Neuromorphic Image Reconstruction

Tiago Pereira Torres, João Manuel Xavier and José António Gaspar
Institute for Systems and Robotics (ISR/IST), Instituto Superior Técnico, Universidade de Lisboa, Portugal
Email: tiago.torres@tecnico.ulisboa.pt, {jxavier, jag}@isr.tecnico.ulisboa.pt

***Abstract*—Neuromorphic vision sensors record changes in log-intensity and can achieve lower latency, higher temporal resolution and higher dynamic range than conventional active pixel cameras. Furthermore, the in-sensor differential compression presents the challenge of visualizing the camera's output. As such, we focus on mosaic reconstruction. A video reconstruction method is studied and adapted to build a baseline mosaic. A graph interpretation of the event stream is defined from which a constrained least squares log-intensity reconstruction method is introduced. Both mosaicing approaches are quantitatively evaluated.**

Keywords: Neuromorphic camera, event camera, DVS, mosaicing, image reconstruction, graph signal processing.

I. Introduction

Visual systems play an important role in survivability in the animal world. Higher animals depend heavily on it to sense the environment around them and have developed systems which can operate successfully in both high, low and mixtures of both lighting conditions. As such with the intent of improving current cameras they are a prime subject for study and subsequent hardware reproduction.

Neuromorphic hardware draws inspiration from these systems to develop sensors [1], [2]. There are several documented neuromorphic vision sensors ranging from, asynchronous intensity sensors [3], spatial contrast sensors [4] and velocity sensors [5].

The focus of this work is on dynamic vision sensors (DVSs) which record temporal magnitude changes [6]–[9]. Hybrid approaches that record both magnitude and temporal changes [10]–[12] also exist. In contrast to conventional video cameras which record a sequence of pictures (intensities of all pixels), these sensors detect and transmit asynchronous changes in logarithmic light intensity. These changes are recorded as events which consist of a timestamp, pixel position and polarity of the log-intensity change. Due to the analog continuous measurement and log-intensity acquisition these systems are able to provide 1) lower latency 2) faster updates in areas of the image where intensity changes 3) higher dynamic range, than conventional cameras [6]. And unlike conventional cameras, the asynchronous nature of the acquisition results in image compression at the sensor level. This compression thus presents the inverse problem of finding the absolute intensity of pixels.

We set out to develop a technique for image reconstruction, since proper perception of scene details and uniform features are hard for one to achieve from the output of a temporal contrast camera. An example of the difficulty of perceiving static features is presented in Fig. 1 where a set of events is displayed. Furthermore, an intensity reconstruction can be used as an intermediate step in adapting current computer vision techniques to work with neuromorphic cameras. In the field of image reconstruction, we set out to build a mosaic image of the scene observable by the camera. Since neuromorphic cameras possess a high temporal resolution and good dynamic range, they are prime candidates for fast scene acquisition under with mixed, difficult, lighting conditions. Furthermore, current neuromorphic cameras have low resolution and thus mosaicing is a technique that can improve both the breadth and definition of the observed scene.

Fig. 1. Batch of 10^4 triggered events on a camera pan to the left while observing Lena's image. Only the polarity of the last event of each pixel is shown. On and off events are represented in grey and black respectively. Dataset obtained with a simulator of event cameras [13].

These vision systems are referred here as event cameras since this work focuses on signal processing and not on the possible bio-analogies or specific hardware implementations.

A. Related work

Event cameras have proved useful in several applications and one area in which they have been used profusely is simultaneous localization and mapping (SLAM) [14]–[18]. Some of these SLAM techniques intersect the scope of this work since they either estimate an image as an intrinsic part of the algorithm or one is easy to be produced as a by-product.

Reinbacher et al. [16] propose a panoramic tracking method that also generates a grayscale map of the scene. This map represents the likelihood of an event occurring at a position, which is proportional to the gradient magnitude of the log-intensity. The authors also introduce the concept of segmenting the camera path over the mosaic in order to estimate events.

Rebecq et al. [18] propose a 6 degree of freedom (DOF) tracking and mapping technique that relies on image-to-model alignment for pose tracking and a projective voxel grid from which a point cloud is extracted. The authors also present an image/video reconstruction but point only to references, not detailing how it was achieved from the quantities they estimate.

Two works stand out in image reconstruction from an event camera. In Bardow et al. [19] the authors jointly estimate the velocity field and log-intensity in a spatiotemporal window with an optimization process. Their objective is to obtain a smooth velocity and log-intensity estimate of the scene the camera is observing. The data term used models the occurrence and no occurrence of an event. Furthermore, multiple total variation (TV) regularization terms are added for spatiotemporal optical flow and spatial image intensity. The observed drawbacks of this approach are 1) noticeable image

saturation in some pixels 2) interpolation of log-intensity change provided by an event due to the sliding window approach. 3) lack of statistical modelling of the acquisition process.

In Reinbacher et al. [20] image reconstruction is done in two steps. In the first, events are "added" to an estimated image according to the observation model, i.e. each new event updates the intensity estimate by a multiplicative factor. And in the second step, the main focus is regularization of image intensities across spatiotemporal near events. This approach was thought to be more promising as it 1) included statistical modelling for the sensor observations 2) claimed better results 3) provided a software demo. As such, this work was used as an image reconstruction starting point with the objective of generating a mosaic.

In mosaicing from an event camera the work by Kim et al. [14], [15] stands out. They perform image reconstruction and tracking on the mosaic and in 6 DOF respectively. They resort to a pixel-wise extended Kalman filter (EKF) to estimate image gradients from the event stream, discarding the explicit relation between intensities introduced by an event. The gradient image is integrated to yield a log-intensity image.

Furthermore, in the literature, there is also a lack of quantitative comparisons between the different event camera image and mosaic reconstruction algorithms.

B. Problem Formulation

In the work by Kim et al. [14], [15] the mosaic representation stores gradients. This leads to a large representation since it requires six numbers to represent each pixel, two of which are dedicated to storing the gradient vector and four of which store its variance matrix. Furthermore, the gradient representation does not ensure global accuracy of the generated mosaic as the gradient representation disregards constant factors.

Our goal is thus to produce techniques that resort to the working principle of the camera to estimate directly an intensity or log-intensity spherical mosaic. The first approach we propose is adapting an existing regularized image reconstruction algorithm to mosaic reconstruction. The second approach we propose is based on exploiting the structure between mosaic pixels caused by events.

II. Background

A. Perspective Camera

The perspective camera model is characterized with the help of homogeneous coordinates. A point in $x \in \mathbb{R}^2$ is mapped to the homogeneous coordinate $\tilde{x} \in \mathbb{P}^2$, $\tilde{x} = h(x) = \begin{bmatrix} x_1 & x_2 & 1 \end{bmatrix}^T$ and reversely a point in homogeneous coordinates is transformed to inhomogeneous as $x = h^{-1}(\tilde{x}) = \begin{bmatrix} \tilde{x}_1/\tilde{x}_3 & \tilde{x}_2/\tilde{x}_3 \end{bmatrix}^T$. The perspective camera model takes a point in space $X_C \in \mathbb{R}^3$ in the same reference frame as the camera and projects it to the sensor domain, $x_C \in \Omega_C \subset \mathbb{R}^2$ with $x_C = h^{-1}(KX_C)$, where $K \in \mathbb{R}^{3\times3}$ is the camera calibration matrix.

Camera pose is given as a transformation from camera to world coordinates, $X_W = RX_C + T$ with $R \in \mathrm{SO}(3), T \in \mathbb{R}^3$, rotation matrix and translation respectively. Therefore a point in the world X_W is projected on to the camera sensor with

$$x_C = h^{-1}\left(KR^{-1}\left(X_W - T\right)\right) \tag{1}$$

and back-projection is achieved with

$$X_W = RK^{-1}h\left(x_C\right) + T. \tag{2}$$

B. Working Principle of Event Cameras

Event cameras considered in this work record log-intensity changes in the scene. These changes are encoded as events. An event is a triple (x, θ, t) where $x \in \Omega_C \subset \mathbb{N}_+^2$ is the pixel coordinate in the camera domain, $\theta \in \{-1, 1\}$ is the event polarity that encodes if there was an increase or decrease in log-intensity and $t \in \mathbb{R}_+$ is the time at which the event occurred in camera time. The notation used throughout this work uses a transformation of the camera stream where it includes knowledge of the camera acquisition parameters and introduces a temporal index. Therefore an event in the data stream $\mathcal{D} = \left\{d^{(k)} : k = 1, \ldots, K\right\}$ is represented as $d^{(k)} = \left(x^{(k)}, \delta^{(k)}, t^{(k)}\right)$, where k indexes events such that $k' > k$ if and only if $t^{(k')} > t^{(k)}$ and $\delta^{(k)}$ is log-intensity change calculated from event polarity and camera event trigger parameters at $t^{(k)}$.

Simply put, an event is triggered once the intensity value $\Gamma^{(k)}_{x^{(k)}}$ observed by a pixel $x^{(k)}$ changes enough over time such that

$$\log \Gamma^{(k)}_{x^{(k)}} - \log \Gamma^{(k-\tau)}_{x^{(k-\tau)}} = \delta^{(k)} \tag{3}$$

where $k - \tau = \max\left(k' : k' < k,\ x^{(k')} = x^{(k)}\right)$ is the index of the previous event that fired at pixel $x^{(k)}$. Note that since the trigger condition is on logarithmic intensities, due to the properties of the logarithm, one can interpret it as a temporal contrast condition. When working with log-intensities the trigger condition reduces to

$$\gamma^{(k)}_{x^{(k)}} - \gamma^{(k-\tau)}_{x^{(k-\tau)}} = \delta^{(k)}. \tag{4}$$

Note that instead of sampling log-intensity at a fixed temporal rate, it is continuously measured and events are generated when log-intensity changes by a trigger threshold.

III. Image Reconstruction from Events

In this section are the main aspects and analysis of the image reconstruction proposed by Reinbacher et al. [20]. This approach iteratively estimates the observed camera image with each event (or small batches) therefore creating a video sequence. Fundamentally it is an event accumulator in the camera domain, coupled with a filter. The filtering is posed as an optimization problem whose objective function is a maximum likelihood estimator coupled with TV regularization.

A. Event Update

The event trigger condition (3) can be rewritten as

$$\Gamma^{(k)}_{x^{(k)}} = \Gamma^{(k-\tau)}_{x^{(k)}} . \exp\left(\delta^{(k)}\right) \tag{5}$$

for the k-th event, $d^{(k)} = \left(x^{(k)}, \delta^{(k)}, t^{(k)}\right)$, in the stream. By representing the event trigger condition this way it is possible to interpret it as relating two images of the scene at different times and therefore also as an update function. Proceeding with this thought, one can say the image that is to be updated is a previously known estimate of the intensities observed by the camera $u^{(k-1)}$ and that the application of the update function results in a noisy image observation $f^{(k)}$. In the presence of noise, one can estimate an image in which its effects are suppressed by leveraging knowledge about the noise process. As such the relationship between the estimated image for the

previous event $u^{(k-1)}$ and an updated image $f^{(k)}$ with the k-th event is

$$f_{\tilde{x}}^{(k)} = \begin{cases} u_{\tilde{x}}^{(k-1)} . \exp\left(\delta^{(k)}\right) & , \tilde{x} = x^{(k)} \\ f_{\tilde{x}}^{(k-1)} & , \tilde{x} \neq x^{(k)} \end{cases}, \tag{6}$$

for all $\tilde{x} \in \Omega_C$, and that upon initialization $f^{(0)} = u^{(0)}$. Note that the explicit definition of the update function is different from Reinbacher et al. [20], since the authors do not make clear where intensities of $f_{\tilde{x}}^{(k)}$ come from, for pixels that did not generate an event $\tilde{x} \neq x^{(k)}$. If the case is that pixels that did not trigger an event keep the same intensity the definitions coincide. If the case is that the whole image $u^{(k-1)}$ is meant to be copied to $f^{(k)}$ and pixels that did not trigger an event are updated, the definitions differ. We argue that the first definition is more sensible since it keeps pixel intensities intact throughout subsequent filtering performed to incorporate events in other pixels. Otherwise, pixel intensities of areas without events would drift due to filtering and the information of the previously included events would be lost.

B. Objective Function

The observed image $f^{(k)}$ is assumed to be corrupted by Poisson noise [21]. This, in turn, implies that noise is dependent on image intensity. The maximum likelihood (ML) estimator [20] can be used as basis for a data term of a regularization process that reconstructs conventional images from events.

To maintain smooth areas with sharp transitions the TV regularizer is used on image intensities, as it models natural image statistics where image patches contain small variation but present sharp intensity transitions between them. Furthermore, event timestamp data is introduced in the regularizer with the assumption that temporal close events are triggered by the same structure and hence it is favourable that they have similar intensities. Taking into account both these quantities results in a spatiotemporal regularization.

Gradients $\nabla_{g^{(k)}}$ in this approach are calculated with the metric $g^{(k)}$ of the manifold $S^{(k)}$ defined by

$$\begin{aligned} \varphi : \Omega_C &\to S^{(k)} \in \mathbb{R}^3 \\ x &\mapsto \left[x_1,\ x_2,\ \eta_x^{(k)}\right]^T. \end{aligned} \tag{7}$$

The time since the last event at pixel x is $\eta_x^{(k)}$ and is defined for the k-th event as

$$\eta_x^{(k)} = \begin{cases} 0 & , x = x^{(k)} \\ t^{(k)} - t^{(k-\tau)} & , x \neq x^{(k)} \end{cases}, \tag{8}$$

where $k - \tau = \max\left(k' : k' < k,\ x^{(k')} = x\right)$. The manifold can be interpreted as a way to encode optical flow, which has proved useful in removing artefacts and noise from videos [22]. By formulating the problem on manifold S, the timestamp data influences the estimation whenever it is not uniform across pixels. With this extra information, the TV penalizes small gradients in both intensity and time, while favouring sharp transitions between zones that generated events at different times and have different intensities.

Reconstruction is thus performed by minimizing the sum of the regularizer and the data term

$$u^{(k)} = \arg\min_u \int_{S^{(k)}} \left\|\nabla_{g^{(k)}} u_s\right\|_1 + \lambda\left(u_s - f_s^{(k)} \log u_s\right)\, ds \tag{9}$$

in which λ is a parameter that penalises deviations of the image to be estimated u from the noisy observed image $f^{(k)}$.

C. Discretization

From (9) one writes the continuous formulation for reconstruction in the image space Ω_C as

$$\begin{aligned} u^{(k)} = \arg\min_u \quad & \int_{\Omega_C} \sqrt{G_x^{(k)}}\Big[\left\|\nabla_{g^{(k)}} u_x\right\|_1 + \\ & \lambda\left(u_x - f_x^{(k)} \log u_x\right)\Big]\, dx \\ \text{s.t.} \quad & u_x \in [u_{min}, u_{max}] \end{aligned}. \tag{10}$$

In order to implement this formulation it is necessary to create a discrete model suitable for numeric computation. Therefore the image domain Ω_C is discretized into a $V_C \times U_C$ grid where images are represented as matrices in $\mathbb{R}^{V_C \times U_C}$. In this method, the notation for indexing the discretized image domain Ω_C is i, j instead of vector x. With this the minimization problem (10) becomes

$$\begin{aligned} \min_u \quad & \left\|L_{g^{(k)}} u\right\|_{g^{(k)}} + \lambda \sum_{i,j}\left(u_{ij} - f_{ij}^{(k)} \log u_{ij}\right)\sqrt{G_{ij}^{(k)}} \\ \text{s.t.} \quad & u_{ij} \in [u_{min}, u_{max}] \end{aligned} \tag{11}$$

where $u_{ij}, f_{ij}^{(k)}, G_{ij}^{(k)} \in \mathbb{R}^{V_C \times U_C}$ and the first term is defined with the g-tensor norm for $A \in \mathbb{R}^{V_C \times U_C \times 3}$ as $\|A\|_{g^{(k)}} = \sum_{i,j} \sqrt{G_{ij}^{(k)} \sum_l \left(A_{ijl}\right)^2}$, where $G_{ij}^{(k)} \geq 1$. This norm can be interpreted as the ℓ_1-norm in the first two indexes of A (indexes i, j i.e. the image domain) of the weighted sum by $\sqrt{G_{ij}^{(k)}}$ of the ℓ_2-norm along the third direction of A (index l). When applied as $\left\|L_{g^{(k)}} u\right\|_{g^{(k)}}$ it yields a ℓ_1-norm on the image domain of the weighted gradient norm of each pixel.

D. Iterations for the Discretized Formulation

The optimization problem (11) can be approximated by a primal-dual formulation which may be solved by Algorithm 1 of Chambolle, Pock [23]. It estimates iteratively a dual variable in one space in order to provide a better approximation of R, and then the variable of interest, the primal, in another. As such the primal-dual formulation of the discrete formulation (11) is

$$\begin{aligned} \min_u \max_p \quad & \langle p, L_{g^{(k)}} u\rangle - \|p\|^*_{g^{(k)}} + \\ & \lambda \sum_{i,j}\left(u_{ij} - f_{ij}^{(k)} \log u_{ij}\right)\sqrt{G_{ij}^{(k)}}. \\ \text{s.t.} \quad & u_{ij} \in [u_{min}, u_{max}] \end{aligned} \tag{12}$$

The step updates for (12) are then

$$\begin{cases} u^{(n+1)} = (I + \tau\partial D)^{-1}\left(u^{(n)} - \tau L^*_{g^{(k)}} p^{(n)}\right) \\ p^{(n+1)} = (I + \sigma\partial R^*)^{-1}\Big(p^{(n)} + \\ \qquad\qquad \sigma L_{g^{(k)}}\left[u^{(n+1)} + \theta\big(u^{(n+1)} - u^{(n)}\big)\right]\Big) \end{cases}, \tag{13}$$

with $\theta \in [0, 1]$. Note that these variables are indexed by n as they represent iterates to solve the optimization problem (12) for a fixed $f^{(k)}$. As an optional feature, the manifold image may be filtered since it is used to compute a gradient used in the regularization. Hence, to compute a denoised version of the

time since last event image, it is assumed that spatiotemporal close events have been triggered by the same entities and therefore the modified Rudin-Osher-Fatemi (ROF) model [23] is used.

E. Effects of Formulation Terms in Reconstruction

In this section successive incremental steps are taken to detail the effects of each component of the optimization until the complete formulation (10) presented by Reinbacher et al. is reached. To simulate the intermediate formulations the step iterates of the primal-dual algorithm (13) are used with slight modifications in order to highlight desired features. In the formulations without manifold, the time since last event for every pixel was set to zero, effectively creating a flat manifold. For formulations with just the regularizer, $\lambda = 0$ was set.

1) Integration of Events: A first approach to produce the video sequence is to update the initial estimate with the stream of events by using the camera's firing principle (3) as used in equation (6) and setting $f^{(k)} = u^{(k)}$. Since it does not filter images, the simple event integration leads to very dissimilar pixel values in which it is impossible to perceive what the camera is recording.

2) Regularized Image Reconstruction: In this sub-section, we aim to present the formulation without the manifold. We start by analysing the role of the data term and then the regularizer.

The first approach is to use just the data term D, which is the ML estimator. Since $u_x - f_x^{(k)} \log u_x$ is a convex function of u_x whose minimizer is $u_x = f_x^{(k)}$ we may conclude that the result of only using the data term is equal to one obtained by simple event integration.

To model natural image statistics a total variation regularizer is used, which enforces patches with small gradient and allows sharp transitions between them. To study the regularizer, the formulation (10) is used with $\lambda = 0$. But, it is clear right away that the solution is any u_x constant for all $x \in \Omega_C$. Therefore, to gain insight on the effects of the regularizer the optimization iterates (13) are slightly modified. Since λ is zero to cancel out the data term, the standard way to introduce data from events is broken. As such in the first iterate of (13) the optimization variable u is initialized as $f^{(k)}$ and updated normally in the following iterates. Comparing the results, with the ones from event integration one is able to observe the effects of the regularizer, which filtered the image by significantly attenuating noise and produced an image in which it is already possible to discern what the camera is recording.

Joining the data fidelity term and the regularizer, in the Cartesian plane, produced a smoother and more accurate images when compared to the previous results.

3) Regularized Image Reconstruction on Manifold: In this sub-section, the manifold is introduced in the reconstruction. It is noted that the manifold used in these simulations was derived from event timestamp data in seconds. The authors [20] do not specify the time unit in which the manifold is defined. This should be explicit and a multiplicative factor should be introduced to change the scale of the manifold, as it would allow to modulate the amount with which image gradients are affected by manifold geometry.

By mapping the images u and f onto the manifold, one introduces extra information from the data stream into the reconstruction. Performing the optimization on the manifold can be interpreted as taking images on the image plane Ω_C and stretching them to comply with the height data of map (7). As defined, when a gradient is evaluated in this manifold, it is large between patches that were generated at different time instants and also large between patches that have different intensities. By the same reasoning as in the previous section, in order to simulate this formulation the optimization iterates is slightly modified by initializing u as $f^{(k)}$. The results obtained, when compared to the equivalent test without the manifold present no significant difference.

As tested, when the data term is coupled with the regularizer in the manifold no further qualitative gains are achieved.

IV. Mosaicing with Event Cameras

Mosaicing is an image processing technique that takes the output of one or more cameras and combines them, usually to obtain a larger field of view. By merging successive measurements from the same camera, or multiple cameras with different perspectives, on an area of the mosaic one obtains super-resolution properties.

A. Mosaicing from Reconstructed Video

In order to obtain a baseline method for event mosaic reconstruction one can resort to video frames $u^{(k)}$, i.e. use the video reconstruction algorithm of Sec. III to estimate snapshots of the scene. The pipeline, starts by using a batch of K' events to build a video frame $u^{(k)}$. From there the timestamp of event $\lfloor k - (K'+1)/2 \rceil$ is used to interpolate camera pose from the set of known poses $\mathcal{P}$. This pose is used to project the image on to the mosaic with the framework detailed in Sec. IV-A2.

1) Mosaicing with a Pan-Tilt Camera: The goal is to define how the image observed by the camera sensor is projected on to the mosaic. This mosaic stores the intensity information that is obtained when the camera is panned and tilted over the scene.

In order to store the whole scene and since camera motion is assumed to consist of only rotations a spherical projection is used. The mosaic reference frame is defined to be the same as world coordinates $X_M \triangleq X_W$, such that the forward spherical projection is defined as

$$\begin{cases} \phi & = \tan^{-1}\left(X_{M_1}, X_{M_3}\right) \\ \theta & = \sin^{-1}\left(\frac{X_{M_2}}{\sqrt{X_{M_1}^2 + X_{M_2}^2 + X_{M_3}^2}}\right) \\ x_{M_1} & = f_{M_1}\phi + c_{M_1} \\ x_{M_2} & = f_{M_2}\theta + c_{M_2} \end{cases} \tag{14}$$

where $f_{M_1}, f_{M_2}, c_{M_1}, c_{M_2}$ are projection parameters that determine respectively the size and centre of the mosaic. The geometry of this projection is depicted in Fig. 2. Note that the 2-argument arctangent function is used in order to preserve quadrant information and therefore $\phi \in \,]-\pi, \pi]$, whilst at this stage $\theta \in [-\pi/2, \pi/2]$.

Once again back-projection can not retrieve the exact 3D point as the distance of X_M to O_M is lost in the forward projection (14). In order to invert the underdetermined system of equations (14) the equation $X_{M_3} = \cos(\phi)$ is introduced. This equation respects the geometry of the projection, depicted in Fig. 2, but constrains the point (X_{M_1}, X_{M_3}) to the unit circle

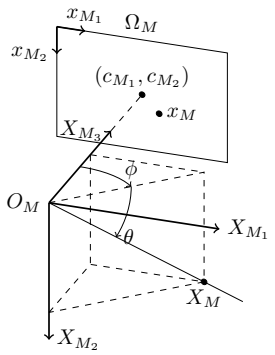


Fig. 2. Spherical projection illustration for a point X_M in the mosaic reference frame. The projection angles (ϕ, θ) are shown for the ray that passes through the origin and X_M. The corresponding projected point x_M is shown in the mosaic domain Ω_M. Note that Ω_M is indexed with an affine transformation of the projection angles and hence a straightforward projection is when $(\phi, \theta) = (0, 0)$ wich yields $x_M = (c_{M_1}, c_{M_2})$. This is why, for illustrative purposes, the mosaic domain is represented in that respective location, but another one could be chosen.

in the $X_{M_1} O_M X_{M_3}$ plane. With some algebraic manipulation, one can now write the reverse projection as

$$\begin{cases} \phi & = \frac{x_{M_1} - c_{M_1}}{f_{M_1}} \\ \theta & = \frac{x_{M_2} - c_{M_2}}{f_{M_2}} \\ X_{M_3} & = \cos(\phi) \\ X_{M_1} & = \sin(\phi) \\ X_{M_2} & = \tan(\theta) \end{cases} . \tag{15}$$

Note that due to the domain of the tangent function now $\theta \in]-\pi/2, \pi/2[$, which means that the poles can not be mapped.

2) Mosaic Discretization: Since the models presented in the previous chapter are for continuous domains, some modifications are needed when they are used with real cameras and computer representations. The camera sensor domain is actually a discrete grid and the mosaic is discretized to be represented in the computer. As such the camera and mosaic domains are respectively, $\Omega_C = [1, V_C] \times [1, U_C]$ and $\Omega_M = [1, V_M] \times [1, U_M]$. Intensities in the mosaic are represented as $\Gamma \in \mathbb{R}^{V_M \times U_M}$ and log-intensities as $\gamma \in \mathbb{R}^{V_M \times U_M}$. The projection parameters are thus defined as $f_{M_1} = \frac{U_M}{2\pi}$, $f_{M_2} = \frac{V_M}{\pi}$, $c_{M_1} = \frac{U_M}{2}$, $c_{M_2} = \frac{V_M}{2}$ in order for the whole scene to be able to be stored in the mosaic. Indexing these domains requires positive natural numbers which can be obtained by rounding to the nearest integer the results of the projection maps (1), (14), but this is insufficient for a complete representation. As soon as the pixel grids are introduced, the one to one correspondence between mosaic and camera domains is broken. Therefore in the discrete setting, all the pixels in a projection patch need to be computed in order to have a full map from one domain to the other.

The set of camera poses associated with the event stream $\mathcal{D}$ is $\mathcal{P} = \{p^{(l)} : l = 1, \ldots, L\}$. A pose is $p^{(l)} = (R^{(l)}, T^{(l)}, s^{(l)})$ where its elements are respectively a rotation matrix, a position and time in which they were recorded. The algorithms detailed in the thesis systematize how the projection map $\Pi(x^{(k)}, t^{(k)}, \mathcal{P})$ projects a camera pixel on to the mosaic can be computed.

B. Mosaicing Directly from Events

The following sections focus on building an algorithm to generate a panoramic image of the scene directly from the event stream. A graph that relates scene points is built based on the event firing principle and knowledge of the camera pose. This graph is then used in a quadratic optimization problem with linear constraints to estimate scene log-intensities.

1) Single Pixel Camera: Consider a single pixel event camera with a known camera pose, e.g. provided by an odometry sensor. When a pan and tilt motion is performed, one can build an oriented path in the scene of intensities observed by the pixel. Furthermore, this path can be segmented by events and is the basis for a graph in which vertices represent scene points that triggered a brightness change and edges denote brightness changes between scene points. The log-intensity observed by the camera is defined as a function on this graph, i.e. each node has an assigned log-intensity.

2) Building the Graph from the Event Stream of a Multipixel Camera: The focus now is on a camera with $V_C \times U_C$ pixels. Due to the underlying mosaic representation, a camera pixel can image multiple pixels of the mosaic. As such, each camera pixel builds a graph like in the single pixel case but vertices are now a set of mosaic pixels. For each camera pixel $x \in \Omega_C$ a time series of sets is created. For $k = 0, \ldots, K$, let the set $\mathcal{A}_x^{(k)}$ denote the most recent (at the time of the k-th event, $t^{(k)}$) mosaic region that projected to camera pixel x and triggered an event. This time series is built recursively for all $x \in \Omega_C$, by initializing $\mathcal{A}_x^{(0)} \leftarrow \Pi(x, 0, \mathcal{P})$, where Π is the projection map. And updating with the k-th event

$$\begin{cases} \mathcal{A}_x^{(k)} \leftarrow \Pi\left(x^{(k)}, t^{(k)}, \mathcal{P}\right) & , x = x^{(k)} \\ \mathcal{A}_x^{(k)} \leftarrow \mathcal{A}_x^{(k-1)} & , x \neq x^{(k)} \end{cases} . \tag{16}$$

Note that the projection region of a camera pixel only changes when an event is triggered. To make the notation more clearly distinguish between the two mosaic sets that triggered an event the set $\mathcal{B}_x^{(k)} \triangleq \mathcal{A}_x^{(k-1)}$ is introduced.

The graph mentioned in the single pixel camera case, can be defined as the directed weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ for the multipixel event camera. Its vertex set is the union of all the projection sets without repetitions

$$\mathcal{V} = \text{unique}\left(\left\{\mathcal{A}_{x^{(k)}}^{(k)}, \mathcal{B}_{x^{(k)}}^{(k)} \text{ for } k = 1, \ldots, K\right\}\right). \tag{17}$$

Edges are defined between sets generated by an event, excluding self-loops,

$$\mathcal{E} = \Big\{ (v', v) : v' = \mathcal{A}_{x^{(k)}}^{(k)},\ v = \mathcal{B}_{x^{(k)}}^{(k)}, \\ \mathcal{A}_{x^{(k)}}^{(k)} \neq \mathcal{B}_{x^{(k)}}^{(k)},\ k = 1, \ldots, K \Big\} \tag{18}$$

and edge weights are event log-intensity change

$$\mathcal{W} = \Big\{ w((v', v)) : w\left(\left(\mathcal{A}_{x^{(k)}}^{(k)}, \mathcal{B}_{x^{(k)}}^{(k)}\right)\right) = \delta^{(k)}, \\ \mathcal{A}_{x^{(k)}}^{(k)} \neq \mathcal{B}_{x^{(k)}}^{(k)},\ k = 1, \ldots, K \Big\}. \tag{19}$$

Edges are kept as ordered pairs of vertices in order to preserve edge direction. This is crucial as without direction the weight could not define unequivocally a change in log-intensity between node values. Self-loops are not allowed in (18), (19) as edge weights are different from zero and if it were to exist a self-loop in the graph it would imply that a set of mosaic pixel log-intensities differs from itself a non-zero amount, which is impossible.

The objective is now devising a technique to compute log-intensities for all pixels in the vertices, i.e. computing the function whose argument is a pixel and returns a log-intensity.

