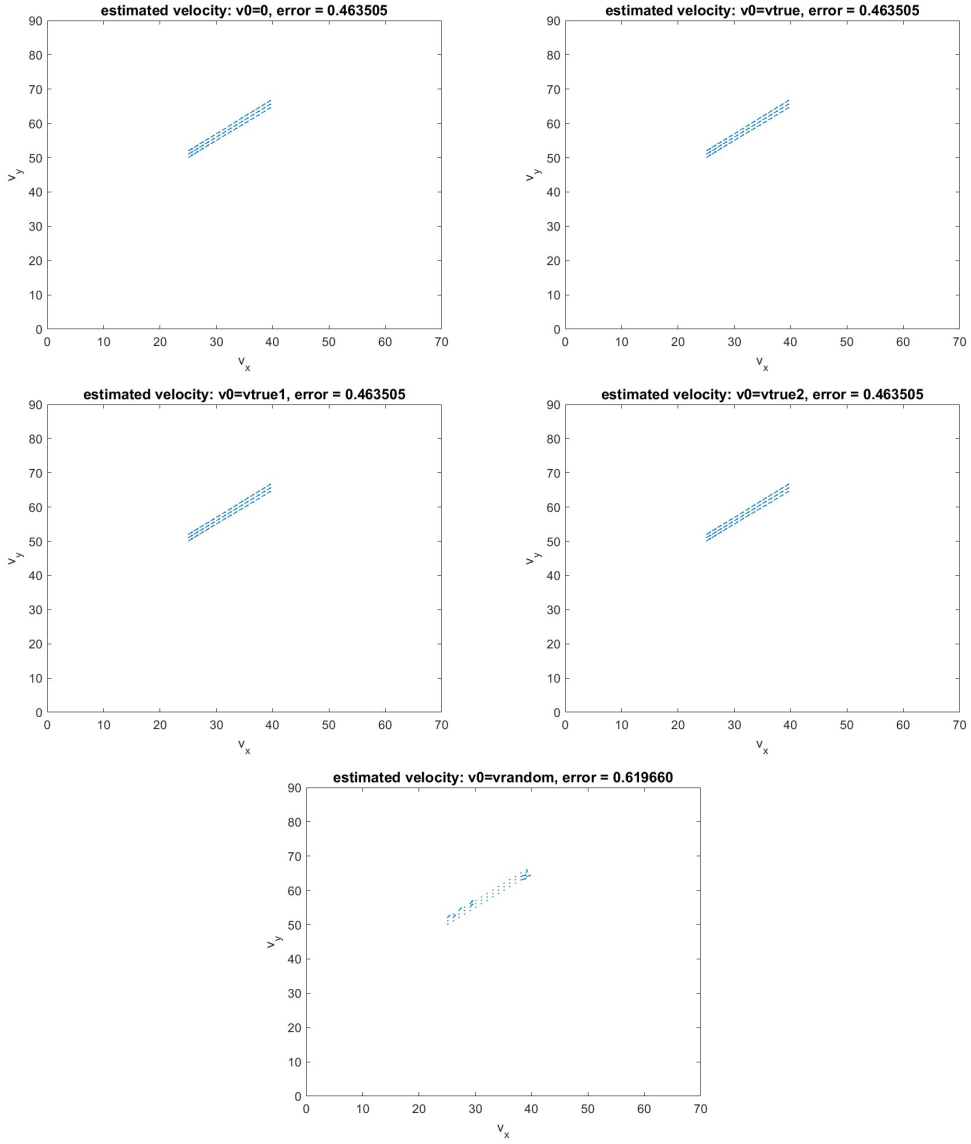

Figure 4.19: Plots of estimated velocity with respect to different choices of starting guesses and associated relative error reported in Table 4.9. The camera domain has a size of 90×70.

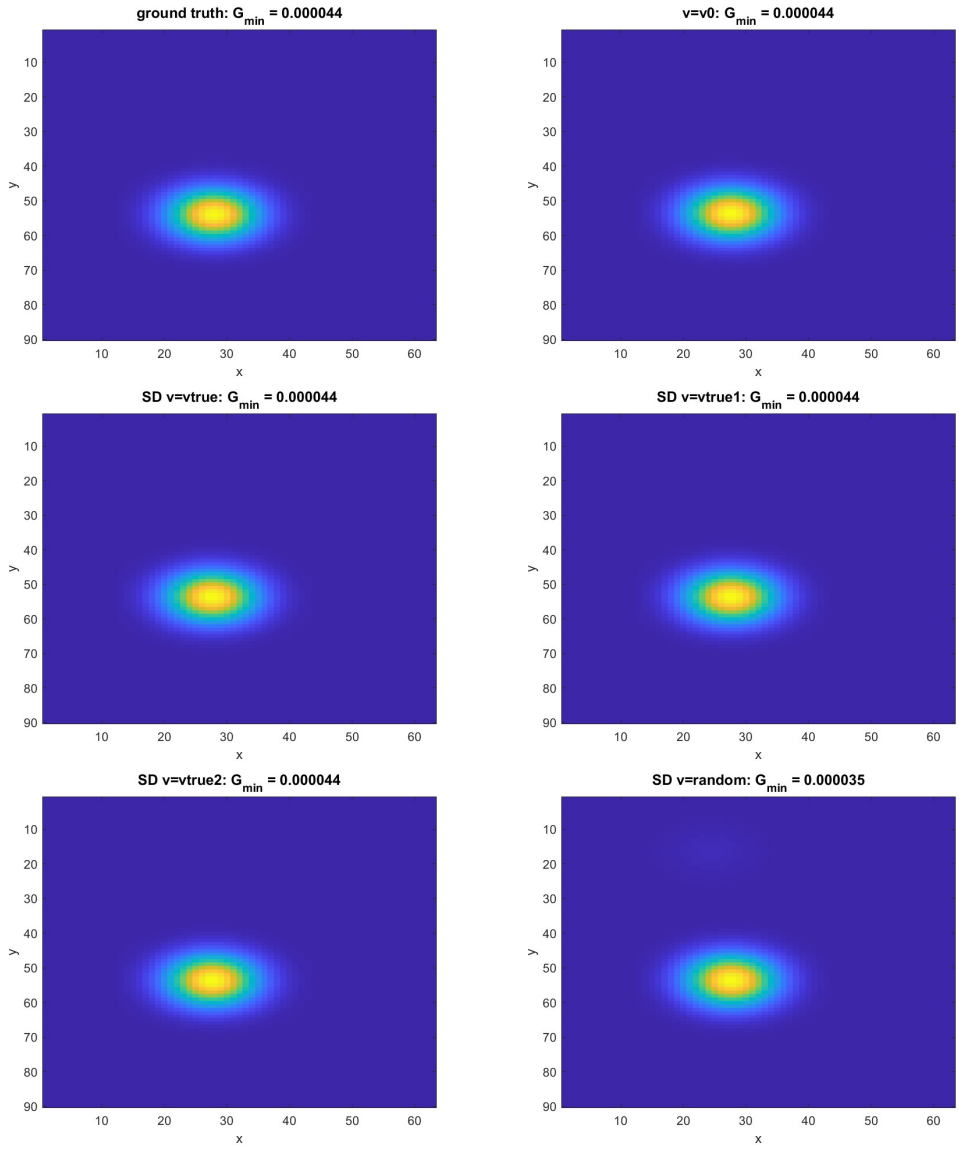

Figure 4.20: Representation of the image $H(i, j)$ of the events warped to the reference time t_{min} by varying the starting guesses and the associated value of the function G reported in Table 4.9, compared to the ground truth.

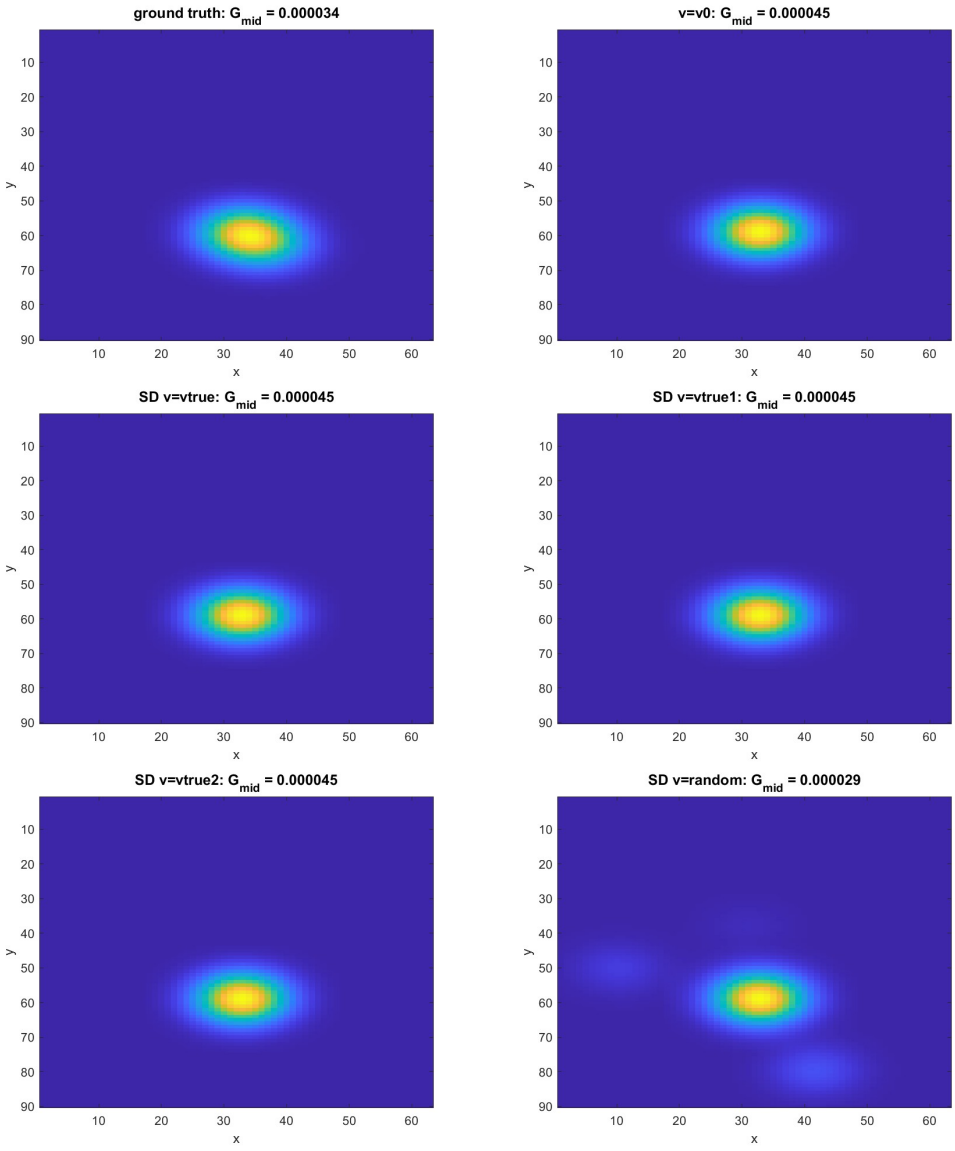

Figure 4.21: Representation of the image $H(i, j)$ of the events warped to the reference time t_{mid} by varying the starting guesses and the associated value of the function G reported in Table 4.9, compared to the ground truth.

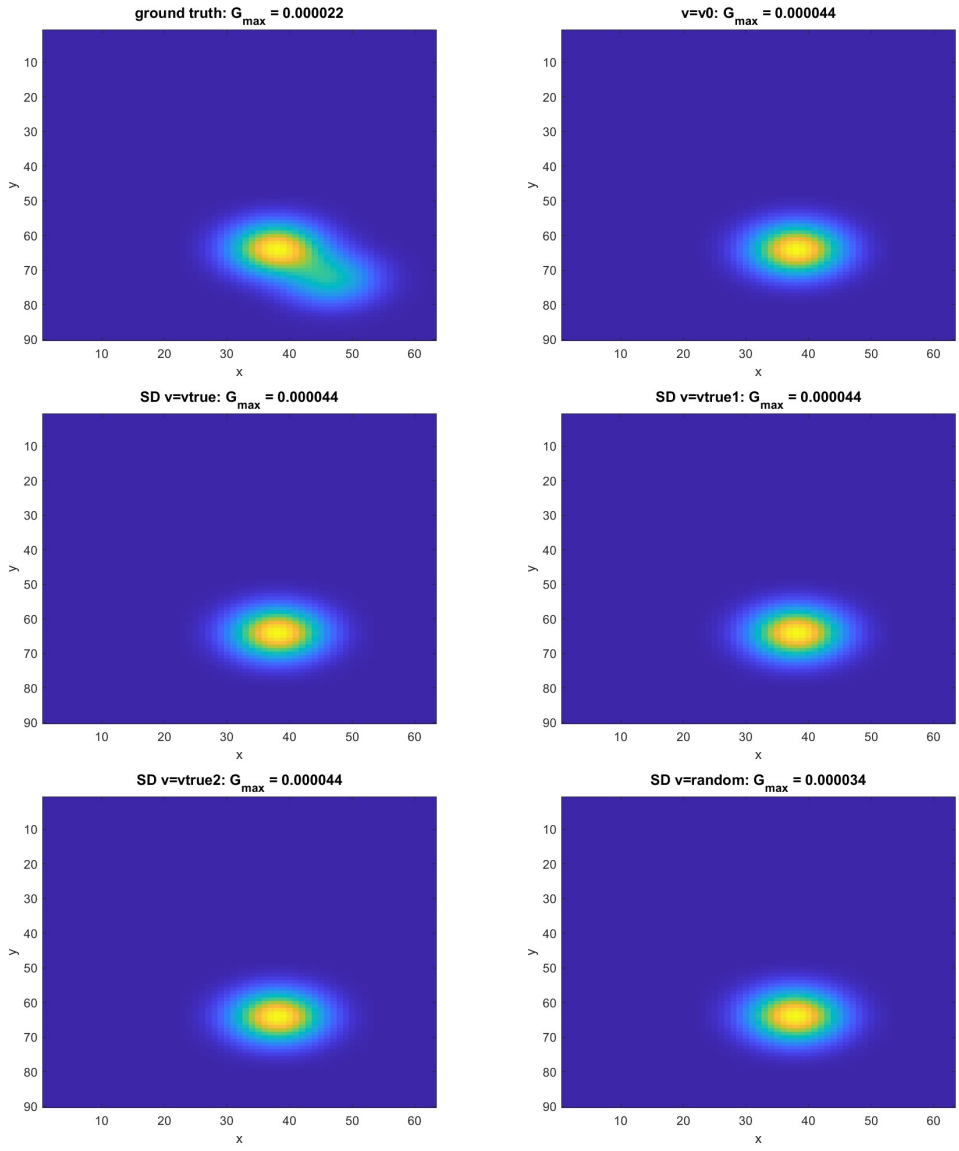

Figure 4.22: Representation of the image $H(i, j)$ of the events warped to the reference time t_{max} by varying the starting guesses and the associated value of the function G reported in Table 4.9, compared to the ground truth.

	$\mathbf{v}_0$	$\mathbf{v}_{true}$	$\mathbf{v}_{true1}$	$\mathbf{v}_{true2}$	$\mathbf{v}_{rand}$
E_{rel}	$2.361 \cdot 10^{-1}$	$7.828 \cdot 10^{-2}$	$4.761 \cdot 10^{-1}$	$4.698 \cdot 10^{-1}$	$6.723 \cdot 10^{-1}$
G_{min}	$4.310 \cdot 10^{-2}$	$4.410 \cdot 10^{-2}$	$1.961 \cdot 10^{-2}$	$3.426 \cdot 10^{-2}$	$1.164 \cdot 10^{-2}$
G_{mid}	$3.465 \cdot 10^{-2}$	$3.016 \cdot 10^{-2}$	$3.176 \cdot 10^{-2}$	$5.498 \cdot 10^{-2}$	$6.849 \cdot 10^{-3}$
G_{max}	$2.741 \cdot 10^{-2}$	$2.401 \cdot 10^{-2}$	$4.367 \cdot 10^{-2}$	$4.469 \cdot 10^{-2}$	$5.580 \cdot 10^{-3}$
$f(v)$	-8.991	-8.260	-8.155	-12.150	-1.988
$iter$	7052	490	1535	60	1627

Table 4.8: Experiment with synthetic data 2: Values of the relative error, of the function G associated with the reference times t_{min}, t_{mid}, t_{max}, of the objective function f and of the number of iterations with kernel size equal to 7 and standard deviation equal to 1 with an enlarged domain.

