Learning Motion Segmentation and Estimation With Event Cameras

Anton Mitrokhin*, Chengxi Ye*, Cornelia Fermüller, Yiannis Aloimonos, Tobi Delbruck, Brianna Cash

Abstract

We present a deep learning approach to the problem of dynamic scene recovery from event data, which in addition to camera pose and depth estimation segments moving objects and estimates their 3D rigid motion. Our approach uses both unsupervised and supervised components in a multi-level architecture, and takes advantage of the idiosyncratic structure of the data. Additionally, we have created the first event-based dataset for independent motion detection – featuring both real and a synthetic recordings of indoor sequences, with accurate ground truth depth and motion and object masks. Experiments show that our method achieves high accuracy in depth estimation and object segmentation, and good performance on camera and object pose estimation.

1. Introduction

The estimation of 3D motion and scene geometry has been of great interest in Computer Vision and Robotics for its many applications in various areas, such as mobile robotics, autonomous driving, augmented reality and video processing. Most works consider the scene to be static, in which case the problem involves estimating the camera's 3D rigid motion and the scene depth. However, in the general case, the scene may contain independently moving objects, which also need to be segmented and their 3D motion estimated. Earlier work studied the problem in the so-called Structure from Motion (SfM) pipeline [14]. That is,"scene independent" constraints were developed (e.g. the epipolar constraint and depth positivity constraint) to first estimate 3D motion from image measurements, and the estimates are subsequently used for scene reconstruction. In recent years, most works have adopted the SLAM paradigm [10], where depth, 3D motion and image measurements are estimated together using iterative probabilistic approaches, usually initialized with SfM estimates. Most recently learning based approaches (e.g. [36]), have attracted the attention of researchers interested in visual geometry. However, while the problem of detecting moving objects has been studied both in the model-based and learning-based

*The authors contribute equally to this work.

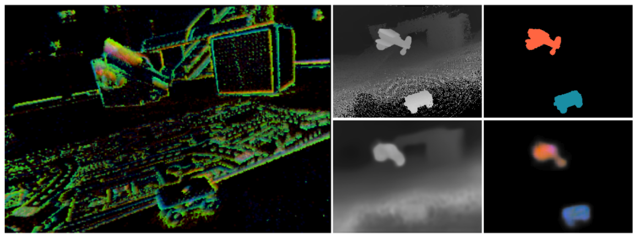

Figure 1. *Depth and per-pixel pose inference on sparse event data, on our EV-IMO dataset. The left image is the event camera output. The top row is the ground truth depth and pose (the color corresponds to linear velocity), the bottom row is network output. Camera pose is also estimated but not visualized. Best viewed in color.*

formulation, estimating object motion in addition to spatio-temporal scene reconstruction is still unexplored. An exception is the work in [28], which however does not provide an evaluation. In this paper we propose a learning-based solution to this problem using as input event-based data.

Event-based processing has long been of interest to computational neuroscientists, but recently a new type of imaging device, known as silicon retina, developed in the neuromorphic community is gaining the interest of Computer Vision and Robotics researchers. Specific devices are the original dynamic vision sensor (DVS), and improved sensors combining the DVS concept with static images and IMU sensors, such as the DAVIS, ATIS, and Samsung's DVS. The dynamic vision sensor does not record image frames, but asynchronous temporal changes in the scene, in form of a continuous stream of events, each of which is generated when a pixel detects a change in log light intensity. The unique properties of these readings, namely high dynamic range, high temporal resolution, low latency and low bandwidth appear very promising features for motion processing in challenging scenarios. However, software development is lagging behind and applications in which the sensor can have a large impact, still need to be demonstrated.

Independent motion detection and estimation seem ideal applications for DVS. In comparison to video, the data from this sensor, the event clouds, are much sparser than sequences of images. Conceptually, event clouds encode the cues of motion and contour. This is a good fit for the problem of motion segmentation, which requires "image motion" information to encode spatio-temporal geometry, and some "static" information for grouping the moving objects. Thus, it appears that DVS provides the minimal information necessary for motion segmentation, but with low latency and high dynamic range. DVS, thus, could be a real asset in solving he problem of motion segmentation under even challenging conditions.

The data we obtain from the DVS is a continuous stream of events. Each event, $e(x, y, t, p)$ is encoded by its pixel position x, y, timestamp, t, accurate to microseconds, and polarity, $p \in \{-1, 1\}$, indicating whether the intensity decreased or increased. We accumulate events within a small time interval (usually 20 ms), and use the accumulated events, which we call *event slice* in our algorithm.

We introduce a compositional network, which estimates in supervised mode the depth and a probability map for independent movements, and in unsupervised mode the camera and object pose (see Fig. 1).