3) Event Constraints: The graph built with the events is now used to compute constraints on the log-intensity, where each constraint is a modified version of the event trigger condition (4). This modification is made to allow comparisons between two sets of pixels instead of single pixels. As such the event trigger condition is,

$$\mathcal{F}(\gamma_{v'}) - \mathcal{F}(\gamma_v) = w((v',v)), \quad \forall (v',v) \in \mathcal{E} \tag{20}$$

where $\mathcal{F}$ is the mixture model and γ_v are the the mosaic log-intensities for each pixel in the set v. In this work two simplified mixture models are considered, the first,

$$\frac{1}{|v'|}\sum_{x\in v'}\gamma_x - \frac{1}{|v|}\sum_{x\in v}\gamma_x = w((v',v)), \quad \forall (v',v) \in \mathcal{E} \tag{21}$$

computes the mean of each set and uses them as the log-intensities that triggered the event, i.e. gives an uniform weight for pixels in a set. The second,

$$\sum_{x\in v'}\gamma_x - \sum_{x\in v}\gamma_x = w((v',v)), \quad \forall (v',v) \in \mathcal{E} \tag{22}$$

sums all the intensities for each set, effectively giving an uniform weight for all pixels in both sets.

4) Quadratic Log-Intensity Reconstruction: Given the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ built from the stream of events and an initial guess on mosaic log-intensities $\bar{\gamma}$, the proposed method minimizes the sum of squared errors between the guess and a solution that respects event constraints. Since we seek to estimate individual pixels, let $\mathcal{V}' = \{x : x \in v, v \in \mathcal{V}\}$ denote the set of all the pixels whose log-intensity is to be estimated. The problem is thus formalized as

$$\begin{aligned} &\underset{\{\gamma_x : x\in\mathcal{V}'\}}{\arg\min} \sum_{x\in\mathcal{V}'} (\bar{\gamma}_x - \gamma_x)^2 \\ &\text{s.t.} \frac{1}{|v'|}\sum_{x\in v'}\gamma_x - \frac{1}{|v|}\sum_{x\in v}\gamma_x = w((v',v)), \forall (v',v) \in \mathcal{E} \end{aligned} \tag{23}$$

for the mean mixture of log-intensities, and as

$$\begin{aligned} &\underset{\{\gamma_x : x\in\mathcal{V}'\}}{\arg\min} \sum_{x\in\mathcal{V}'} (\bar{\gamma}_x - \gamma_x)^2 \\ &\text{s.t.} \sum_{x\in v'}\gamma_x - \sum_{x\in v}\gamma_x = w((v',v)), \ \forall (v',v) \in \mathcal{E} \end{aligned} \tag{24}$$

for the uniform mixture of log-intensities. Note that the initial guess $\bar{\gamma}$ can be the original mosaic with some noise added, only a part of the mosaic or a simple initialization with a uniform mosaic.

Note that the reconstruction is performed for pixels in $\mathcal{V}'$, which are mosaic pixels between which an event was triggered. As such this reconstruction is done for a subset of mosaic pixels. It does not provide mosaic inpainting as usually happens with total variation functionals.

C. Implementation of Event-Based Mosacing

In this section is detailed a technique to solve the aforementioned optimization problem, some implementation aspects and limits.

1) Minimization Method: In order to compute a simple solution for the problems, (23) and (24), they are rewritten in a manner that the optimization variable is a vector instead of a 2-D matrix. This takes the form of

$$\begin{aligned} \underset{z}{\arg\min} \quad & z^T A z + b^T z \\ \text{s.t.} \quad & Cz = d \end{aligned}, \tag{25}$$

where $z \in \mathbb{R}^{|\mathcal{V}'|}$ is the log-intensity of the pixels involved in the constraints generated by the events. To match the pixel coordinates in the set $\mathcal{V}'$ and its corresponding optimization variable in z an invertible proxy function $\mathcal{H} : \mathcal{V}' \to \{1, \ldots, |\mathcal{V}'|\}$ is introduced. Matrices $A \in \mathbb{R}^{|\mathcal{V}'|\times|\mathcal{V}'|}, b \in \mathbb{R}^{|\mathcal{V}'|}$ can be defined such that for each $x \in \mathcal{V}'$

$$[A]_{ij} = \begin{cases} 1 & , i = j = \mathcal{H}(x) \\ 0 & , \text{otherwise} \end{cases}, b_{\mathcal{H}(x)} = -2\bar{\gamma}_x, \tag{26}$$

where it is noted that A is the identity matrix. In order to construct the constraints an edge labelling $\mathcal{L} : \mathcal{E} \to \{1, \ldots, |\mathcal{E}|\}$ is used, such that for an edge $(v', v) \in \mathcal{E}$

$$\begin{aligned} [C]_{\mathcal{L}(v',v),j} &= \begin{cases} \frac{1}{|v'|} - \frac{1}{|v|} & , j \in \{\mathcal{H}(x) : x \in v' \cap v\} \\ \frac{1}{|v'|} & , j \in \{\mathcal{H}(x) : x \in v' \setminus v\} \\ -\frac{1}{|v|} & , j \in \{\mathcal{H}(x) : x \in v \setminus v'\} \\ 0 & , \text{otherwise} \end{cases}, \\ d_{\mathcal{L}(v',v)} &= w((v',v)) \end{aligned} \tag{27}$$

for the mean mixture of log-intensities and

$$\begin{aligned} [C]_{\mathcal{L}(v',v),j} &= \begin{cases} 1 & , j \in \{\mathcal{H}(x) : x \in v' \setminus v\} \\ -1 & , j \in \{\mathcal{H}(x) : x \in v \setminus v'\} \\ 0 & , \text{otherwise} \end{cases}, \\ d_{\mathcal{L}(v',v)} &= w((v',v)) \end{aligned} \tag{28}$$

for the uniform mixture of log-intensities where in both cases $C \in \mathbb{R}^{|\mathcal{E}|\times|\mathcal{V}'|}, d \in \mathbb{R}^{|\mathcal{E}|}$. In the mean mixture of log-intensities (27) whenever both mosaic projection sets v', v intersect and have the same cardinality, the pixels of the intersection will not be constrained in the optimization. The same also happens for the uniform mixture of log-intensities (28) but independently of set cardinalities. Both these behaviours although not ideal are a consequence of the mixture models (21), (22) used since the variables in the intersection of v', v cancel out.

With the Karush-Kuhn-Tucker conditions [24] it is possible to compute a system of equations for the minimizer

$$\begin{bmatrix} A + A^T & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} z \\ \lambda \end{bmatrix} = \begin{bmatrix} -b \\ d \end{bmatrix}. \tag{29}$$

This system may be solved directly or by block elimination with Schur complements [24] in order to reduce computational complexity. This technique is applicable whenever $A + A^T$ is invertible which in this case is always since $A + A^T = 2I$. The algorithm is presented in Alg. 1.

Algorithm 1 Steps to compute the mosaic reconstruction with a batch of events

procedure EVENT BATCH UPDATE(b, C, d)
  Compute $S = -\frac{1}{2}CC^T$ and $\tilde{b} = d + \frac{1}{2}Cb$
  Find λ by solving $S\lambda = \tilde{b}$
  Find z by computing $z = \frac{1}{2}\left(-b - C^T\lambda\right)$
end procedure

2) Specific Aspects: The success of the algorithm is dependent on matrix S in Alg. 1 being invertible, which is not always the case.

In Fig. 3 are shown the effects of changing parameters in this optimization problem. The first effect, is how the number of edges $|\mathcal{E}|$ changes with mosaic resolution. Note that the plots of $|\mathcal{E}|$ are mostly hidden behind the yellow curves for $\text{s-rank}\,(C)$. In Figs. 3a, 3c one can see (the deviation from a unit slope line for batch sizes less than 10^2) that the number of edges $|\mathcal{E}'|$ is not equal to the number of events in the batch since there are discarded events. This is due to insufficient mosaic resolution that for small camera movements produces equal projection sets $\mathcal{A}^{(k)}_{x^{(k)}} = \mathcal{B}^{(k)}_{x^{(k)}}$.

The second is how the number of referenced mosaic pixels $|\mathcal{V}'|$ is increased when the mosaic pixel count quadruples.

The third is the effect of mosaic pixel mixture functions on the invertibility of matrix S, which is crucial in Alg. 1. Here, matrices S_1, S_2 are computed respectively from matrices (27), (28) as specified in Alg. 1. To get a sense of the invertibility of S_1, S_2 a lower bound of the 1-norm condition number [25], [26], $\hat{\kappa}$ is used. Note that since it is a lower bound, S_1, S_2 can be non-invertible and $\hat{\kappa}$ finite. Although both mixture functions appear to produce similar condition numbers, the mean mixture of log-intensities did provide a more stable point in which its condition number $\hat{\kappa}\,(S_1)$ goes to infinity with respect to the batch size.

In fourth, for batch sizes around 10^4 and above the slope of the number of referenced mosaic pixels $|\mathcal{V}'|$ starts to taper off. This happens independently of mosaic resolution. An explanation for this might be that as batch size increases more complex relationships are added to the graph making it more connected, instead adding new nodes with simple relations. As such the number of variables $|\mathcal{V}'|$ starts to tend toward the number of constraints $|\mathcal{E}|$. On the other hand, problems from camera pose uncertainty (and subsequent accumulation of projection errors) and errors in camera pixel triggering might lead to a malformed graph. Note that by the time the linear growth of $|\mathcal{V}'|$ is broken, S can not be successfully inverted (see lots of $\hat{\kappa}$). Resorting to the structural rank [27], which is the maximum rank a sparse matrix can achieve with its pattern of zero entries, reveals that for the most part when $|\mathcal{E}| < |\mathcal{V}'|$, $\text{s-rank}\,(C) = |\mathcal{E}|$ and $\text{s-rank}\,(S) = |\mathcal{E}|$. This is a good indication, but since the structural rank $\text{s-rank}\,(S) \geq \text{rank}\,(S)$ no conclusive remark can be made about the invertibility of matrix S with this metric when $\text{s-rank}\,(S) = |\mathcal{E}|$. When $\text{s-rank}\,(C) < |\mathcal{E}|$, which happens for data-points above batch sizes of 10^4, matrix S is not invertible.

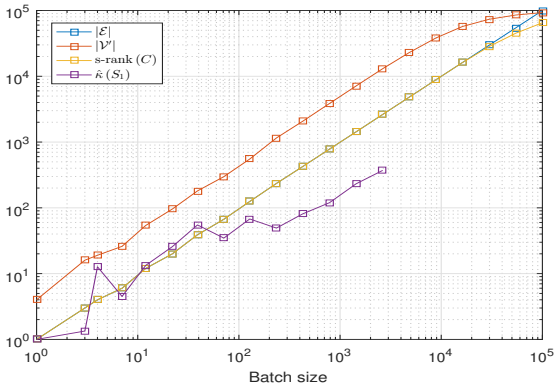
In fifth, it is noted that both $|\mathcal{E}|, |\mathcal{V}'|$ have the same values for different mixture models as they are properties of the graph.

Instead of using the complete dataset all at once, disjoint subsets of the event stream of K' events are used to build the graph and run the optimization problem.

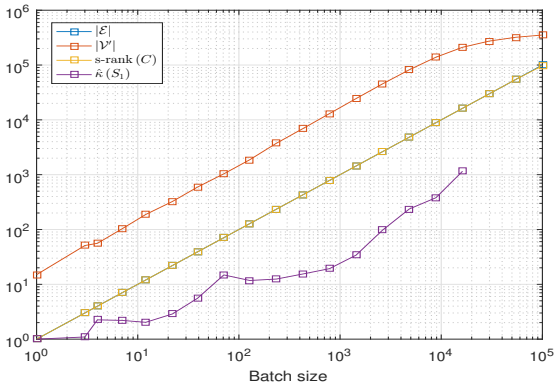
In order for tests to run within reasonable time and with stable outputs the chosen parameters to run this algorithm were $(V_M, U_M) = (1024, 2048)$ and a batch size of 10^3.

V. Mosaicing Experiments and Results

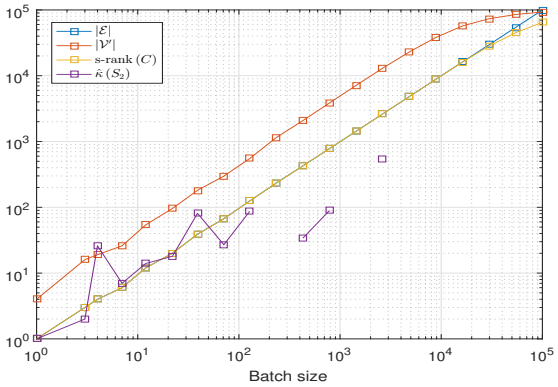
In this chapter are presented experimental results from the proposed mosaicing methods and metrics to objectively compare their results. The focus of the evaluation is on the quality of the reconstructions relative to the ground truth and execution times.



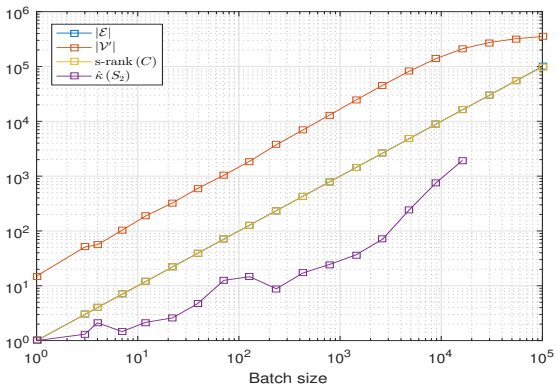
(a) Mean pixel mixture model, $(V_M, U_M) = (1024, 2048)$.



(b) Mean pixel mixture model, $(V_M, U_M) = (2048, 4096)$.



(c) Uniform pixel mixture model, $(V_M, U_M) = (1024, 2048)$.



(d) Uniform pixel mixture model, $(V_M, U_M) = (2048, 4096)$.

Fig. 3. Effects of batch size in the number of graph edges created, number of mosaic pixels calculated and lower bound on the 1-norm condition number of the sparse matrices S. Plots are shown for two different mosaic pixel mixture models and two mosaic resolutions. Points with infinite value are not plotted.

Fig. 4. Ground truth intensity mosaic observed by the camera.

Fig. 5. Intensity mosaic reconstruction of method 1.

A. Dataset

The dataset used in the experiments and the objective comparisons is a dataset that covers part of a scene. It consists of a camera rotation, along its optical axis, to the right by ninety degrees and a left pan that traverses the scene two and a quarter times at slightly different heights. The ground truth intensity mosaic observable by the camera in this dataset is presented in Fig. 4.

B. Mosaic Reconstruction Method 1

In this section are presented the results of mosaicing from reconstructed video, as detailed in Sec. IV-A.

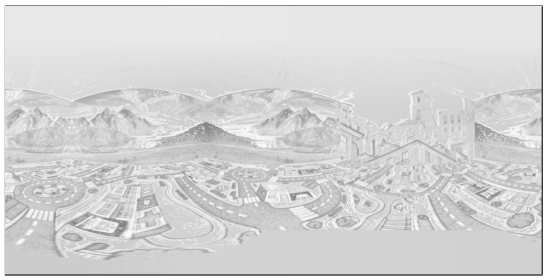
The reconstruction parameters for method 1 are adjusted as to (i) speed up processing by using K' events per batch (ii) match the camera's trigger thresholds, by setting δ^+ and δ^- (iii) adequate image persistence to camera movement speed by setting λ (iv) output an image in the $[0, 1]$ range for posterior comparison with the ground truth by adjusting u_{min} and u_{max}. The values of the parameters used are presented in Tab. I, and

TABLE I
PARAMETERS USED IN THE MOSAIC RECONSTRUCTION OF METHOD 1.

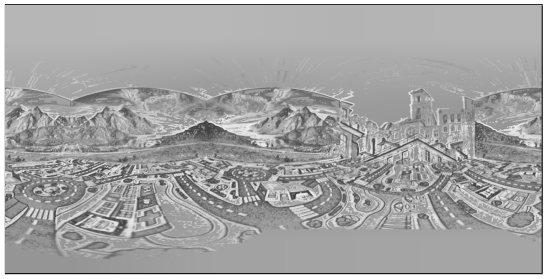
λ	δ^+	δ^-	u_{min}	u_{max}	K'
500	0.2	-0.2	2.2204×10^{-16}	1	10^4
	$PDmaxIter$	τ	σ	θ	
	50	$\frac{1}{8+4\sqrt{2}}$	1	1	

the resulting mosaic is presented in Fig. 5.

This mosaicing method provides a reconstruction that enables a fair perception of the scene. But it leaves some streaking on the trailing edge of the image observed by the camera, with which it is possible to discern camera direction in the last pass over an area (most visible in the top-right). This is due to small blemishes present in the reconstruction (visible in the top-left) coupled with the persistence of projected images in the mosaic domain. This technique also introduces pasting artefacts (visible in the top-centre), which are likely due to the limited area that is being accounted for in the reconstruction, i.e. the camera domain, resulting in a local reconstruction that is not consistent in the whole mosaic.



(a) Mean mixture of log-intensities.



(b) Uniform mixture of log-intensities.

Fig. 6. Log-intensity mosaic reconstruction of method 2.

C. Mosaic Reconstruction Method 2

In this section are presented the results of mosaicing directly from events, as introduced in Sec. IV-B. The output of method 2 is shown in Fig. 6 for the mean mixture of log-intensities (21) and the uniform mixture (22) functions. Mosaic initialization was naïve (mosaic pixel values set to zero) for both and the reconstruction parameters are $\delta^+ = 0.2,\ \delta^- = -0.2,\ K' = 10^3$.

The reconstruction from the mean mixture of log-intensities is smoother and visually more similar to the ground truth than the one with the uniform mixture. This difference is most notable in the mosaic centre in the mountains and roundabout.

D. Mosaic Reconstruction Quality

Since method 2 builds a non-dense mosaic image the metrics used to evaluate its reconstruction and the one obtained with method 1 will take into account only the pixels computed in the former. Therefore metrics that rely on dense local pixel structure, e.g. structural similarity (SSIM), to evaluate performance will not be used. Furthermore, the mosaic images from method 2 are exponentiated to yield intensity mosaics.

1) Affine Transformation of Intensities: Since ground truth and reconstructed images come from different sources they possess varying offsets and scales.

As such image intensities are subjected to an affine transformation before the metrics presented in the following sections are computed. Since there are multiple offsets in the mosaics, each mosaic has its mean subtracted in order to have intensities centred about 0. And although event trigger parameters are the same, the methods provided mosaics with varying intensity spreads, therefore mosaic images were scaled to have unit variance. This normalization of image intensities is computed for the last keyframe of each reconstruction and used to scale all the previous keyframes in order to minimize variability before computing metrics. Formally, for the mosaic of each reconstruction technique μ, σ are computed as

$$\mu = \frac{\sum_{x \in \mathcal{V}'} \Gamma_x}{|\mathcal{V}'|}, \quad \sigma^2 = \frac{\sum_{x \in \mathcal{V}'} (\Gamma_x - \mu)^2}{|\mathcal{V}'| - 1} \tag{30}$$

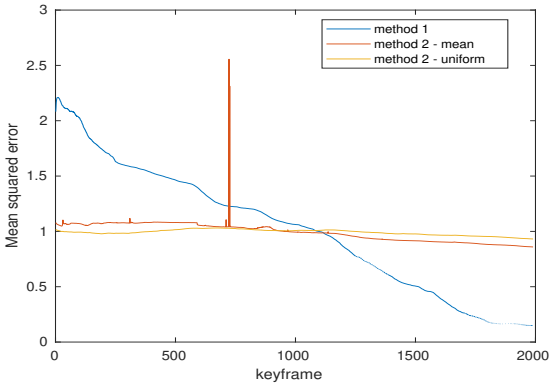


Fig. 7. MSE metric for methods 1 and 2.

for the last keyframe and the transformation $B_x = (\Gamma_x - \mu)/\sigma,\ x \in \Omega_M$ is applied to all keyframes. To obtain the normalized ground truth A the same equations are used, but note that it is a single image of the observable scene and therefore all reconstruction methods are compared to the final mosaic image, shown in Fig. 4.

2) Metric: In order to measure deviation from the reference mosaic three metrics are used. The first is the mean squared error (MSE),

$$\mathrm{MSE}\,(A,B) = \sum_{x \in \mathcal{V}'} \frac{(A_x - B_x)^2}{|\mathcal{V}'|} \tag{31}$$

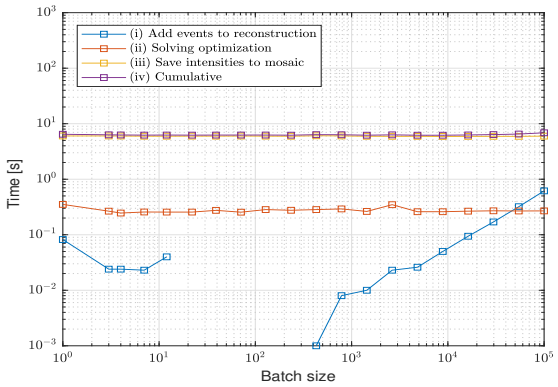
where $\mathcal{V}'$ is the subset of computed mosaic pixels and A, B are respectively the ground truth and the reconstruction.

The metric is presented in Fig. 7 and it was calculated at the same dataset pose/time as the keyframes from the simulator, which corresponds to a sample period of 1 ms. Note that the curves of the metrics for method 1 show a dotted pattern instead of continuous. This is due to the event batch size being constant and in those sections of the dataset less events per time unit are generated, it thus happens that a batch size might include multiple keyframes.

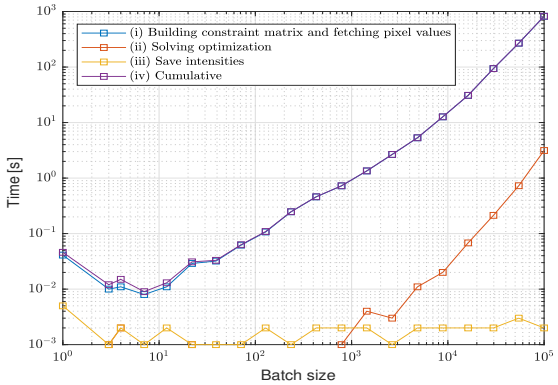
In terms of MSE, Fig. 7, method 1 presents itself as the best method. Method 2 with mean mixture function comes second with the best reconstruction, but around the 750th keyframe there is a huge spike in the MSE as the reconstruction produced a mosaic with extreme mosaic intensities. This is an inherent feature of this mixture function. Furthermore, this metric presents a problem as it has minimal variation for the reconstructions of method 2, even though they go from zero to semi-dense mosaics where a fair perception of the scene is possible. Simply by looking at the metric one would assume that the final mosaic is very similar to the initialization. Once again, note that these metrics are only evaluating intensities that were estimated by both method 1 and 2.

E. Execution Time

Due to the large amounts of events that event cameras can generate it is important to have fast methods to process data. In Fig. 8 are presented wall-clock execution times for different batch sizes using the MATLAB implementations. For method 1 the legend entries refer to (i) the time it takes to build image f with equation (6), (ii) how long the primal-dual iterates of equation (13) take, (iii) how much time image projection on to the mosaic takes, (iv) sum of the previous. The legend for method 2 refers to (i) the time it takes to build matrices b, C, d for which it is required to find pixel projections on the



(a) Method 1.



(b) Method 2.

Fig. 8. Breakdown of batch processing wall-clock times by method and operation performed.

mosaic and retrieving their intensities, (ii) how long it takes to compute intensities with Alg. 1, (iii) the time it takes to write the reconstruction back on to the mosaic, (iv) sum of the previous. The temporal breakdown for method 2 does not detail the usage of a specific mixture function as there is no significant difference between the ones introduced.

In the execution-time breakdown for method 1, the time to process a batch is relatively constant and the majority of time taken to process a batch is devoted to finding pixel projections. For batch sizes above 10^3 the time to include events in the reconstruction increases linearly with batch size, where it starts to take on a more significant portion of the batch processing time. For method 2 the execution-time increases linearly between batch sizes of 10 and 10^4, and as the first method, the majority of time is dedicated to projections. Batch sizes of 10^4 or more have a considerable penalty as it takes an order of magnitude more to process an event, which is likely due to the size of memory operations. For batch sizes of up to 10^4 method 2 is faster as it only needs to compute one projection for each event, whilst method 1, independently of batch size, computes a projection for the whole camera frame, which totals $240 \times 180 = 43200$ projections. Although it is possible to add large numbers of events per batch in method 1, the reconstruction quality is reduced as the number of events is increased and eventually the optimization step can not filter properly the image. In contrast method 2 fares the best when more events per batch are used, up to the point where the reconstruction becomes infeasible, as discussed in Sec. IV-C2.

VI. Conclusion and Future Work

In this section the work is discussed, the contributions made are highlighted and hints for future work are proposed.

A. Discussion

The video reconstruction presented in Sec. III proved qualitatively successful. But the reconstruction quality is extremely dependent on multiple parameters, leading to a high dimensionality search space in which fine tuning is needed to provide the best results for different environments. Furthermore the added complexity of performing the reconstruction on a manifold did not provide significant gains in qualitative reconstruction quality.

The first method presented in Sec. IV shows how a video reconstruction algorithm can easily be adapted to build a mosaic, even though some limitations of the camera domain reconstruction are present.

The graph interpretation of the event stream in Sec. IV presents is conceptually sound, but it could fall apart when 1) pixel information is lost 2) substantial noise is present in either poses or pixel event triggering. Any one or the combination of both can make the graph an incoherent representation of the observed intensities.

Two linear pixel log-intensity mixture models were proposed for method 2 since they are simple and allow it to be a quadratic program. But these mixture models are not guaranteed to be the best way to compute an overall log-intensity from a set of pixels and vice versa, as used in this work.

Some difficulties were encountered in Sec. V as mosaics with a dense and semi-dense reconstruction were to be compared. Based on the used metrics, method 1 fared better than method 2, but for a human interpretation of the scene one can argue that method 2 is also a viable alternative as its non-dense estimation allows a good perception of the scene.

B. Contributions

The contributions of this work are the interpretation and formalization of the event stream as a graph and proposal of a fast and simple reconstruction algorithm. Furthermore the equations of Reinbacher et al. [20] were detailed, components of their algorithm were studied and it was used as a tool to build a benchmark mosaic. Finally metrics were proposed and an evaluation of performance against the ground truth was performed.

C. Future Work

By simplification of the projection subsystem, the method proposed in Sec. IV may be convenient for image or video reconstruction in the camera plane/grid, i.e. without the mosaic. Graph interpretations of the event stream that lead to other graph representations would also be interesting to pursue.

ACKNOWLEDGEMENTS

We thank Engineer Nuno Barroso Monteiro for the help with the state of the art research, computational tools and datasets used in this work.

REFERENCES

[1] C. A. Mead and M. Mahowald, "A silicon model of early visual processing," *Neural Networks*, vol. 1, no. 1, pp. 91 – 97, 1988.

[2] K. Boahen, "Neuromorphic microchips," *Scientific American*, vol. 292, no. 5, pp. 56–63, 2005. [Online]. Available: http://www.jstor.org/stable/26060995

[3] C. Shoushun and A. Bermak, "Arbitrated time-to-first spike CMOS image sensor with on-chip histogram equalization," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 3, pp. 346–357, 3 2007.

[4] J. A. Leñero-Bardallo, T. Serrano-Gotarredona, and B. Linares-Barranco, "A five-decade dynamic-range ambient-light-independent calibrated signed-spatial-contrast AER retina with 0.1-ms latency and optional time-to-first-spike mode," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 57, no. 10, pp. 2632–2643, 10 2010.

[5] C. M. Higgins and S. A. Shams, "A biologically inspired modular VLSI system for visual measurement of self-motion," *IEEE Sensors Journal*, vol. 2, no. 6, pp. 508–528, 12 2002.

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

[7] T. Delbruck and R. Berner, "Temporal contrast AER pixel with 0.3%-contrast event threshold," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, 5 2010, pp. 2442–2445.

[8] J. A. Leñero-Bardallo, T. Serrano-Gotarredona, and B. Linares-Barranco, "A 3.6μs latency asynchronous frame-free event-driven dynamic-vision-sensor," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 6, pp. 1443–1455, 6 2011.

[9] T. Serrano-Gotarredona and B. Linares-Barranco, "A 128×128 1.5% contrast sensitivity 0.9% FPN 3μs latency 4mW asynchronous frame-free dynamic vision sensor using transimpedance preamplifiers," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 3, pp. 827–838, 3 2013.

[10] C. Posch, D. Matolin, and R. Wohlgenannt, "A QVGA 143dB dynamic range frame-free PWM image sensor with lossless pixel-level video compression and time-domain CDS," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 1, pp. 259–275, 1 2011.

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

[12] C. Li, C. Brandli, R. Berner, H. Liu, M. Yang, S. C. Liu, and T. Delbruck, "Design of an RGBW color VGA rolling and global shutter dynamic and active-pixel vision sensor," in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, 5 2015, pp. 718–721.

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

[14] H. Kim, A. Handa, R. Benosman, S.-H. Ieng, and A. Davison, "Simultaneous mosaicing and tracking with an event camera," in *Proceedings of the British Machine Vision Conference*. BMVA Press, 2014.

[15] H. Kim, S. Leutenegger, and A. J. Davison, "Real-time 3D reconstruction and 6-DoF tracking with an event camera," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 349–364.

[16] C. Reinbacher, G. Munda, and T. Pock, "Real-time panoramic tracking for event cameras," in *2017 IEEE International Conference on Computational Photography (ICCP)*, 5 2017, pp. 1–9.

[17] H. Rebecq, G. Gallego, and D. Scaramuzza, "EMVS: event-based multi-view stereo," Tech. Rep., 2016. [Online]. Available: http://infoscience.epfl.ch/record/221504

[18] H. Rebecq, T. Horstschaefer, G. Gallego, and D. Scaramuzza, "EVO: a geometric approach to event-based 6-DOF parallel tracking and mapping in real time," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 593–600, 4 2017.

[19] P. Bardow, A. J. Davison, and S. Leutenegger, "Simultaneous optical flow and intensity estimation from an event camera," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 6 2016, pp. 884–892.

[20] C. Reinbacher, G. Graber, and T. Pock, "Real-time intensity-image reconstruction for event cameras using manifold regularisation," in *2016 British Machine Vision Conference (BMVC)*, 2016.

[21] F. Alter, Y. Matsushita, and X. Tang, "An intensity similarity measure in low-light conditions," in *Computer Vision – ECCV 2006*, A. Leonardis, H. Bischof, and A. Pinz, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 267–280.

[22] R. Szeliski, *Computer Vision - Algorithms and Applications*, ser. Texts in Computer Science. Springer, 2011.

[23] A. Chambolle and T. Pock, "A first-order primal-dual algorithm for convex problems withapplications to imaging," *Journal of Mathematical Imaging and Vision*, vol. 40, no. 1, pp. 120–145, 5 2011.

[24] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004. [Online]. Available: http://web.stanford.edu/~boyd/cvxbook/

[25] W. Hager, "Condition estimates," *SIAM Journal on Scientific and Statistical Computing*, vol. 5, no. 2, pp. 311–316, 1984.

[26] N. Higham and F. Tisseur, "A block algorithm for matrix 1-norm estimation, with an application to 1-norm pseudospectra," *SIAM Journal on Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1185–1201, 2000.

[27] K. J. Reinschke, *Multivariable Control A Graph-theoretic Approach*, ser. Lecture notes in Control and Information Sciences. Springer-Verlag, 1988.