Regularization term

We now consider problem ($\mathcal{B}$) and we apply to the same set of synthetic data the Accelerated Proximal Gradient algorithm, adding as regularization the Vectorial Total Variation multiplied by the regularization parameter μ. To select the best value of μ we run the algorithm with different regularization parameters, for each of them we compute the relative error (4.1). We select the μ values that give the smaller relative error.

As we observe from Figure 4.23 the results are slightly worse than the same experiment without regularization term. Also in this case we can observe lower relative error associated with worse results. From Tab 4.10 we can observe that the best regulariztion parameter μ depends on the initial guess but in general it satisfied $\mu \in [1 \cdot 10^{-6}, 9 \cdot 10^{-6}]$. Also, the number of iterations depends on the initial guess.

4.4 Test on real data

We are now ready to test the algorithm on the real data. In this section, we will consider two different sets of events collected from the passage of

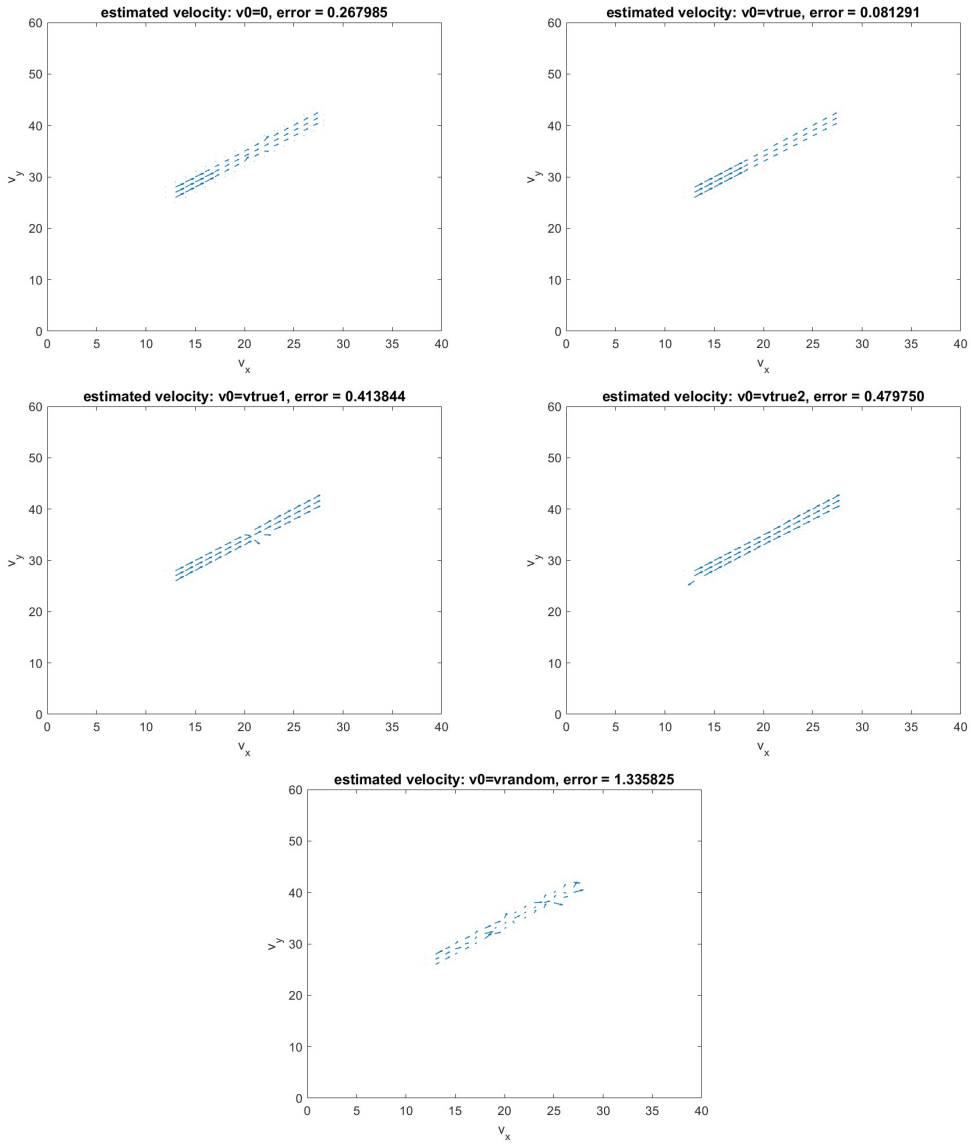

Figure 4.23: Plots of estimated velocity with respect to different choices of starting guesses and associated relative error reported in Table 4.10. The camera domain has a size of 90×70.

	$\mathbf{v}_0$	$\mathbf{v}_{true}$	$\mathbf{v}_{true1}$	$\mathbf{v}_{true2}$	$\mathbf{v}_{rand}$
E_{rel}	$4.635 \cdot 10^{-1}$	$4.635 \cdot 10^{-1}$	$4.761 \cdot 10^{-1}$	$4.635 \cdot 10^{-1}$	$6.196 \cdot 10^{-1}$
G_{min}	$4.368 \cdot 10^{-5}$	$4.368 \cdot 10^{-5}$	$4.368 \cdot 10^{-5}$	$4.368 \cdot 10^{-5}$	$3.506 \cdot 10^{-5}$
G_{mid}	$4.476 \cdot 10^{-5}$	$4.476 \cdot 10^{-5}$	$4.476 \cdot 10^{-5}$	$4.476 \cdot 10^{-5}$	$2.898 \cdot 10^{-5}$
G_{max}	$4.433 \cdot 10^{-5}$	$4.433 \cdot 10^{-5}$	$4.433 \cdot 10^{-5}$	$4.433 \cdot 10^{-5}$	$3.365 \cdot 10^{-5}$
$f(v)$	-2.389	-2.389	-2.389	-2.389	-1.704
$iter$	274	126	170	49	1744

Table 4.9: Experiment with synthetic data 2 with Steepest Descent algorithm: values of the relative error, of the function G associated with the reference times t_{min}, t_{mid}, t_{max}, of the objective function f and of the number of iterations with kernel size equal to 31 and standard deviation equal to 5 with an enlarged domain of 24 pixels.

the same parcel with heigh $15cm$, used in Section 2.4. In both situations, the parcel is moving on the conveyor with fixed velocity $\mathbf{v}_{conv} = (1.5, 0)m/s$. However, in the first dataset, which is the same used in the experiment with constant velocity in Section 2.4, the camera is well-calibrated and it is aligned to the conveyor, then the optical flow is mostly constant in every pixel of the parcel. In the second case, the acquisitions are made with a camera that is rotated with respect to the conveyor plane. This inclination causes an optical flow that changes in every pixel position and it is faster in the points nearer to the camera and slower in the ones farther away. Both of these datasets are provided by the company Datalogic S.p.A..

As a starting guess we consider the results given by the algorithm applied to the loss function where the velocity $\mathbf{v}$ is assumed to be constant (Section 2.4). That is

$$\mathbf{v}_0 = \mathbf{v}_{opt} = (1250.68164, 20.5165)pixel/s.$$

In both experiments, we are going to use the multi-reference focus objective function as in the previous examples.

We apply the Steepest Descent algorithm and the Accelerated Proximal Gradient with the Total Variation Regularizer. In both cases, we use a kernel

	$\mathbf{v}_0$	$\mathbf{v}_{true}$	$\mathbf{v}_{true1}$	$\mathbf{v}_{true2}$	$\mathbf{v}_{rand}$
μ	$4.524 \cdot 10^{-6}$	$1.730 \cdot 10^{-3}$	$2.103 \cdot 10^{-6}$	$9.310 \cdot 10^{-6}$	$5.344 \cdot 10^{-6}$
E_{rel}	$2.679 \cdot 10^{-1}$	$8.129 \cdot 10^{-2}$	$4.138 \cdot 10^{-1}$	$4.797 \cdot 10^{-1}$	$1.335 \cdot 10^{0}$
G_{min}	$4.310 \cdot 10^{-2}$	$4.411 \cdot 10^{-2}$	$1.961 \cdot 10^{-2}$	$3.426 \cdot 10^{-2}$	$1.164 \cdot 10^{-2}$
G_{mid}	$3.465 \cdot 10^{-2}$	$3.016 \cdot 10^{-2}$	$3.176 \cdot 10^{-2}$	$5.498 \cdot 10^{-2}$	$6.849 \cdot 10^{-3}$
G_{max}	$2.741 \cdot 10^{-2}$	$2.400 \cdot 10^{-2}$	$4.367 \cdot 10^{-2}$	$4.469 \cdot 10^{-2}$	$5.580 \cdot 10^{-3}$
$f(v)$	-8.568	-8.143	-11.814	-11.766	-2.008
$iter$	310	63	184	249	589

Table 4.10: Experiment with synthetic data 2 with Accelerated Proximal Gradient algorithm: values of the selected best regularization parameter μ, the relative error, of the function G associated with the reference times t_{min}, t_{mid}, t_{max}, of the objective function f and of the number of iterations with kernel size equal to 7 and standard deviation equal to 5 with an enlarged domain of 12 pixels.

support of dimension 7×7 and a standard deviation equal to 1. We use these values because, in the previous example, they give us the best results, moreover increasing them leads to a significant increment of time. Same speech for the enlargement of the camera domain, we add just 2 pixels. The tolerance for the stopping criteria is set to 10^{-9} for both algorithms. Regarding the regularization parameter for the Accelerated Proximal Gradient, we set 10^{-6}, applying research for the best value as before would need too much time.

4.4.1 Dataset 1: camera aligned to the world frame

In this first example, we consider the same dataset used in Section 2.4, therefore we also consider the same region of interest.
Even though we have selected a smaller set of events, the dataset remains very large, containing $4.4 \cdot 10^5$ data. For this reason, running the Steepest Descent algorithm or the Accelerated Gradient Descent algorithm requires many seconds to estimate just one iteration. Therefore to resolve this prob-

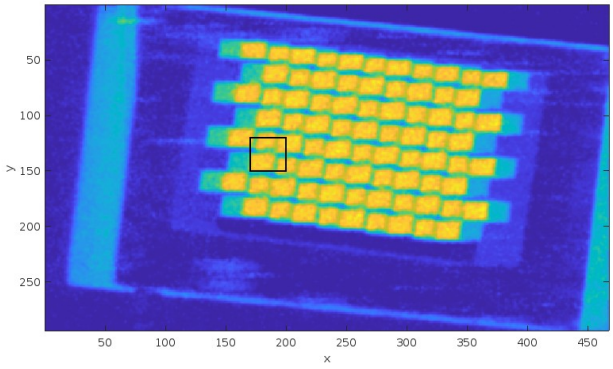

Figure 4.24: Selected spatial and temporal region of interest and selected tile marked in black to which the algorithm is applied.

lem we select a tile of dimension 30 containing $8.8 \cdot 10^3$ events. Figure 4.24 shows the events contained by the region of interest and the selected tile, marked in black.

First, we apply the Steepest Descent algorithm. From Figure 4.25 and 4.26 we can see that the value of the objective function and, as a consequence, the value of the velocities remains near the starting guess, stopping after more than 1000 iterations. Figure 4.27 shows the three images of H associated with the reference times t_{min}, t_{mid}, t_{max}. We can observe that in the last case, the events fall outside the image domain, therefore having more time available, we should enlarge the camera domain.

The Accelerated Proximal Gradient in many more iterations estimates an optical flow that is distant from the starting guess as we can see from Figure 4.28. The value of $F(\mathbf{v})$ that is the sum of the multi-reference function and the regularization term, as is shown in ($\mathcal{B}$), starts from a value of about -15 and reaches the minimum of -30.85 (Figure 4.29). Also from Figure 4.30, we can deduce that this estimation is more precise than the previous one, indeed the events seem to be more aligned: the contrast maximization collects them together in some common points.

Table 4.11 collects the values of the multi-reference focus f and of the total function F for the Proximal Gradient evaluated on the estimated velocity.

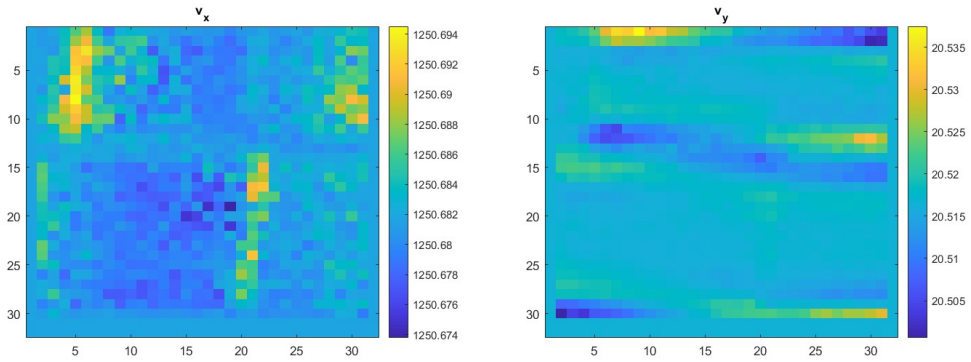

Figure 4.25: Optical flow estimation of the components $\mathbf{v}_x$ (on the left) and $\mathbf{v}_y$ (on the right), applying the Steepest Descent algorithm.

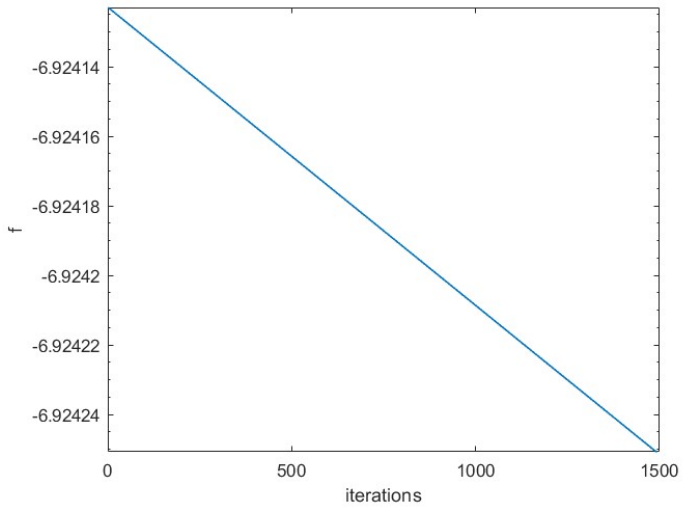

Figure 4.26: Convergence of the multi-reference objective function f along the iterations applying the Steepest Descent algorithm.

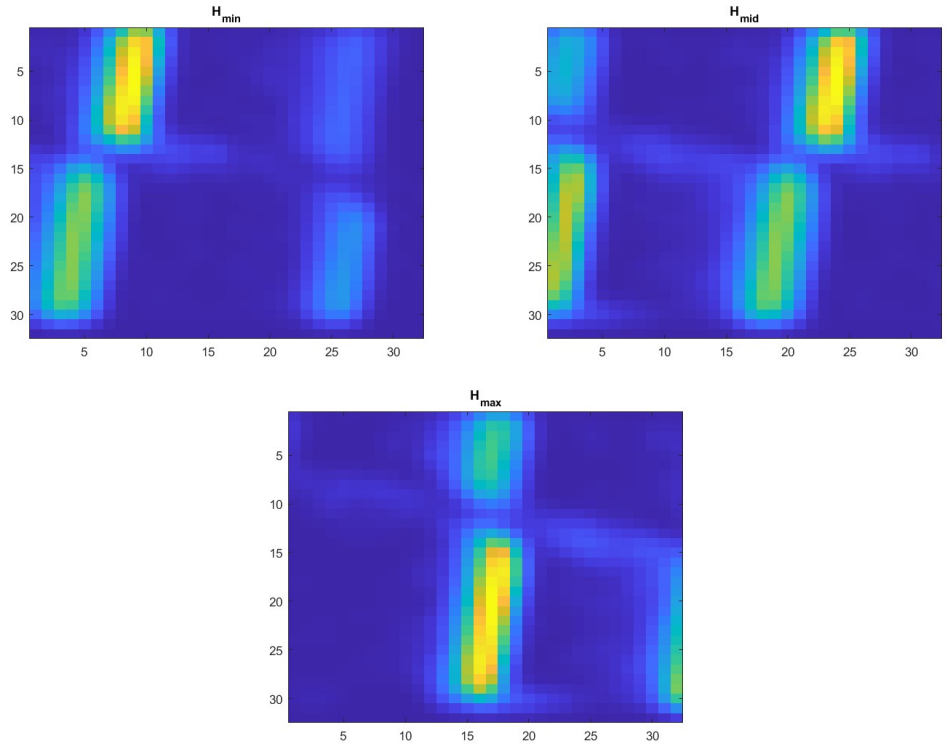

Figure 4.27: Graphical representation of the image H of warped events at the reference times t_{min}, t_{mid}, t_{max} applying the Steepest Descent algorithm.

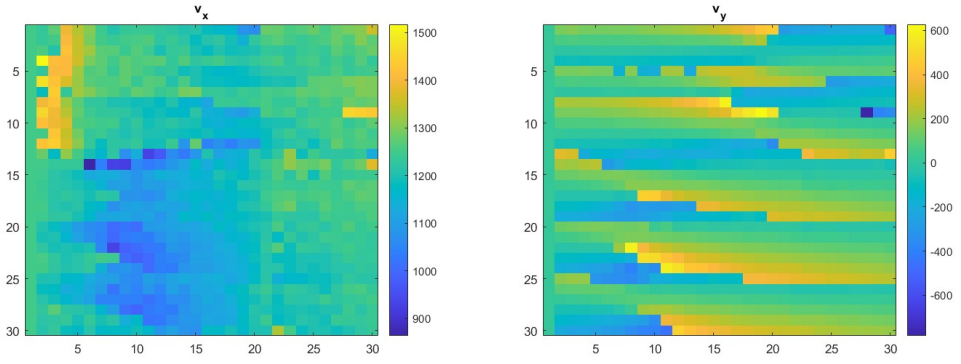

Figure 4.28: Optical flow estimation of the components $\mathbf{v}_x$ (on the left) and $\mathbf{v}_y$ (on the right), applying the Accelerated Proximal Gradient algorithm.

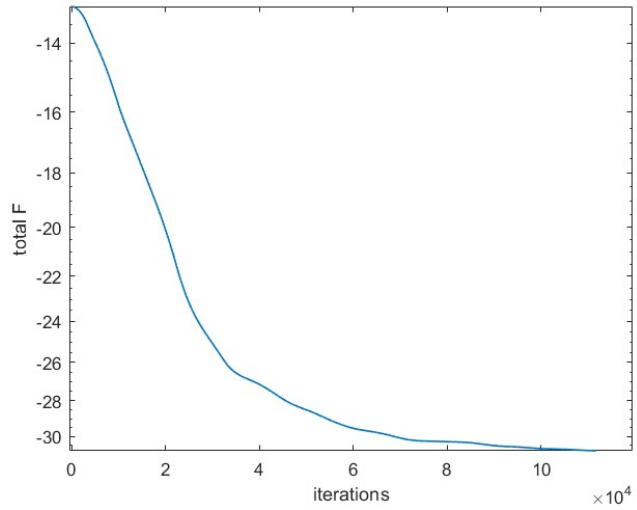

Figure 4.29: Convergence of the total function F along the iterations, applying the Accelerated Proximal Gradient algorithm.

We can observe that the value of f for the Accelerated Proximal Gradient is much lower. Also, we consider the values of G associated with t_{min}, t_{mid}, t_{max}, the number of iterations needed, and the time required. It is clear that the Accelerated Proximal Gradient is much slower, it spends much more time than the Steepest Descent.

4.4.2 Dataset 2: camera rotated with respect to the world frame

We consider now, the second dataset of the events where the camera is inclined with respect to the conveyor. Also in this case the total number of events is high: $2.7 \cdot 10^7$. Therefore we consider a region of interest such that:

- $t_k \in [t_{start}, t_{start} + \Delta t] = [4.24 \cdot 10^6, 4.24 \cdot 10^6 + 3 \cdot 10^4] \mu s \ \forall k = 1, ...N_e$
- $j_k \in [x_{min}, x_{max}] = [410, 940]$ and $i_k \in [y_{min}, y_{max}] = [20, 530] \ \forall k = 1, ...N_e$.

As a consequence, the camera domain is $h = 510$ high and $w = 530$ wide and

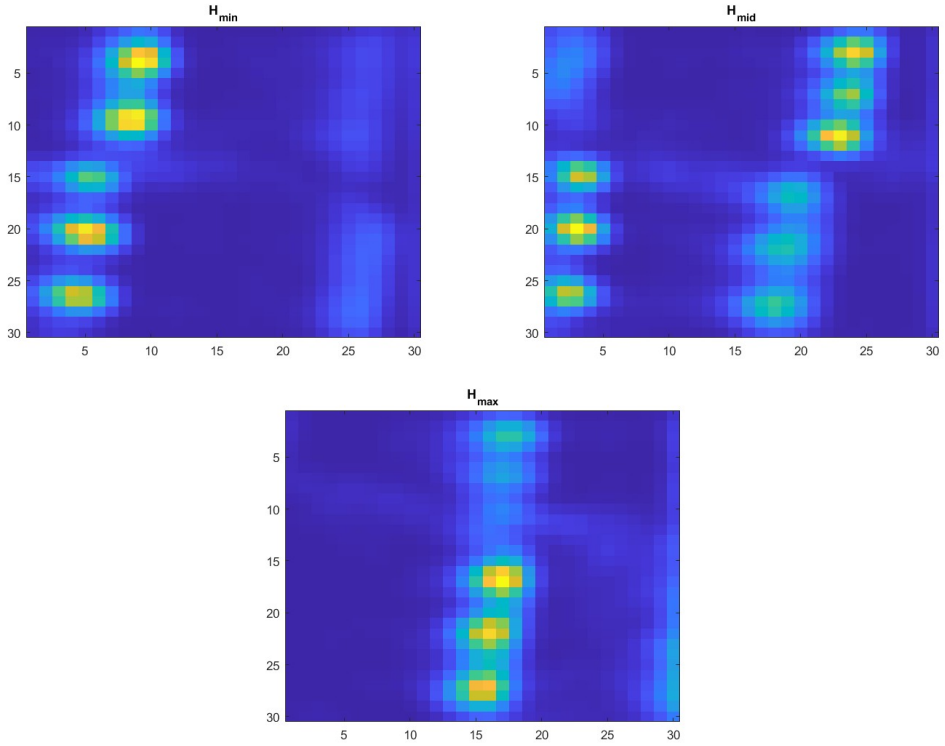

Figure 4.30: Graphical representation of the image H of warped events at the reference times t_{min}, t_{mid}, t_{max} applying the Accelerated Proximal Gradient algorithm.

	SD	APG
$f(\mathbf{v})$	-6.924	-31.020
$F(\mathbf{v})$	/	-30.850
G_{min}	16.302	43.867
G_{mid}	21.292	55.359
G_{max}	10.092	15.186
$iters$	1492	111862
$time$	$6m$	$7h$

Table 4.11: Values for the Steepest Descent Algorithm and the Accelerated Proximal Gradient of the multi-reference focus objective function f, of the total function F, of G associated to the reference times t_{min}, t_{mid}, t_{max}, of the number of iterations and of the time required.

contains $3.8 \cdot 10^5$ events.
In this experiment, we consider a tile of dimension 80×80 because the edges of the parcel are more blurred, and considering a tile with a smaller dimension wouldn't involve enough events to provide a correct estimation. Figure 4.31 is a graphical representation of the events contained in the region of interest and of the selected tile that is marked in black.

First, we apply the Steepest Descent algorithm. In Figure 4.32 there are the results of the two components of the velocities. As before, as we can see also from Figure 4.33, the value of f doesn't change much. Figure 4.34 presents the alignment of the events with respect to the three reference times, reducing the tolerance for the stopping criterium could allow an improvement of the solution.