The contributions of this work are:

- The first NN for estimating both camera and object 3D motion using event-based data
- The first NN architecture taking advantage of the idiosyncratic nature of event clouds
- We create *EV-IMO* - the first dataset for motion segmentation with ground truth depth, per-object mask, camera and object motion.
- A shallow low parameter multi-level feature NN architecture, with performance close to the full-sized network.

2. Related Work

2.1. Event based Optical Flow, Depth and 3D Motion Estimation

Some of the first computational methods on event-based data focused on optical flow estimation using concepts known from video processing. For example, Benosman *et al.* [4] developed a gradient based method, Barranco *et al.* [1] proposed a gradient-based concept applied to accumulated events, Benosman *et al.* and Mueggler *et al.* [3, 21] fit local planes to time stamps in local spatial neighborhoods, and Orchard *et al.* [22] developed a spiking neural network implementation of a similar fitting. Looking at the literature, we also find matching approaches e.g. [19], simple bio-inspired models implemented on embedded hardware [9], and frequency based techniques [2, 5]. Recently Zhu *et al.* [38] proposed a self-supervised deep learning approach using the intensity signal from the DAVIS camera for supervision.

3D motion has been addressed in the visual odometry and SLAM formulation assuming rotation only [23], with known maps [30, 6, 12], by combining event-based data with image measurements [18, 27], and for the general case using event data in conjunction with IMU data [37]. Other recent approaches jointly reconstruct the image intensity of the scene and estimate 3D motion. The first approach considered rotation only [16] and later Kim *et al.* [17] addressed the problem in the general case.

2.2. Learning in Structure from Motion

In pioneering work, Saxena *et al.* [25] demonstrated that shape can be learned from single images, and this work inspired many other supervised approaches on learning depth (e.g. [11]). The concept has recently been adopted in the SfM pipeline. Tateno *et al.* [26] predict the depth of key frames. Garg *et al.* [13] and Xie *et al.* [31] introduced self-supervised learning of depth using stereo and video, respectively. Most recently, Zhou *et al.* [36] took it a step further, and showed how to estimate 3D motion and depth through the supervision of optical flow. Wang *et al.* [29] instead of predicting depth in a separate network component propose to incorporate a Direct Visual Odometry (DVO) pose predictor. Mahjourian *et al.* [20] in addition to image alignment enforce alignment of the geometric scene structure in the loss function. Yang *et al.* [32] added a 3D smoothness prior to the pipeline, which enables joint estimation of edges and 3D scene. Yin *et al.* [35] include a non-rigid motion localization component to also detect moving objects. Our architecture is most closely related to SfM-Net [28], which learns using supervised and non-supervised components depth and 3D camera and object motion. However, due to the lack of a dataset, the authors did not evaluate the object motion estimation. Finally, there are two related studies in the literature on event-based data: Zhu *et al.* [37] proposed the first unsupervised learning approach and applied it for optical flow estimation using the DAVIS sensor, where the supervision signal comes from the image component of the sensor. The arXiv paper [34] first adopted the full structure from motion pipeline of [36]. Different from our work, this paper, like [37], does not take advantage of the structure of event clouds. Most important, our work also detects, segments, and estimates the 3D motion of independently moving objects, and provides the means for evaluation.

3. The Architecture

3.1. Ego-motion Model

Here we give a brief overview of several equations that are used in this paper, following the notation in [7]. We assume that the camera motion is rigid and can be described by a translational velocity v and a rotational velocity ω. We also assume that the camera intrinsic matrix K is provided. We preapply K^{-1} to work with the calibrated coordinates. Let $\mathbf{X} = (X, Y, Z)^T$ be the world coordinates of a point, which are related to the pixel coordinates as:

$$\begin{cases} x = \frac{X}{Z} \\ y = \frac{Y}{Z}. \end{cases} \tag{1}$$

The pixel velocity or the so-called optical flow (x', y') is obtained as:

$$\begin{cases} x' = \frac{X'Z - XZ'}{Z^2} = \frac{X' - \frac{XZ'}{Z}}{Z}, \\ y' = \frac{Y'Z - YZ'}{Z^2} = \frac{Y' - \frac{YZ'}{Z}}{Z}. \end{cases} \tag{2}$$

Under the rigid motion assumption, we have: $\mathbf{X}' = -\mathbf{v} - \omega \times X$, where $\times$ denotes the cross product. The expanded version of the equation is:

$$\begin{cases} X' = -v_X - \omega_Y Z + \omega_Z Y, \\ Y' = -v_Y - \omega_Z X + \omega_X Z, \\ Z' = -v_Z - \omega_X Y + \omega_Y X. \end{cases} \tag{3}$$

Substituting Eq. 2 into Eq. 3 we have:

$$\begin{cases} x' = \frac{-v_X - \omega_Y Z + \omega_Z Y - \frac{X}{Z}(-v_Z - \omega_X Y + \omega_Y X)}{Z}, \\ y' = \frac{-v_Y - \omega_Z X + \omega_X Z - \frac{Y}{Z}(-v_Z - \omega_X Y + \omega_Y X)}{Z}. \end{cases} \tag{4}$$

Using the camera projection equations (Eq. 1), the previous equations can be simplified into the algebraic form:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \frac{1}{Z} \begin{pmatrix} -1 & 0 & x \\ 0 & -1 & y \end{pmatrix} \begin{pmatrix} v_X \\ v_Y \\ v_Z \end{pmatrix} + \begin{pmatrix} xy & -1 - x^2 & y \\ 1 + y^2 & -xy & -x \end{pmatrix} \begin{pmatrix} \omega_X \\ \omega_Y \\ \omega_Z \end{pmatrix} = A\mathbf{p} \tag{5}$$

To put it in simpler words, for each pixel, given the inverse depth and the ego-motion velocities $\mathbf{v}, \omega$, we can calculate the optical flow or pixel velocity using a simple matrix multiplication (Eq. 5). Here $\mathbf{p}$ is the 3D motion (or relative pose) $(\mathbf{v}, \omega)^T$, and A is a 2×6 matrix.

In this paper, we model individual moving objects by allowing the relative pose p to take different values in different regions of the scene. The relative pose of the i-th pixel p^i is represented as:

$$p^i = p_{ego} + r^i, \tag{6}$$

where p_{ego} is the ego-motion pose of the camera, and r^i is the residual pose (motion relative to the background) of the $i - th$ pixel.

3.2. The Pipeline

We adopt an encoder-decoder network [24] to estimate the *scaled* inverse depth from a single event slice and another network of similar architecture to estimate pixel wise 3D motion (relative pose) from consecutive event slices (Fig. 3). The two network outputs are used to generate the optical flow (Fig. 2).

3.3. A Mixture Model for Ego-motion and Individual Moving Objects

The pose network utilizes a mixture model to predict pixel-wise pose. The network outputs a mixture of motion poses $(p_0, p_1, ..., p_C)$ at the end of its encoding part. For the motion estimation problem, we make p_0 to be the ego-motion pose p_{ego}. We then let the network generate a few candidate residual poses $(r_1, ..., r_C)$, and add them to p_0 to get the candidate relative poses with respect to the camera.

The network then goes through the decoding part to predict pixel-wise mixture weights or motion masks for the poses. We use the mixture weights and the pose candidates to generate pixel-wise pose. The mixture weights sum to 1 for each pixel.

For a moving object that has a non-zero residual pose, the mixture model captures all the motion that can be linearly spanned by the candidate residual poses. It captures different directions by linearly combining the residual poses. It can also represent a same motion direction at a slower speed by uniformly reducing the weights assigned to the candidate residual poses and increasing the weight assigned to the ego-motion pose.

3.4. Loss functions

We describe the loss functions used in the framework.

3.4.1 Event Warping Loss

We generate the 'soft pose' for the i-th pixel with the mixture model:

$$p^i = \sum_{c=0}^{C} M_c^i p^c \tag{7}$$

The pixel wise pose is then used to calculate the optical flow with Eq. 5.

In the training process, we calculate the optical flow and inversely warp events to compensate for the motion. This is done by measuring the warping loss at two scales, first for a rough estimate, between slices, then for a refined estimate within a slice where we take full advantage of the timestamp information in the events.

Specifically, first using the optical flow estimate, we inversely warp neighboring slices to the center slice.To measure the alignment quality at the coarse scale in the temporal domain, we take 3-5 large event slices, where each consists of 0.05 seconds of motion information, and use the absolute difference in event counts after warping as the loss:

$$Loss_{coarse} = \sum_{-N \le n \le N, n \ne 0} |I_n^{warpped} - I^{middle}|.$$

To refine the alignment, we process the event point clouds and divide the slices into smaller slices of $0.01 - 0.05$ seconds. Separately warping each of the hundreds of resulting slices allows us to utilize fully the time information contained in the continuous event stream.

We stack warped slices and use a sharpness loss to estimate the warping quality. Intuitively speaking, if the pose is perfectly estimated, the stacking of inversely warped slices should lead to a motion-deblurred sharp image. Let $S = \sum_{n=-N}^{N} |I_n^{warped}|$ be the stacking of inversely warped event slices, where n represents the n-th slice in a stack. Our basic observation is that the sparse quasi-norm $||\cdot||_p$ for $0 < p < 1$ favors a sharp non-negative image over a blurred one. That is, $\sum_i |x_i|^p \geq (\sum_i |x_i|)^p$ for $0 < p < 1$. Based on this observation, we calculate the quasi-norm of S to get the fine scale loss: $Loss_{fine} = ||S||_p, 0 < p < 1$.

3.4.2 Motion Mask Loss

Given the ground truth motion mask, we apply a binary cross entropy loss on the mixture weight of the ego-motion pose component. To enforce the mixture assignment is locally smooth, we also apply a smoothness loss on the first-order gradients of all the mixture weights.

3.4.3 Depth Loss

When the ground truth depth is available, we put constraints that the depth network output should be consistent with the ground truth. We adjust the network output and the ground truth to the same scale then apply penalties on their deviation: $loss = max(\frac{truth}{predict}, \frac{predict}{truth}) + \frac{|predict-truth|}{truth}$. Additionally we apply a smoothness penalty on the second-order gradients of the prediction values.

3.5. Evenly Cascading Network Architecture

We adopt a few more flexible designs to circumvent drawbacks of the standard encoder-decoder neural network design. In a standard neural network, the architecture is heavily hard coded. The scaling of feature maps is limited to integer values, because of the use of pooling and striding operations. In our network, we replace all scaling operations with bilinear interpolation, and therefore allow the feature map sizes to scale fractionally.

In the encoding layers, our network evenly downscales the previous feature maps by a scaling factor ($s < 1$) to get coarser and coarser features until the feature sizes fall below a predefined threshold. In the decoding layers, the feature maps are reversely upscaled by a scaling factor $1/s$. Since bilinear interpolation is locally differentiable, the gradients can be easily calculated for back propagation training. The network construction is automatic and is controlled by the scaling factor. This construction easily allows us to test networks of different depths. To our surprise, our network starts to provide meaningful results even though being very shallow (2-encoding layers), and having very few parameters (2-4k). This is probably due to the preservation and

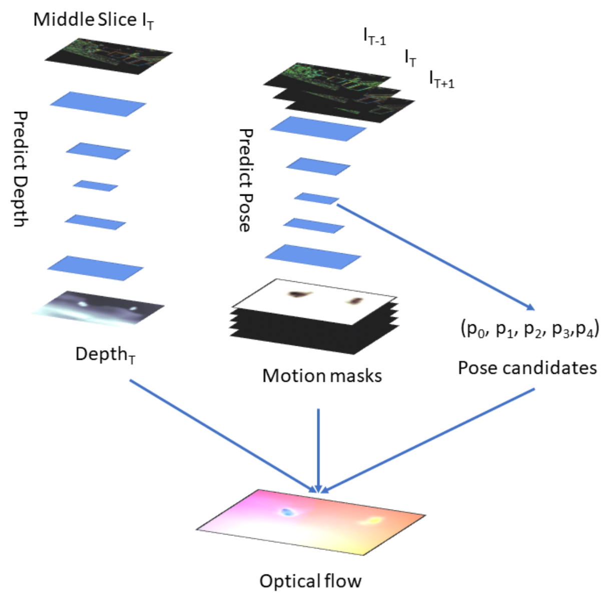

Figure 2. *A depth network (left) with an encoder-decoder architecture is used to estimate scene depth. A pose network (right) takes consecutive event slices to generate a mixture model for the pixel-wise pose. A mixture of poses and mixture probabilities are outputs of this network. The outputs of the two networks are used to generate the pixel velocities or optical flow.*

efficient utilization of multi-level features, as we explain below.

The transform of features resembles the celebrated cascade algorithm in wavelet analysis. The encoding stage [33] is analogous to the wavelet packet decomposition [8], which decomposes signals into two streams of low/high frequency coefficients. Each layer of our encoding stage contains two streams of features (Fig. 3). One stream improves the low-level features from previous layers through residual learning [15]. The other stream generates channels of higher level features from existing features. At the end of the encoding stage, the network possesses multiple levels of coarsened feature representations. Our poses prediction is made at this stage with multi-level features (Fig. 2). Our decoding stage is similar to the 'merging' operation in wavelet reconstruction. In each decoding layer, the highest level features, together with the corresponding features in the encoding layers are convolved and added back to the lower level features to improve them. At the end of the decoding stage, the network acquires a set of modulated high resolution low-level features for the final task. It is important to point out that all modulation signals are added to the lower-level features as is common in residual learning.

The evenly-cascaded (EC) structure facilitates training by providing easily-trainable shortcuts in the architecture. The mapping f from network input to output is decomposed into a series of progressively deeper, and therefore harder to

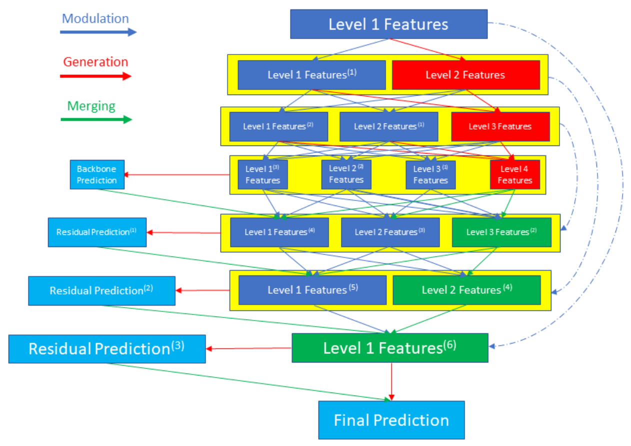

Figure 3. *The encoder-decoder structure. In the encoding stage, feature maps gradually decrease in size, while in the decoding stage the feature maps expand in size. Each encoding layer has two streams of feature maps. The downsampled features in the previous layer generate modulation signals via convolution. The modulation signal is added to the downsampled features themselves to improve them. The high-level features are concatenated with existing feature channels (red, the modulation times for the feature is braced in the superscript). At the end of the encoding stage (the layer in the middle), the network aggregates multiple levels of features. In the decoding stage, the highest-level features are merged back into the lower-level features as modulation. The corresponding encoding layer is also used as modulation (blue dash lines). We make predictions of depth at multiple scales using different features. A coarse, backbone prediction is made using multiple levels of features. The backbone prediction is then upscaled and refined using modulated lower-level features.*

train functions: $f = f_1 + f_2 + ... + f_N$. The leftmost pathway in Fig. 3 contains the easiest to train, lowest-level features, and is maintained throughout the whole network. Therefore this construction alleviates the vanishing gradient problem in neural network training, and allows the network to selectively enhance and utilize multiple levels of features.

3.6. Prediction of Depth and Component Weights

In the decoding stage, we make predictions using features at different resolutions and levels (Fig. 3). Initially, both high and low-level coarse features are used to predict a backbone prediction map. The prediction map is then upsampled and merged into existing feature maps for refinements in the remaining decoding layers. In the middle stage, high level features as well as features in the encoding layers are merged with the low level features to serve as modulation streams. Lower level features, enhanced by the modulation streams are used to estimate the prediction residue, which are usually also low-level structures. The residue is add to the current prediction map to refine it. The final prediction map is therefore obtained through successive upsamplings and refinements.

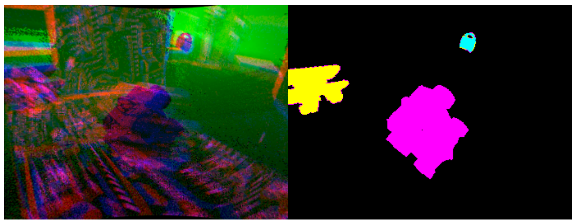

Figure 4. *An example frame from the EV-IMO dataset. The event information is overlaid on the depth ground truth on the left image, the per-object masks are provided on the right image.*

4. EV-IMO Dataset

A major contribution of our work is the *EV-IMO* dataset - the first event camera dataset to include multiple independently moving objects and camera motion (featuring extremely high speeds), while providing accurate depth maps, per-object masks and trajectories at over 200 frames per second. The next sections describe our automated labeling pipeline, which allowed us to record more than 30 sequences with a total length of over an hour. The source code for the dataset generation will be made available, to make it easier to expand the dataset in the future. A sample frame from the dataset is presented on Fig. 4.

4.0.1 Methodology

Event cameras such as the DAVIS are designed to capture high speed motion and work in difficult lighting conditions. For such conditions classical methods of collecting depth ground truth, by calibrating a depth sensor with the camera, are extremely hard to apply - the motion blur from the fast motion would render such ground truth unreliable. Depth sensors have severe limitations in their frame rate as well. Furthermore it would be impossible to automatically acquire object masks and manual (or semi-manual) annotation would be necessary. To circumvent these issues we designed a new approach:

- A static high resolution 3D scan of the objects, as well as 3D room reconstruction is performed before the dataset recording takes place.
- The VICON® motion capture system is used to track both the objects and the camera during the recording.
- The camera center as well as the object and room point clouds are calibrated with respect to the the VICON® coordinate frame.
- For every pose update from the VICON motion capture, the 3D point clouds are transformed and projected on the camera plane, generating the per-pixel mask and ground truth depth.

This method allows to record accurate depth at a very high frame rate, avoiding the problems induced by frame-based

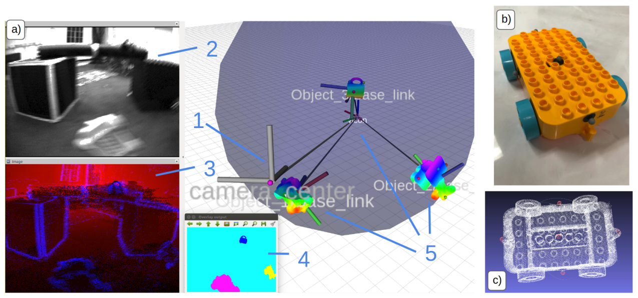

Figure 5. *a) - the main interface of the automatic annotation tool: 1 - DAVIS camera center frame, 2 - grayscale intensity image, 3 - depth image computed from room and object static 3D scans, 4 - independently moving object masks, 5 - independently moving object pointclouds brought in the world coordinate frame. b) - an example of an object used for the dataset. c) 3D scan of the object.*

collection techniques. While we acknowledge that this approach is requires expensive equipment, we argue that our method is superior for event-based sensors, since it allows to acquire the ground truth at virtually any event time stamp (by simply interpolating poses) - a property impossible to achieve with manual annotation.

4.0.2 Dataset Generation

Each of the candidate objects (up to 3 were recorded) were fitted with VICON® motion capture markers and 3D scanned using an industrial high quality 3D scanner. We use RANSAC to locate marker positions in the point cloud frame and using acquired point correspondences we transform the point cloud to the world frame at every update of the VICON.

To compute the position of the DAVIS camera center in the world frame we follow a simple calibration procedure, using a wand that is tracked by both VICON and camera.

The static pointcloud is then projected to the pixel coordinates (x, y) in the camera center frame following equation 8:

$$(x, y, 1)^T = KCP_{davis}^{-1}P_{cloud}X_i \qquad (8)$$

Here, K is the camera matrix, P_{davis} is a transformation between VICON markers on the DAVIS camera and the world, C is the transformation between VICON markers on the DAVIS and DAVIS camera center, P_{cloud} is the transformation between markers in the 3D pointcloud and VICON markers in the world coordinate frame, and X_i is the point in the 3D scan of the object.

4.0.3 Sequences

A short qualitative description of the sequences is given in Table 1. We recorded 6 sets, each consisting of 3 to 19 sequences. The sets differ in the background (in both depth and the amount of texture), the number of moving objects, motion speeds and lighting conditions.

Table 1. EV-IMO sequences

	background	speed	texture	occlusions	objects	light
Set 1	boxes	low	medium	low	1-2	normal
Set 2	floor/wall	low	low	low	1-3	normal
Set 3	table	high	high	medium	2-3	normal
Set 4	tabletop	low	high	high	1	normal
Set 5	tabletop	medium	high	high	2	normal
Set 6	boxes	high	medium	low	1-3	dark / flicker

5. Experiments

We present the first method for motion segmentation on event-based cameras with neural networks. The complexity of this task arises from the fact that event-based sensors provide extremely sparse data, and the motion can be only seen on the edges of the object. Nevertheless, we are able to estimate a 6 DoF camera egomotion, dense depth map and dense linear velocity for the independently moving objects on the scene.

For all our experiments, we selected the sequences from appropriate sets, representing the descriptions given in the Table 1 most accurately. The networks were trained on the remaining sequences of the appropriate set. For the Intersection over Union (IoU) scores, presented in Table 2 the inferenced object mask was thresholded at 0.5.

Our baseline architecture contains approximately 2 million parameters, however we understand that for many applications (such as autonomous robotics) the precision is not as an important factor as the computational efficiency and speed. We train an additional shallow network with just 40 thousand parameters. We find that the 40k network is not capable of predicting object pose reliably, but it produces reasonable camera pose, depth and motion mask.

5.0.1 Qualitative Evaluation

Apart from the quantitative comparison we present a qualitative evaluation in the Fig. 7 and Fig. 6. The per-object pose visualization (Fig. 7, columns 4 and 5) directly map the 3D linear velocity to RGB color space. The network is capable of predicting masks and pixel wise pose in scenes with different amount of motion, number of objects or texture.

Fig. 6 shows how the quality of the depth and motion mask output is affected by reducing the size of the network. While the background depth is affected only to a small degree, the quality of the object mask and depth suffers a notably.

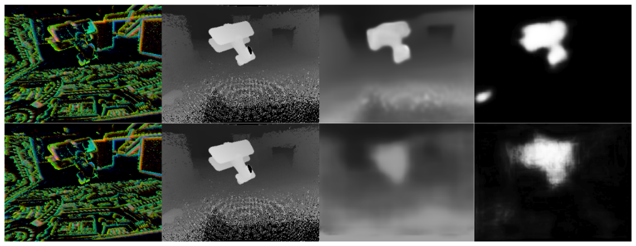

Figure 6. *Comparison of the full network inference quality (2M parameters, top row) with the small version (40k parameters, bottom row)*

5.0.2 Segmentation and Motion Estimation

We compute the classical Average Endpoint Error (AEE) in 3D to evaluate the quality of motion estimation on the independently moving objects and the egomotion. Since we estimate the full 6 DoF pose for the camera motion, we compute separate errors - AEE_{tr} and AEE_{rot}, for linear and rotation components of the pose.

The per-pixel linear pose on the independently moving objects is averaged with the object mask. To evaluate the segmentation we compute the commonly used Intersection over Union (IoU) metrics. The network is evaluated on the first 3 sets of *EV-IMO*, since they include the majority of independently moving object scenes and feature the most varying camera and object motions. Our results are presented in Table 2.

Table 2. Evaluation on segmentation and motion estimation. The numbers in braces are values for the 40k version of the network.

	Cam AEE_{tr}	Cam AEE_{rot}	Obj AEE	IOU
Set 1	0.07 (0.09)	0.05 (0.08)	0.19	0.83 (0.63)
Set 2	0.17 (0.23)	0.16 (0.24)	0.38	0.75 (0.58)
Set 3	0.23 (0.28)	0.20 (0.26)	0.43	0.73 (0.59)

5.0.3 Comparison With Previous Work

We also evaluate our approach against a recent method [34] - *ECN* network, which estimates optical flow and depth on the event-based camera output. The method was originally designed and evaluated on a road driving sequence (which features a notably more simple and static environment, as well as significantly rudimentary egomotion). Still, we were able to tune [34] and train it on *EV-IMO*. We provide the comparison for the depth for our baseline method, the smaller version of our network (with just 40k parameters) and *ECN* in Table 3.

Here we conducted experiments on sequences featuring a variety of backgrounds and textures (the lack of texture is a limiting factor for event-based sensors). *ECN* should not be affected by the presence of the independently moving objects, since it infers depth from a single frame.

Table 3. Evaluation of the depth estimation

	Error metric			Accuracy metric		
	Abs Rel	RMSE log	SILog	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
			Baseline Approach			
low texture	0.16	0.26	0.07	0.87	0.95	0.97
cube background	0.13	0.20	0.04	0.87	0.97	0.99
fast motion	0.31	0.32	0.12	0.74	0.90	0.95
			40k Network			
low texture	0.24	0.33	0.11	0.75	0.90	0.95
cube background	0.20	0.26	0.07	0.77	0.92	0.97
fast motion	0.33	0.34	0.15	0.65	0.87	0.95
			ECN			
low texture	0.67	0.59	0.33	0.27	0.52	0.80
cube background	0.60	0.56	0.30	0.29	0.53	0.78
fast motion	0.47	0.48	0.23	0.45	0.69	0.86

6. Conclusions

Event-based sensing promises advantages over classic video processing in applications of motion processing because of the data's unique properties of sparseness, high temporal resolution, and low latency. In this paper, we provided a solution to the problem of dynamic scene recovery assuming rigid object motions, which we consider an application that can draw significant benefits from utilizing event based data. We presented a NN using a combination of unsupervised and supervised components, with the latter employed in processes that generalize well across different scenes. We presented the first ever method with evaluation of both camera and object estimation, which was achieved through the creation of a new state of the art indoor dataset - *EV-IMO*, recorded with the use of a VICON® motion capture system. Experimental results demonstrate good performance on very challenging data of fast motion, confirming the intuition of the approach.

Future work will delve into a number of issues regarding the design of the NN and usage of event data. Specifically, we consider it crucial to study event stream augmentation using partially or fully simulated data. We also plan to investigate ways to include tracking and connect the estimation over successive time slices, and investigate different alternatives of including the grouping of objects into the pipeline.

References

[1] F. Barranco, C. Fermüller, and Y. Aloimonos. Contour motion estimation for asynchronous event-driven cameras. *Proceedings of the IEEE*, 102(10):1537–1556, 2014. 2

[2] F. Barranco, C. Fermüller, and Y. Aloimonos. Bio-inspired motion estimation with event-driven sensors. In *International Work-Conference on Artificial Neural Networks*, pages 309–321. Springer, 2015. 2

[3] R. Benosman, C. Clercq, X. Lagorce, S.-H. Ieng, and C. Bartolozzi. Event-based visual flow. *Neural Networks and Learning Systems, IEEE Transactions on*, 25(2):407–417, 2014. 2

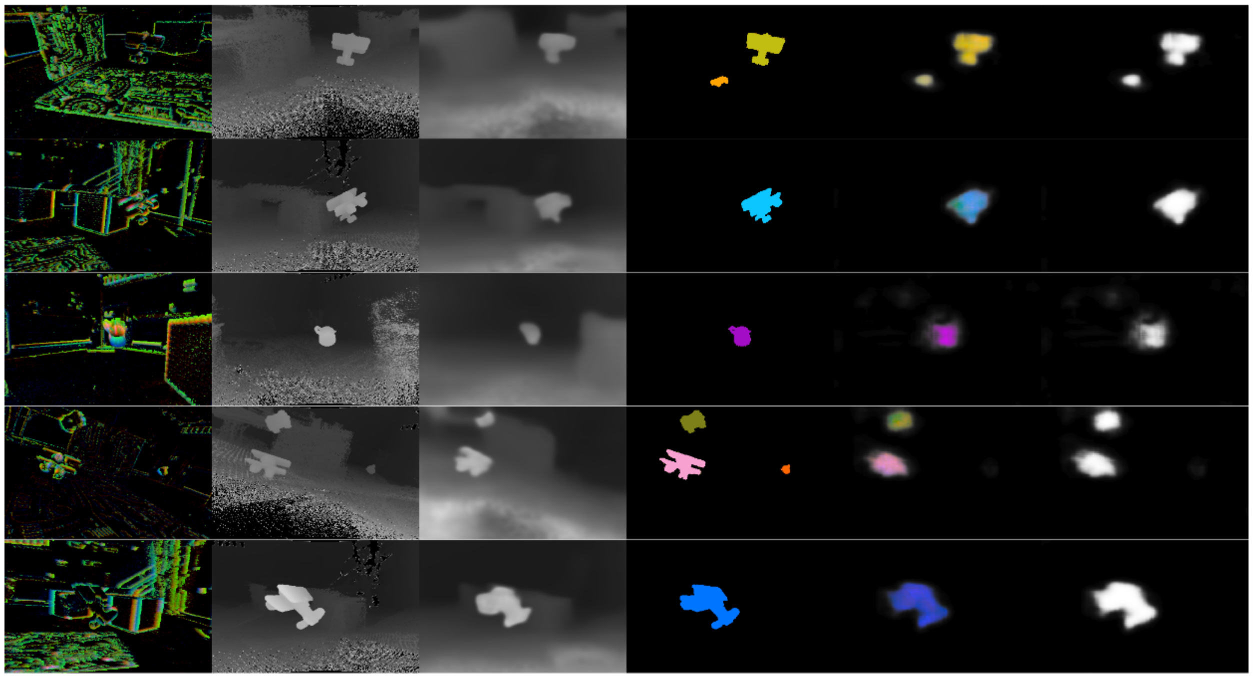

Figure 7. *Qualitative results from our evaluation. The table entries from left to right: DVS input, ground truth for depth, network output for depth, ground truth pixel wise pose, predicted pixel wise pose, predicted motion mask. Examples were collected from EV-IMO dataset. Best viewed in color.*

[4] R. Benosman, S.-H. Ieng, C. Clercq, C. Bartolozzi, and M. Srinivasan. Asynchronous frameless event-based optical flow. *Neural Netw.*, 27:32 – 37, Mar. 2012. 2

[5] T. Brosch, S. Tschechne, and H. Neumann. On event-based optical flow detection. *Frontiers in neuroscience*, 9, 2015. 2

[6] A. Censi, J. Strubel, C. Brandli, T. Delbruck, and D. Scaramuzza. Low-latency localization by active led markers tracking using a dynamic vision sensor. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 891–898. IEEE, 2013. 2

[7] F. Chaumette and S. Hutchinson. Visual servo control, Part I: Basic approaches. *IEEE Robotics and Automation Magazine*, 13(4):82–90, 2006. 2

[8] R. R. Coifman and M. V. Wickerhauser. Entropy-based algorithms for best basis selection. *IEEE Trans. Inf. Theor.*, 38(2):713–718, Sept. 2006. 4

[9] J. Conradt. On-board real-time optic-flow for miniature event-based vision sensors. In *Robotics and Biomimetics (ROBIO), 2015 IEEE International Conference on*, pages 1858–1863. IEEE, 2015. 2

[10] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (6):1052–1067, 2007. 1

[11] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, pages 2366–2374, 2014. 2

[12] G. Gallego, J. E. A. Lund, E. Mueggler, H. Rebecq, T. Delbrück, and D. Scaramuzza. Event-based, 6-dof camera tracking for high-speed applications. *CoRR*, abs/1607.03468, 2016. 2

[13] R. Garg, V. K. BG, G. Carneiro, and I. Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *European Conference on Computer Vision*, pages 740–756. Springer, 2016. 2

[14] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. 1

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 4

[16] H. Kim, A. