Technical Report

IRI-TR 1607

Towards SLAM with an events-based camera

Ignasi Clavera
Advisors:
Joan Solà
Juan Andrade-Cetto

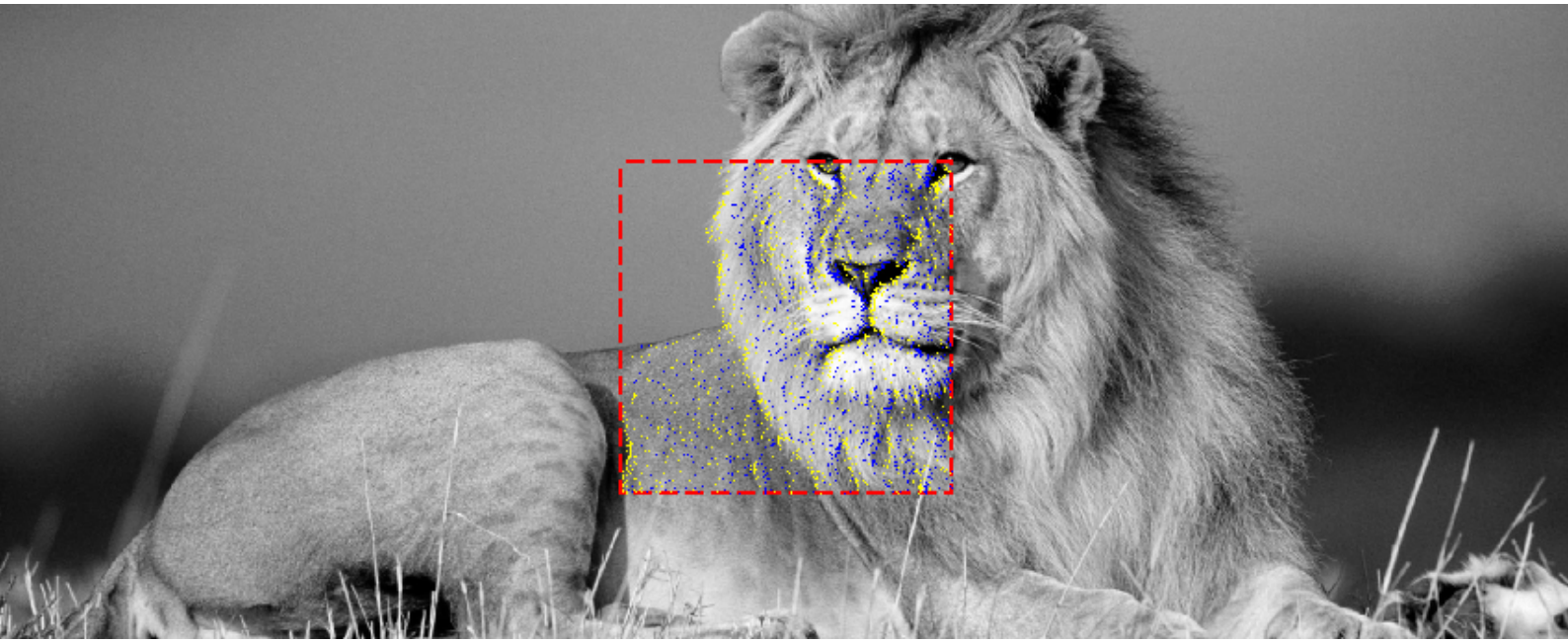
July 2016

Institut de Robòtica i Informàtica Industrial

Towards SLAM with an Event-based Camera

Ignasi Clavera Gilaberte

July 2016



Contents

1 Introduction **2**

2 Preliminaries **2**
2.1 Generative event model . . . 2
2.2 Scene and motion modeling . . . 2
2.3 Measurement equation . . . 3

3 Diferent Approaches **4**
3.1 Event-Based Image Recovering with Known Velocity . . . 4
3.1.1 Observations . . . 5
3.2 Event-Based Image Recovering . . . 5
3.2.1 Observations . . . 6
3.3 Information Filter . . . 6
3.3.1 Observations . . . 8

4 Glossary **8**

1 Introduction

Event-based cameras have an incredible potential in real-time and real-world robotics. They would enable more efficient algorithms in applications where high demanding requirements, such as rapid dynamic motion and high dynamic range, make standard cameras run into problems rapid dynamic motion and high dynamic range. While traditional cameras are based in the frame-base paradigm - a shutter captures a certain amount of pictures per second -, the bio-inspired event cameras have pixels that respond independently to the change of log-intensity generating asynchronous events. An special appeal for this type of cameras is their low band-width, since the stream of events contain all the information getting rid of the redundancy. This sensors that mimic some properties of the human retina has microseconds latency and 120 dB dynamic range (in contrast to the 60 dB of the standard cameras).

However, the current impact of the event cameras has been tiny due to the necessity of completely new algorithm, there is no global measurement of the intensity which would allow the use of current methods. The fact that an event corresponds to an asynchronous local intensity difference turns out to be a challenging problem if one wants to recover the motion as well as the scene. This article tries to illustrate the several problems that are needed to face when dealing with this problem and some of the different approaches taken.

First of all, we will explain the generative model of the event camera and the preliminaries, followed by the different approaches. Finally will the conclusions and a glossary of the code.

2 Preliminaries

In this section we will explain the definitions and assumptions considered in all the different approaches we have attempted in order to solve the problem stated.

2.1 Generative event model

The pixels of the event camera are completely independent of each other that spikes events in continuous time. Each pixel measure the logarithmic change of intensity, when this logarithmic change is equal to a pre-defined threshold it fire an event time-stamped relative to a global clock.

One event is defined as a tuple $\mathbf{e}_j = (u_j, v_j, \rho_j, t_j)$, where $\mathbf{p_j} = (u_j, v_j)^\top$ are the pixel location in the camera frame, so $\mathbf{p_j} \in \Omega$ being Ω the camera domain. In our specific case $\Omega = [128] \times [128]$ since the camera has a resolution of 128x128 pixels, but we will refer the domain as Ω for a more general case. We refer as t_k the time-stamp to microsecond resolution of the j-th event and $\rho = \pm 1$ is its polarity, the sign of the brightness change. If we regard the event camera as an ideal sensor, then an event is fired just when the log intensity change value is equal to a threshold C:

$$|log(I(\mathbf{p}, t)) - log(I(\mathbf{p}, \tau(\mathbf{p}, t)))| \geq C \tag{1}$$

Where $I(\mathbf{p}, t)$ is the intensity at the pixel $\mathbf{p}$ at time t and $\tau_\mathbf{p}$ is the time when the previous event at time t occurred in the pixel $\mathbf{p}$. The polarity ρ of an event will be 1 whenever $\log(I(\mathbf{p}, t)) - log(I(\mathbf{p}, \tau(\mathbf{p}, t))) \geq C$ and -1 when $\log(I(\mathbf{p}, t)) - log(I(\mathbf{p}, \tau(\mathbf{p}, t))) \leq -C$.

2.2 Scene and motion modeling

We assume that the camera is seeing a 2D flat world represented with a surface S, the scene. The event-camera has the same optics as a pinhole model. Therefore, the projection operation is described by $\mathbf{p} = \pi(\mathbf{x})$, where $\mathbf{p} = (u, v)$ is the location of the pixel in the camera frame and $\mathbf{x} = (x, y)$ is the 2D point in the scene S.

The scene S is modeled as static and since our future goal would be to estimate the gradients, $\mathbf{g}^S : S \to \mathbb{R}^2$, we assume that are invariant through time. However, the gradients observed by the event-camera do change with the time: $\mathbf{g} : \Omega \times \mathbb{R} \to \mathbb{R}^2$ and the static assumption can be written in the camera frame such as:

$$\mathbf{g}(\mathbf{p} - \mathbf{u}\Delta t, t + \Delta t) = \mathbf{g}(\mathbf{p}, t) \tag{2}$$

Equation 2 basically says that two pixels that were looking in a different instant at the same point in the world saw the same. In this equation $\mathbf{u}$ is the velocity of the camera in pixels/second - event-cameras have not notion of frame-to-frame displacement. Equation 2 using the intensity I instead of the gradient is a common start point for optical flow estimation and the displaced frame is often approximate with it first-order Taylor series:

$$\mathbf{g}(\mathbf{p} - \mathbf{u}\Delta t, t + \Delta t) \approx \mathbf{g}(\mathbf{p}, t) - \mathbf{g_p}(\mathbf{p}, t)\mathbf{u}\Delta t + \mathbf{g_t}(\mathbf{p}, t)\Delta t \tag{3}$$

Therefore, equation 2 results in:

$$\mathbf{g_p}(\mathbf{p},t)\mathbf{u} - \mathbf{g_t}(\mathbf{p},t) = \mathbf{0} \tag{4}$$

For brevity and clarity we have write the partial derivatives with subscripts, e.g.: $\mathbf{g_t} \triangleq \frac{\partial \mathbf{g}}{\partial t}$.

The have considered that the movement of the camera is just a translation, therefore the velocity of all the pixels remains equal. The aim of this assumption is to be a baseline for the initial theoretical development because it should be easy to extrapolate to the more general case. Furthermore, as typical SLAM approach we chose a motion model given by a constant velocity. This holds true for smooth motions, which given the extreme low latency of the event camera is easy to satisfy.

2.3 Measurement equation

This section we will describe the measurement equation $h(\mathbf{g}, \mathbf{u})$. The Equation presented in 1 is not a good choice for modeling events, since the measurement function would be discrete valued - not even continuous - and would not appear the velocity $\mathbf{u}$. Therefore, our goal has been to justify another appropriate choice of measurement function which was used in related works [8]. For the sake of simplicity we will consider from now on that the event camera measures the change of intensity instead of the change of log-intensity - it is just a modification on the scale with no significant impact. Therefore, we will write Equation 1 as: $|I(\mathbf{p},t) - I(\mathbf{p},\tau(\mathbf{p},t))| \geq C$. The LHS of the inequality can be regarded such as:

$$\begin{aligned} I(\mathbf{p},t) - I(\mathbf{p},\tau(\mathbf{p},t)) =& I(\mathbf{p},t) - I(\mathbf{p},t-(t-\tau(\mathbf{p},t))) \overset{(*)}{\approx} I_t(\mathbf{p},t)(t-\tau(\mathbf{p},t)) \overset{(**)}{=} \\ & \langle I_{\mathbf{p}}(\mathbf{p},t), \mathbf{p}_t \rangle (t-\tau(\mathbf{p},t)) = \langle \mathbf{g}(\mathbf{p},t), \mathbf{u}(t-\tau(\mathbf{p},t)) \rangle \end{aligned} \tag{5}$$

In $(*)$ it is used that due to the low latency of the event camera t and $\tau(\mathbf{p},t)$ would be relatively similar and therefore we can approximate the difference of the intensity in those times with the derivative. In $(**)$ we have used the brightness constancy of the scene which results from Equation 4 but substituting $\mathbf{g}$ for I and Δt for $(t-\tau(\mathbf{p},t))$ and acknowledging that $I_{\mathbf{p}} = \mathbf{g}$ and $\mathbf{p}_t = \mathbf{u}$.

Therefore, an event would be fired when:

$$\langle \mathbf{g}(\mathbf{p},t), \mathbf{u} \rangle (t-\tau(\mathbf{p},t)) \geq C \tag{6}$$

We will denote $z^k(\mathbf{p})$ the measurement of the camera in the pixel $\mathbf{p}$ which will be the sum of all the events - with its polarity - during the interval $t = t_0 + k\Delta t$, where t_0 and $\Delta t \in \mathbb{R}$ are constants. Consequently, $z^k(\mathbf{p})$ is expressed as:

$$z^k(\mathbf{p}) = \sum_{\substack{\mathbf{e}_j = (\mathbf{p}_j, \rho_j, t_j) \\ t_0 + (k-1)\Delta t \leq t_j < t_0 + k\Delta t}} \rho_j \tag{7}$$

As an event is fired just when the change of brightness is higher than the brightness then, $\rho = \frac{I(\mathbf{p},t) - I(\mathbf{p},\tau(\mathbf{p},t))}{C}$. By Equation 5 we are lead to the following measurement model function:

$$\overline{h}(\mathbf{p},\mathbf{g},\mathbf{u}) = \frac{\langle \mathbf{g}(\mathbf{p},t), \mathbf{u} \rangle}{C} (t-\tau(\mathbf{p},t)) \tag{8}$$

Integrating the expression in Equation 8:

$$h(\mathbf{p},\mathbf{g},\mathbf{u}) = \int_{t_0+(k-1)\Delta t}^{t_0+k\Delta t} \frac{\langle \mathbf{g}(\mathbf{p},t), \mathbf{u} \rangle}{C} (t-\tau(\mathbf{p},t)) dt \approx \frac{\langle \mathbf{g}(\mathbf{p}, t_0 + (k-1)\Delta), \mathbf{u} \rangle}{C} \widehat{\Delta\tau} \Delta t \tag{9}$$

In Equation 9 we have approximate the integral by the initial value times the width of the interval. Besides, we have considered the term $t - \tau(\mathbf{p},t)$ to be constant an equal to a certain temporal window $\widehat{\Delta\tau}$. This equation gives us a differential form for our measurement.

Since Equation 8 and 9 is a hard decision model for the generation of events - it does not take into account the sensor noise, manufacturing errors nor the approximations in the model. In [1] show that the corresponding probability density averaged over all pixels is an unimodal Gaussian-like distribution, which its standard deviation is a function of the threshold C. Finally, our model is represented by Equation 10.