As the last experiment, we tested the second dataset by applying the Accelerated Proximal Gradient (Figure 4.35). Unfortunately, running this code is much more time-consuming than the previous examples, therefore we set as a maximum number of iterations $k_{max} = 600000$. For this reason, the algorithm stops before converging and reaching the minimum point as we can see from Figure 4.36. During the first 45000 iterations, the algorithm

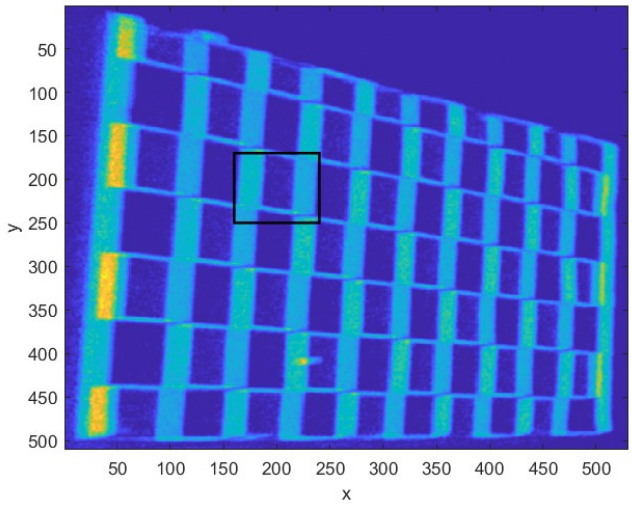

Figure 4.31: Selected spatial and temporal region of interest and selected tile marked in black to which the algorithm is applied.

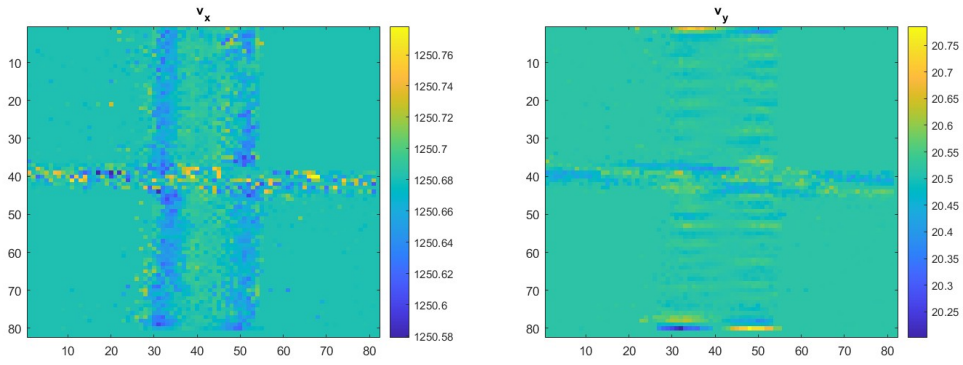

Figure 4.32: Optical flow estimation of the components $\mathbf{v}_x$ (on the left) and $\mathbf{v}_y$ (on the right), applying the Steepest Descent algorithm

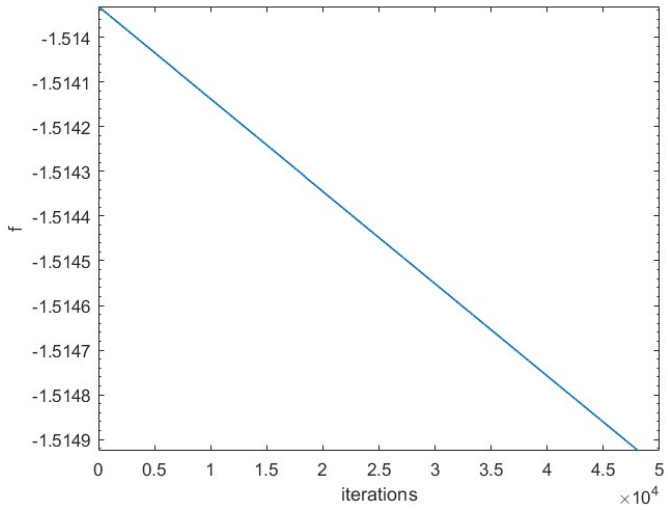

Figure 4.33: Convergence of the multi-reference objective function f along the iterations applying the Steepest Descent algorithm.

decreases fast starting from the value of F of -1.51 and reaching the value of -8.05, then it slows up. Also from Figure 4.37, we can observe that the events are not completely aligned, mainly on the first plot where they are warped at the reference time t_{min}.

Table 4.12 sums up all the results obtained. Also in this case, the Accelerated Proximal Gradient algorithm provides a lower minimum of the objective function f but is much more time-consuming than the Steepest Descent.

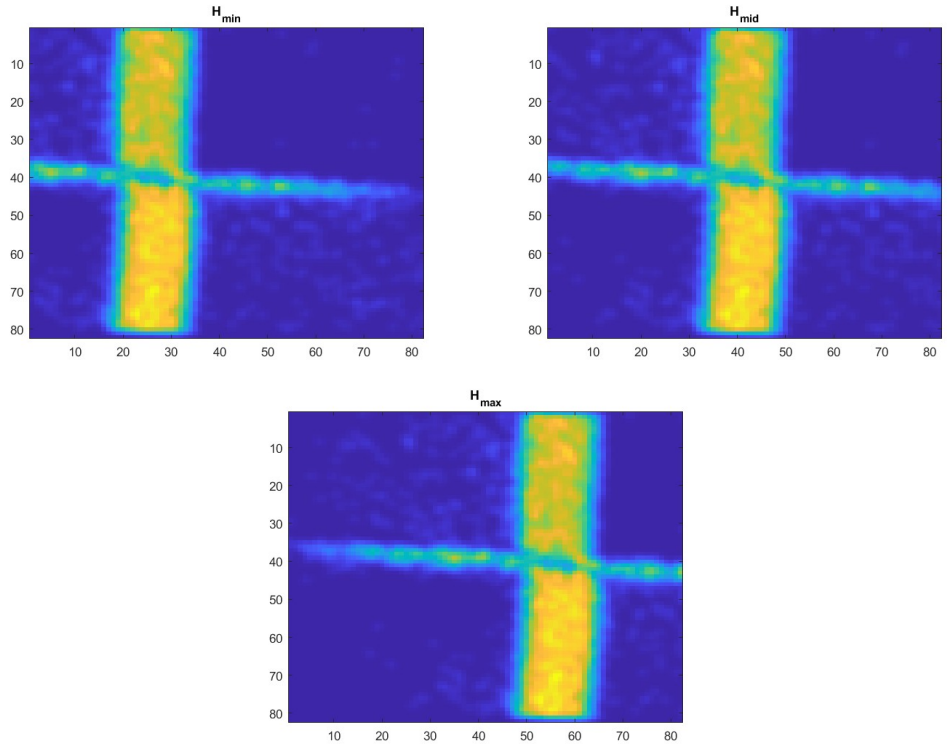

Figure 4.34: Graphical representation of the image H of warped events at the reference time t_{min} applying the Steepest Descent algorithm.

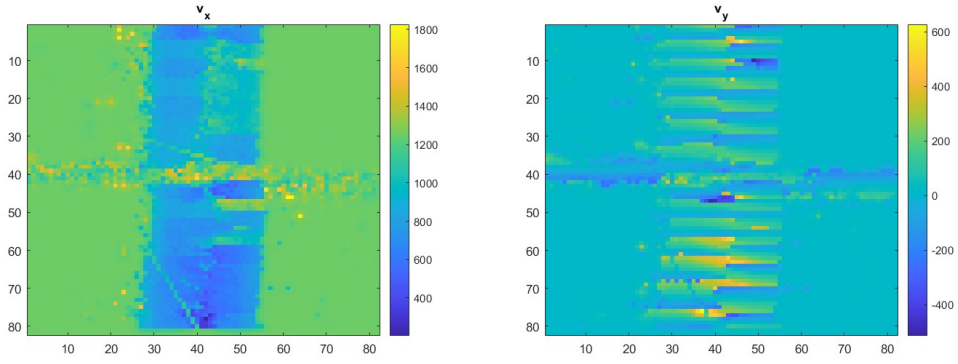

Figure 4.35: Optical flow estimation of the components $\mathbf{v}_x$ (on the left) and $\mathbf{v}_y$ (on the right), applying the Accelerated Proximal Gradient algorithm

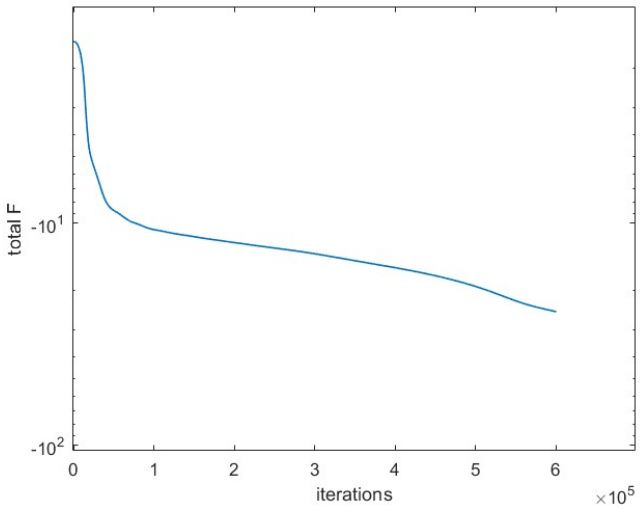

Figure 4.36: Convergence of the total function F along the iterations applying the Accelerated Proximal Gradient algorithm.

	SD	APG
$f(\mathbf{v})$	-1.514	-25.628
$F(\mathbf{v})$	/	-24.957
G_{min}	$1.820 \cdot 10^{-1}$	2.310
G_{mid}	$1.938 \cdot 10^{-1}$	3.792
G_{max}	$1.880 \cdot 10^{-1}$	2.918
$iters$	48109	600000
$time$	$9h$	$87h$

Table 4.12: Values for the Steepest Descent Algorithm and the Accelerated Proximal Gradient of the multi-reference focus objective function f, of the total function F, of G associated to the reference times t_{min}, t_{mid}, t_{max}, of the number of iterations and of the time required.

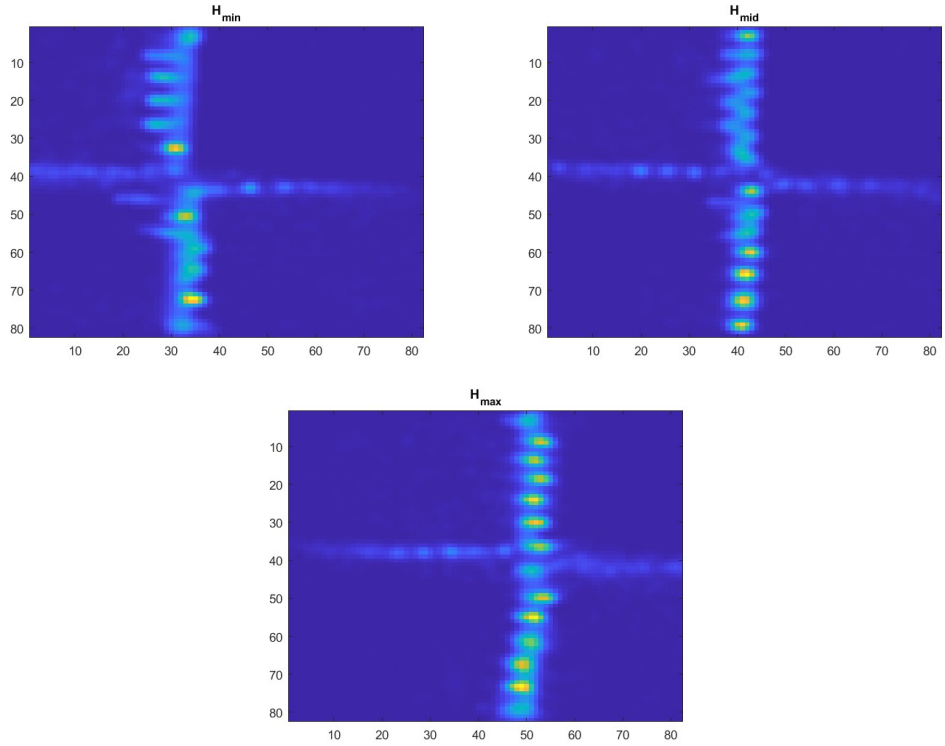

Figure 4.37: Graphical representation of the image H of warped events at the reference time t_{min} applying the Accelerated Proximal Gradient algorithm.

Conclusions and future work

In this project, we have seen how to estimate the optical flow from the events generated by an event camera. One of the main contributions of this work is to estimate an optical flow that is not constant but depends on the pixel location.
Having provided a few applications to synthetic data, we have concluded that the estimation depends a lot on the dimension of the Gaussian kernel support and on the camera domain. Choosing their best values is essential to obtain better estimations.
When we used a synthetic dataset, the best algorithm seemed to be the Steepest Descent, however without the presence of a regularization term, the pixels in which there is no event are not associated with an optical flow. For this reason, when we use real data the Proximal Gradient appears to be more appropriate.

Experiments on real data are very time-consuming and can also require days to terminate a single run, that's why we have to use just a portion of the dataset. In future work, more experiments and tests can be done on synthetic data or mostly on real data.
In particular, it would be interesting to test other experiments changing the dimension and the position of the tile, the dimension of the kernel support, and the standard deviation σ. Moreover enlarging the camera domain with more pixels would probably imply a more accurate estimation, mainly with the Steepest Descent algorithm.
To test the method on real data one could divide the camera domain into

tiles, estimate the dense optical flow for each of them, and then interpolate the results.
Additionally applying other algorithms and/or different regularization terms could improve the results.
Afterward, starting from this thesis, other projects could be done on the same topic. For instance, to obtain a more accurate estimation with real data one could aggregate the events that are generated by the same flow and then estimate the optical flow just for each of these subsets of events.

As we have said in the introduction of this thesis event cameras have a big potential in the estimation of optical flow and have only recently started to be used. Therefore many more studies and insights will be carried out in the following years.

Bibliography

[1] S. Shiba, Y. Klose, Y. Aoki, G. Gallego, *Secrets of event-based optical flow, depth and ego-motion estimation by contrast maximization.* IEE Transactions on Pattern Analysis and Machine Intelligence, pp. 1-18, July 2024.

[2] M. Mahowald and C. Mead, *The silicon retina*, Scientific American, vol. 264, no. 5, pp. 76–83, May 1991.

[3] H. Rebecq, G. Gallego, E. Mueggle and D. Scaramuzza, *EMVS: Event-Based Multi-View Stereo—3D Reconstruction with an Event Camera in Real-Time*, International Journal of Computer Vision, vol. 126, pp. 1394-1414, November 2017.

[4] G. Gallego, "Teaching" *Guillermo Gallego - Event-based Robot Vision*, 2020: https://sites.google.com/view/guillermogallego/teaching/event-based-robot-vision

[5] M. Liu, T. Delbruck, *Adaptive time-slice block-matching optical flow algorithm for dynamic vision sensors* in British Machine Vision Conference (BMVC), pp. 1–12, 2018.

[6] R. Benosman, S.-H. Ieng, C. Clercq, C. Bartolozzi, M. Srinivasan, *Asynchronous frameless event-based optical flow*, Neural Networks, vol. 27, pp. 32–37, 2012.

[7] G. Orchard, R. Benosman, R. Etienne-Cummings, N. V. Thakor, *A spiking neural network architecture for visual motion estimation* in IEEE

Biomedical Circuits and Systems Conference (BioCAS), pp. 298–301, 2013.

[8] T. Brosch, S. Tschechne, H. Neumann, *On event-based optical flow detection* Frontiers in Neuroscience, vol. 9, no. 137, April 2015.

[9] R. Benosman, C. Clercq, X. Lagorce, S.-H. Ieng, C. Bartolozzi, *Event-based visual flow*, IEEE Transactions on Neural Networks and Learning Systems, vol. 25, no. 2, pp. 407–417, 2014.

[10] H. Akolkar, S.-H. Ieng, R. Benosman, *Real-time high speed motion prediction using fast aperture-robust event-driven visual flow*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 44, no. 1, pp. 361–372, 2022.

[11] J. Nagata, Y. Sekikawa, Y. Aoki, *Optical flow estimation by matching time surface with event-based cameras* Sensors, vol. 21, no. 4, 2021.

[12] P. Bardow, A. J. Davison, S. Leutenegger, *Simultaneous optical flow and intensity estimation from an event camera*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 884–892, 2016.

[13] G. Gallego, H. Rebecq, D. Scaramuzza, *A unifying contrast maximization framework for event cameras, with applications to motion, depth, and optical flow estimation*, IEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3867-3876, 2018.

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

[15] G. Gallego, T. Delbruck, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. J. Davison, J. Conradt, K. Daniilidis, D. Scaramuzza, *Event-based vision: a survey*, IEE Transactions on Pattern Analysis and Machine Intelligence, vol. 44, no. 1, pp. 154-180, 2022.

[16] A. Z. Zhu, L. Yuan, K. Chaney, K. Daniilidis, *Unsupervised event-based learning of optical flow, depth, and egomotion*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 989–997, 2019.

[17] J. J. Hagenaars, F. Paredes-Valles, G. C. H. E. de Croon, *Selfsupervised learning of event-based optical flow with spiking neural networks* in Advances in Neural Information Processing Systems (NeurIPS), vol. 34, pp. 7167–7179, 2021.

[18] A. Z. Zhu, L. Yuan, K. Chaney, K. Daniilidis, *EV-FlowNet: Self-supervised optical flow estimation for event-based cameras*, Robotics: Science and Systems (RSS), pp. 1–9, 2018.

[19] M. Gehrig, M. Millh¨ausler, D. Gehrig, D. Scaramuzza, *ERAFT: Dense optical flow from event cameras*, International Conference on 3D Vision (3DV), pp. 197–206, 2021.

[20] Z. Ding, R. Zhao, J. Zhang, T. Gao, R. Xiong, Z. Yu, T. Huang, *Spatio-temporal recurrent networks for event-based optical flow estimation*, AAAI Conference on Artificial Intelligence, vol. 36, no. 1, pp. 525–533, 2022.

[21] C. Lee, A. Kosta, A. Z. Zhu, K. Chaney, K. Daniilidis, K. Roy, *Spike-flownet: Event-based optical flow estimation with energy-efficient hybrid neural networks*, European Conference on Computer Vision (ECCV), pp. 366–382, 2020.

[22] O. Ronneberger, P. Fischer, T. Brox, *U-Net: Convolutional networks for biomedical image segmentation*, International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI), pp. 234–241, 2015.

[23] Z. Teed, J. Deng, *RAFT: Recurrent all pairs field transforms for optical flow*, European Conference on Computer Vision (ECCV), pp. 402–419, 2020.

[24] J. Cuadrado, U. Ranc¸on, B. R. Cottereau, F. Barranco, T. Masquelier, *Optical flow estimation from event-based cameras and spiking neural networks* Frontiers in Neuroscience, vol. 17, p. 1160034, 2023.

[25] F. Paredes-Valles, G. C. H. E. de Croon, *Back to event basics: Self-supervised learning of image reconstruction for event cameras via photometric constancy*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3445–3454, 2021.

[26] H. Liu, G. Chen, S. Qu, Y. Zhang, Z. Li, A. Knoll, and C. Jiang, *TMA: Temporal motion aggregation for event-based optical flow*, International Conference on Computer Vision (ICCV), pp. 9685–9694, October 2023.

[27] D. Gehrig, A. Loquercio, K. G. Derpanis, D. Scaramuzza, *End-to-end learning of representations for asynchronous event based data*, International Conference on Computer Vision (ICCV), pp. 5632–5642, 2019.

[28] T. Stoffregen, C. Scheerlinck, D. Scaramuzza, T. Drummond, N. Barnes, L. Kleeman, R. Mahony, *Reducing the sim-to-real gap for event cameras*, European Conference on Computer Vision (ECCV), pp. 534–549, 2020.

[29] M. Gehrig, M. Muglikar, D. Scaramuzza, *Dense continuous-time optical flow from event cameras*, IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 1–12, 2024.

[30] X. Luo, K. Luo, A. Luo, Z. Wang, P. Tan, S. Liu, *Learning optical flow from event camera with rendered dataset*, International Conference on Computer Vision (ICCV), 2023.

[31] Y. Li, Z. Huang, S. Chen, X. Shi, H. Li, H. Bao, Z. Cui, G. Zhang, *Blinkflow: A dataset to push the limits of event-based optical flow estimation*, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2023.

[32] G. Taverni, D. P. Moeys, C. Li, C. Cavaco, V. Motsnyi, D. S. S. Bello, T. Delbruck, *Front and back illuminated Dynamic and Active Pixel Vision*

Sensors comparison, IEEE Transactions on Circuits Systems II (TCSII), vol. 65, no. 5, pp. 677–681, 2018.

[33] C. Ye, A. Mitrokhin, C. Parameshwara, C. Fermuller, J. A. Yorke, Y. Aloimonos, *Unsupervised learning of dense optical flow, depth, and ego-motion with event-based sensors*, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 5831–5838, 2020.

[34] Y. Tian, J. Andrade-Cetto, *Event transformer FlowNet for optical flow estimation*, British Machine Vision Conference (BMVC), 2022.

[35] F. Paredes-Vall´es, K. Y. Scheper, C. De Wagter, G. C. de Croon, *Taming contrast maximization for learning sequential, low latency, event-based optical flow*, International Conference on Computer Vision (ICCV), pp. 9661–9671, October 2023.

[36] A. Mitrokhin, C. Fermuller, C. Parameshwara, Y. Aloimonos, *Event-based moving object detection and tracking*, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1–9, 2018.

[37] S. K. Nayar, "Linear Camera Model, Camera Calibration", *First Principles of Computer Vision*, Columbia University, 18 April 2021: https://fpcv.cs.columbia.edu/

[38] M. Ng, Z. M. Er, G. S. Soh and S. Foong, *Aggregation functions for simultaneous attitude and image estimation with event cameras at high angular rates*, IEE Robotics and Automation Letters, vol. 7, no. 2, April 2022.

[39] J. Nocedal, S. J. Wright, *Numerical Optimization*, Springer Series in Operations Research and Financial Engineering, 2006.

[40] N. Parikh, S. Boyd, *Proximal Algorithms*, Foundations and Trends in Optimizations, vol. 1, no. 3, pp. 127-239, January 2014.

[41] A. Beck, M. Teboulle, *A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems*, SIAM Journal on Imaging Sciences, vol. 2, no. 1, pp. 183-202, 2009.

[42] A. Beck, *First-order methods in optimization*, SIAM, 2017.

[43] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, *Distributed optimization and statistical learning via the alternating direction method of multipliers*, Foundations and Trends in Machine Learning, vol. 3, no. 1, pp. 1-122, 2010.

[44] T. Pock, *PGMO Lecture: Vision, Learning and Optimization*, lecture 5, February 2020.

[45] D. A. Lorenz, Q. Tran-Dinh, *Non-stationary Douglas-Rachford and alternating direction method of multipliers: adaptive step-sizes and convergence*, Computational Optimization and Applications, vol. 74, pp. 67-92, May 2019.

Ringraziamenti

Vorrei ringraziare prima di tutto il Professor Lanza che si è sempre reso molto disponibile partecipando con entusiasmo e interessamento allo sviluppo di questo progetto. Con i numerosi pomeriggi che mi ha dedicato ha reso evidente la passione che mette nel suo lavoro, trasmettendomi di conseguenza maggior consapevolezza e fiducia nelle mie capacità.
Ringrazio inoltre Martino che con i suoi precisi consigli mi ha guidato durante il percorso di tirocinio e il team R&D di Datalogic che mi ha accolto calorosamente permettendomi di fare una bellissima esperienza.

Un ringraziamento speciale va alla mia famiglia che mi è sempre stata accanto spronandomi a dare il massimo in ogni momento, anche quando gli esami non sempre andavano bene. E' stato un percorso difficile e il vostro supporto è stato fondamentale.
Ringrazio Greg, non avrei potuto condividere questi 5 anni di matematica e di vita con una persona migliore, percorrerli insieme a te li ha resi molto più leggeri e divertenti.
Grazie a Martina, Lucia ed Elena che mi sono state sempre vicine.