$$z(\mathbf{p}) = h(\mathbf{p},\mathbf{g},\mathbf{u}) + v \tag{10}$$

Where $v \sim \mathcal{N}(0, V(C))$

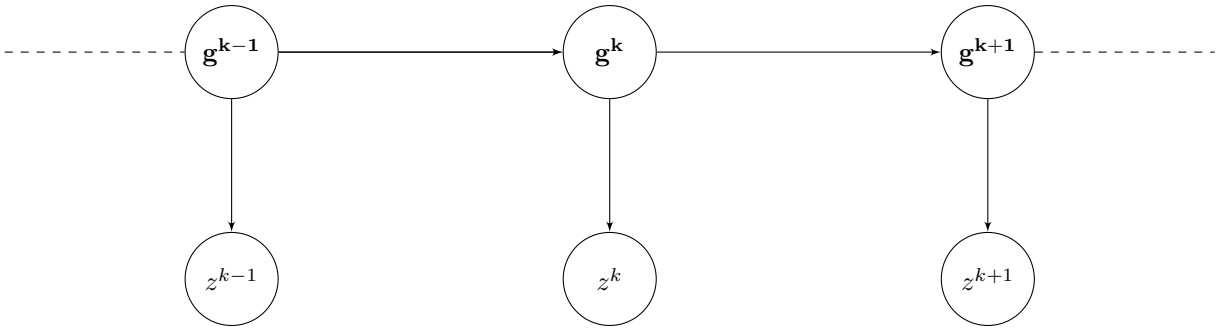


Figure 1: Dynamic Bayes Network with the velocity known

3 Diferent Approaches

In this section we explain the different approaches we took in order to reconstruct the gradient scene. The first one is accomplished by giving the velocity as known data, while the second approach uses the same method than the first one but with regarding the velocity as an unknown variable. Finally, the third approach uses an information filter to recover both, the velocity and the gradients. In each method we will explain the difficulties and the further steps that can be made in each direction.

3.1 Event-Based Image Recovering with Known Velocity

This approach assumes that the real velocity of the camera is known and its objective is to recover the gradient of the image. We represent the problem as a Dynamic Bayes Network (DBN) and for its resolution we use an optimization method in order to minimize a cost function made up as a sum of errors. Subsequently we perform a Extended Kalman Filter (EKF) update in the covariance of the variables.
The DBN is a probabilistic graphical model that represents a set of random variables and their conditional dependencies via a directed acyclic graph, which represents the conditional dependencies between variables through the direction of the arrows [4]. The assumptions made in Section 2 allows us to mathematically write the dependencies between variables. First of all, the equation 2 derived from the static scene assumption allow us to relate the gradients of the image in subsequent instants of time. However, this equation is usually too strong and is wiser to introduce a Gaussian white noise yielding a softer model. Therefore our model is ended up represented by Equation 11.

$$\mathbf{g}^k(\mathbf{p} - \mathbf{u}\Delta t) = \mathbf{g}^{k-1}(\mathbf{p}) + \mathbf{w}^k \tag{11}$$

Where $\mathbf{w}^k \sim \mathcal{N}(\mathbf{0}, \mathbf{W}^k)$. So the gradient just depends on the gradient in the posterior time step where this equation holds,i.e, in the gradients where $\mathbf{p} - \mathbf{u}\Delta t \notin \Omega$ it is not true. On the other hand, the measurement equation is:

$$z^k = h(\mathbf{p}, \mathbf{g}^k, \mathbf{u}^k) + v^k = \frac{\langle \mathbf{g}^k(\mathbf{p}, t), \mathbf{u}^k \rangle}{C}(t - \tau(\mathbf{p}, t)) + v^k \tag{12}$$

Where $v^k \sim \mathcal{N}(\mathbf{0}, \mathbf{V}^k)$. These two equation lead to the DBN in Figure 1.

A DBN contains all dependencies between variables and we can write the joint probability as a product of all the conditionals. In Equation 13 has made a notational abuse, since it lacks that the probability is $\forall \mathbf{p} \in \Omega$

$$P(\{\mathbf{g}^k, z^k\}_{k=1}^K) \propto \prod_{k=1}^{K} P(\mathbf{g}^k|\mathbf{g}^{k-1})P(z^k|\mathbf{g}^k) \tag{13}$$

Therefore, using the notation in [4]:

$$\phi_i = P(\mathbf{g}^i|\mathbf{g}^{i-1}) \propto exp\left(-\frac{1}{2}\left(\mathbf{g}^i - \mathbf{g}^{i-1}(\mathbf{p} + \mathbf{u}\boldsymbol{\Delta}\mathbf{t})\right)^T \mathbf{W}^{i-1}\left(\mathbf{g}^i - \mathbf{g}^{i-1}(\mathbf{p} + \mathbf{u}\boldsymbol{\Delta}\mathbf{t})\right)\right) = exp\left(-\frac{1}{2}\mathbf{e}^{i\,T}\mathbf{W}^{i-1}\mathbf{e}^i\right) \tag{14}$$

$$\phi_k = P(z^k|\mathbf{g}^k) \propto exp\left(-\frac{1}{2}\left(z^k - h(\mathbf{p}, \mathbf{g}, \mathbf{u})\right)\mathbf{V}^{k\,-1}\left(z^k - h(\mathbf{p}, \mathbf{g}, \mathbf{u})\right)\right) = exp\left(-\frac{1}{2}e^{k\,T}\mathbf{V}^{k\,-1}e^k\right) \tag{15}$$

Consequently, the overall joint probability can be written as Equation 16. We have joined the subscripts in a unique one k and expressing $\mathbf{W}^i$ and $\mathbf{V}^j$ as $\mathbf{Y}^k$.

$$P(\{\mathbf{g}^k, z^k\}_{k=1}^K) \propto \prod \phi_k = \prod exp\left(-\frac{1}{2}\mathbf{e}^{k\,T}\mathbf{Y}^{k\,-1}\mathbf{e}^k\right) \tag{16}$$

Maximizing this PDF is just the same as minimizing its *negative log − likelihood*, named cost:

$$\mathbf{F}(\{\mathbf{g}^k\}_{k=1}^K) \triangleq -log\left(P(\{\mathbf{g}^k, z^k\}_{k=1}^K)\right) = \sum \frac{1}{2}\mathbf{e}^{k^T}\mathbf{Y}^{k^{-1}}\mathbf{e}^k \tag{17}$$

And the result is given by:

$$\{\mathbf{g}^k\}_{k=1}^{K*} = \arg\min \sum \frac{1}{2}\mathbf{e}^{k^T}\mathbf{Y}^{k^{-1}}\mathbf{e}^k \tag{18}$$

Once the minimization step is made we perform the EKF update step in the covariances. Assuming that the random variables $\mathbf{g}^k$ are normally distributed, i.e, $\mathbf{g}^k \sim \mathcal{N}\left(\hat{\mathbf{g}}^k, \mathbf{G}^k\right)$, we can proceed using the EKF algorithm. In order to make the lecture easier we will write $\mathbf{q} = \mathbf{p} + \mathbf{u}\Delta t$ and $\mathbf{G}(\mathbf{p})$ the covariancie in the point $\mathbf{p}$.

$$\mathbf{G}^{k|k-1}(\mathbf{p}) = \mathbf{G}^{k-1}(\mathbf{q}) + \mathbf{W}^k \tag{19}$$

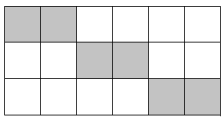
$$\mathbf{S}^k = \mathbf{H}^{k^T}\mathbf{G}^{k|k-1}\mathbf{H}^k + \mathbf{V}^k \tag{20}$$

$$\mathbf{G}^{k|k} = \left(\mathbf{I} - \mathbf{G}^{k|k-1}\mathbf{H}^{k^T}\mathbf{S}^{k^{-1}}\mathbf{H}^k\right)\mathbf{G}^{k|k-1} \tag{21}$$

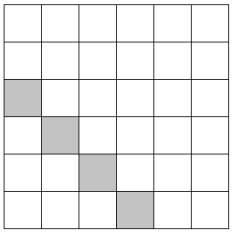
Where $\mathbf{I}$ is the identity matrix and $\mathbf{H}^k$ is the jacobian of h with respect $\mathbf{g}$, i.e $h_{\mathbf{g}}$ in the previous notation.

3.1.1 Observations

The most important highlight in this method is that the equations are completely linear, therefore the optimum exists and is unique. Since the velocity is known the scalar product in the measurement equation became a linear function its matrix can be seen in Figure 2. Moreover, the function that relates in the gradient at the time step k with the gradient at a different time step is linear and sparse. The non-zeros entries of this matrix depends just in the direction that the camera is moving and the approximation function we are using (interpolation, closer neighbor).



(a) Measurement matrix of a 3 pixels camera



(b) Update gradient matrix of a 3 pixels camera with a vertical movement and neighbor approximation

Figure 2: Matrices of the algorithm: gray squares indicate non-zeros entries

3.2 Event-Based Image Recovering

In this section we will explain the generalization of the former approach in order to recover the velocity and therefore make a huge step towards the SLAM with an event-based camera. However, due to the non-linearity of the equations along with the the size of our state vector and the density of the covariances the problem became impossible. The generalization uses the same equations than the last approach assuming constant velocity and regarding the velocity as a variable.

Consequently, the motion equation, Equation 22, gives us a constraint of the variable $\mathbf{u}$ that can be expressed as an error with mean zero and covariance $\mathbf{Q}$.

$$\mathbf{u}^k = \mathbf{u}^{k-1} + \mathbf{q}^k \tag{22}$$

Where $\mathbf{u}^k$ is the velocity at the time step k and $\mathbf{q}^k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}^k)$ is a white noise.

Adding this equation to the former system results in the same cost function, Equation 17 with one more term - because we are assuming that the velocity is the same in all the pixels. That new term is for a certain time step k:

$$\mathbf{e}^k = \mathbf{u}^k - \mathbf{u}^{k-1}\mathbf{Y}^k = \mathbf{Q}^k \tag{23}$$

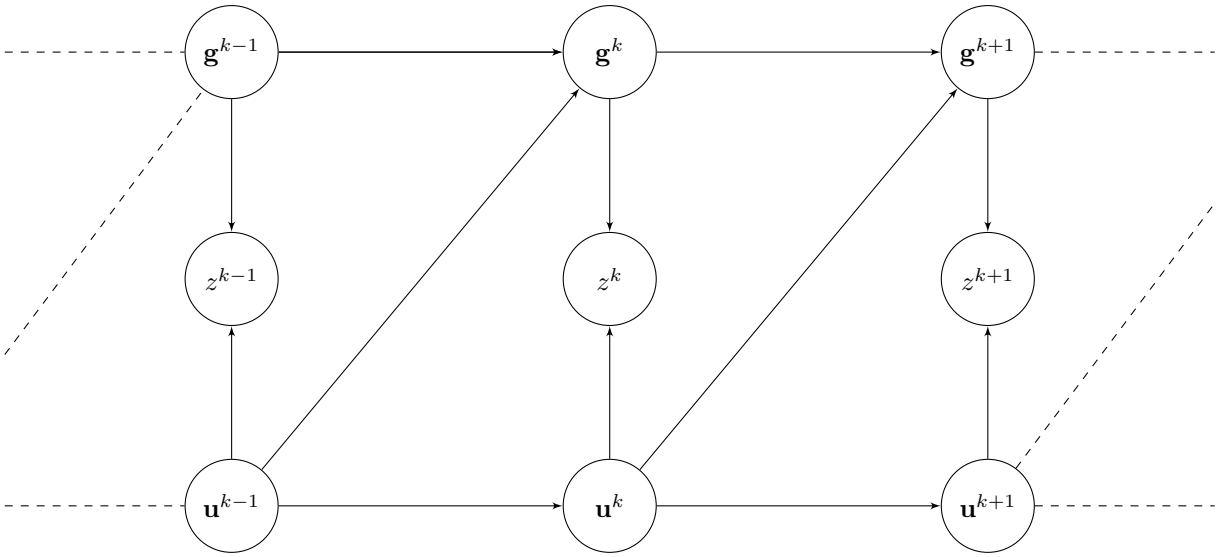


Figure 3: Dynamic Bayes Network with velocity and gradients as variables

The DBN which considers the velocity as variable is shown in Figure 3.

As we did in the first approach, after the minimization step we carry out the EKF update step. The assumption made are that the gradient $\mathbf{g}^k$ are normally distributed, i.e, $\mathbf{g}^k \sim \mathcal{N}\left(\hat{\mathbf{g}}^k, \mathbf{G}^k\right)$, and the same for the velocity $\mathbf{u}^k \sim \mathcal{N}\left(\hat{\mathbf{u}}^k, \mathbf{U}^k\right)$. We consider $\mathbf{q} = \mathbf{p} + \hat{\mathbf{u}}\Delta t$ and $\mathbf{G}(\mathbf{p})$ the covariance in the point $\mathbf{p}$.

$$\mathbf{G}^{k|k-1}(\mathbf{p}) = \mathbf{G}^{k-1}(\mathbf{q}) + \left(\Delta t\ \mathbf{g}_{\mathbf{p}}(\mathbf{q})\right)^T \mathbf{U}^{k-1}(\mathbf{p})\left(\Delta t\ \mathbf{g}_{\mathbf{p}}(\mathbf{q})\right) + \mathbf{W}^k \tag{24}$$

Where we have denoted $\mathbf{g}_{\mathbf{p}}(\mathbf{q})$ the derivative of the gradient with respect the camera frame coordinates. This update equation results from the first order linearization of the Equation 11.

$$\mathbf{U}^{k|k-1} = \mathbf{U}^{k-1} + \mathbf{Q}^k \tag{25}$$

$$\mathbf{S}^k = \mathbf{H}^{k^T}\mathbf{B}^{k|k-1}\mathbf{H}^k + \mathbf{V}^k \tag{26}$$

$$\mathbf{B}^{k|k} = \left(\mathbf{I} - \mathbf{B}^{k|k-1}\mathbf{H}^{k^T}\mathbf{S}^{k^{-1}}\mathbf{H}^k\right)\mathbf{B}^{k|k-1} \tag{27}$$

Where $\mathbf{B}$ is a diagonal block matrix with two blocks: the first one the covariance $\mathbf{G}$ and the second one the covariance $\mathbf{U}$. As before, $\mathbf{I}$ is the identity matrix and $\mathbf{H}^k$ is the jacobian of h with respect $\mathbf{g}$ and $\mathbf{u}$.

3.2.1 Observations

Similar methods have been used successfully using full EKF or bilinear optimization [1] [5]. However, our approach resulted too dense in practice and we never accomplished to track precisely the velocity. One modification in this algorithm that could result in a faster convergence would be to use Equation 4 instead of 11. Hence, the gradient and velocity would be linearly related.

In this approach the matrices are still sparse, the new matrix $\mathbf{H}$ is shown in 4.

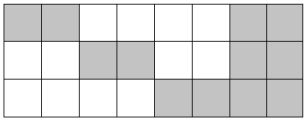


Figure 4: $\mathbf{H}$ matrix with velocity as variable with a 3 pixels camera

3.3 Information Filter

In this last approach, we attempted to use an Extended Information Filter (EIF), the dual algorithm of EKF. The EIF has been used in its sparse form, Sparse Extended Information Filters (SEIF), successfully in SLAM. It allows to exploit the sparsity of the system being nearly constant in time, so irrespective of the size of the map [6]. The EIF algorithm in our problem was designed to allow us to work with sparse inverse of covariance

matrices. However, in its implementation resulted that such matrices became full and therefore ending up with an algorithm no longer useful for real-time robotics.

Below is explained the EIF applied to our problem of recovering both: the gradients and the velocity. Following is described the future modifications and tips to make this problem more clear and understandable. For a more detailed explanation of the equations and steps refer to [7].

The state vector is compromised by the gradients and the velocity at the current time-step, $\mathbf{x}^k = [{\mathbf{g}^k}^T, {\mathbf{u}^k}^T]^T$. We take the first-order linearization of the out models and treat uncertainty as independent of the data - white Gaussian noise. This results in the state vector following a Gaussian distribution.

$$\mathbf{x}^k \sim \mathcal{N}\left(\hat{\mathbf{x}}^k, \mathbf{\Sigma}^k\right) = \mathcal{N}^{-1}\left(\eta^k, \mathbf{\Lambda}^k\right) \tag{28}$$

I. *Measurement Update Step*

The measurament model, Equation 8, is a non-linear function of the state with a Gaussian noise. The update equation follows from the Bayes rule:

$$p(\mathbf{x}^k|\mathbf{z}^k) \propto p(\mathbf{z}^k|\mathbf{x}^k, \mathbf{z}^{k-1})p(\mathbf{x}^k|\mathbf{z}^{k-1}) = p(\mathbf{z}^k|\mathbf{x}^k)p(\mathbf{x}^k|\mathbf{z}^{k-1}) \tag{29}$$

We update the current distribution $p(\mathbf{x}^k|\mathbf{z}^{k-1} = \mathcal{N}^{-1}\left(\overline{\eta}^k, \overline{\Lambda}^k\right)$ to the new posterior canonical form through the following step:

$$\mathbf{\Lambda}^k = \overline{\mathbf{\Lambda}}^k + {\mathbf{H}^k}^T \mathbf{V}^{-1} \mathbf{H}^k \tag{30}$$

$$\eta^k = \overline{\eta}^k + {\mathbf{H}^k}^T \mathbf{V}^{-1}\left(\mathbf{z}^k - \mathbf{h}^k(\overline{\mathbf{x}}^k) + \mathbf{H}^k \overline{\mathbf{x}}^k\right) \tag{31}$$

The $\mathbf{H}$ matrix is the same than in the former method - the jacobian of the measurement equation - and it is shown in Figure 4 which is zero everywhere except in the two main diagonals and the two last columns.

II. *State Augmentation Step*

This step can be seen as a two step process: a state augmentation with a random variable representing the new gradients and the new velocity and a marginalization over the old state vector. We will regard as $\mathbf{R}$ as the covariance of the Gaussian noise, $\mathbf{r}$, in the equations that relates the new state with the previous one.

$$\mathbf{x}^{k+1} = \mathbf{f}\left(\mathbf{x}^k\right) + \mathbf{r}^k \approx \mathbf{f}\left(\hat{\mathbf{x}}^k\right) + \mathbf{F}^k\left(\mathbf{x}^k - \hat{\mathbf{x}}^k\right) + \mathbf{r}^k \tag{32}$$

Where $\mathbf{f}$ is:

$$\mathbf{x}^{k+1} = \begin{bmatrix} \mathbf{g}^{k+1}([1,1]^T) \\ \vdots \\ \mathbf{g}^k(\mathbf{p}) \\ \vdots \\ \mathbf{u}^{k+1} \end{bmatrix} = \mathbf{f}\left(\mathbf{x}^k\right) = \begin{bmatrix} \mathbf{g}^k([1,1]^T + \mathbf{u}\Delta t) \\ \vdots \\ \mathbf{g}^k(\mathbf{p} + \mathbf{u}\Delta t) \\ \vdots \\ \mathbf{u}^k \end{bmatrix} \tag{33}$$

And $\mathbf{F}$ expresses the jacobian of the function $\mathbf{f}$. The marginalization of the former state, $\mathbf{x}^k$, give us the desired distribution over $\mathbf{x}^{k+1}$.

$$p(\mathbf{x}^{k+1}|\mathbf{z}^k) = \int_{\mathbf{x}^k} p(\mathbf{x}^{k+1}, \mathbf{x}^k|\mathbf{z}^k) d\mathbf{x}^k \tag{34}$$

$$\overline{\mathbf{\Lambda}}^{k+1} = {\mathbf{R}^k}^{-1} - \left({\mathbf{R}^k}^{-1}\mathbf{F}^k\right)^T \left(\mathbf{\Lambda}^{k+1} + {\mathbf{F}^k}^T {\mathbf{R}^k}^{-1}\mathbf{F}^k\right) {\mathbf{R}^k}^{-1}\mathbf{F}^k \tag{35}$$

$$\overline{\eta}^{k+1} = {\mathbf{R}^k}^{-1}\left(\mathbf{f}\left(\hat{\mathbf{x}}^k\right) - \mathbf{F}^k\hat{\mathbf{x}}^k\right) - \left({\mathbf{R}^k}^{-1}\mathbf{F}^k\right)^T \left(\mathbf{\Lambda}^{k+1} + {\mathbf{F}^k}^T {\mathbf{R}^k}^{-1}\mathbf{F}^k\right)\left(\eta^k - \left({\mathbf{R}^k}^{-1}\mathbf{F}^k\right)^T \left(\mathbf{f}\left(\hat{\mathbf{x}}^k\right) - \mathbf{F}^k\hat{\mathbf{x}}^k\right)\right) \tag{36}$$

This process of marginalization results in an information matrix completely dense. Initially, in order to store and work with an sparse information matrix we marginalize the gradients and saved the velocity of each time step. This marginalization, however, did not guarantee an sparse matrix and in general it was not sparse.

3.3.1 Observations

This approach was the last one we were trying and was not fully studied. It would be wise to go back to this algorithm - or its dual, the EKF, even though it is already studied - and implemented in a 1-dimensional camera. This algorithm should work as the snake moves: the firsts pixels will recover the gradients and the others, since the image in those points would be already known, will recover the velocity. In one dimension it should recover with no problem both, the gradients and the velocity. And it would be interesting to know how many pixels are needed in order to start recovering the velocity.

References

[1] Event-based Camera Pose Tracking using a Generative Event Model. Guillermo Gallego, Christian Forster, Elias Mueggler, Davide Scaramuzza. October 2015. ArXiv:1510.01972v1 [cs.CV]

[2] M. A. Skoglund, G. Hendeby, and D. Axehill. Extended Kalman filter modifications based on an optimization view point. In 18th International Conference on Information Fusion , 2015.

[3] David J. Fleet, Yair Weiss. Optical Flow Estimation. Computer Vision, University of Toronto.

[4] Joan Solà. Course on SLAM. Institut de Robòtica i Informàtica Industrial (IRI). February 2016.

[5] Bardow, A.J. Davison, S. Leutenegger. Simultaneous Optical Flow and Intensity Estimation from an Event Camera. P.A. Computer Vision and Pattern Recognition (CVPR), 2016.

[6] Matthew R. Walter, Ryan M. Eustice, John J. Leonard. Exactly Sparse Extended Information Filters for Feature-based SLAM. The International Journal of Robotics Research April 2007 vol. 26 no. 4 335-359.

[7] Thrun, S. and Liu, Y. and Koller, D. and Ng, A.Y. and Ghahramani, Z. and Durrant-Whyte, H.. Simultaneous Localization and Mapping With Sparse Extended Information Filters. International Journal of Robotics Research 2004.

[8] H. Kim, A. Handa, R. Benosman, S.-H. Ieng, and A. J. Davison. Simultaneous Mosaicing and Tracking with an Event Camera. In Proceedings of the British Machine Vision Conference (BMVC), 2014.

4 Glossary

This glossary is a list of the scripts and functions - coded in Matlab - that run the algorithms named before. The description of what the file does is explained in the initial lines of each file.

- Calibration
  - Calibration
- Simulator
  - SimulatorEventCamera
- Event-Based Image Recovering with Known Velocity

  In order to properly use the algorithm to recover the gradients knowing the velocity one must first execute `SimulatorEventCamera.m` and then `LoopSolver2LayerVelKnown.m`
  - ActCovGradient
  - ActEKFCovVelKnown
  - ActGradient
  - ActGradOut
  - ComputeCovariances
  - ComputeErrorsVelKnown
  - ComputeJacobiansVelKnown
  - cov2elli
  - dst
  - idst

  - LoopSolver2LayerVelKnown
  - poisson_solver_function
  - Solver2LayersVelKnown

- Event-Based Image Recovering

  In order to properly use the algorithm to recover the gradients and the velocity one must first execute `SimulatorEventCamera.m` and then `LoopSolver2Layers.m`

  - ActCovGradient
  - ActEKFCov
  - ActGradient
  - ActGradOut
  - ComputeCovariances
  - ComputeErrors
  - ComputeJacobians
  - cov2elli
  - LoopSolver2Layera
  - Solver2Layers

- Information Filter

  In order to properly use the algorithm of the EIF to recover the gradients and the velocity one must first execute `SimulatorEventCamera.m` and then `InformationFilterEvCam.m`

  - InformationFilterEvCam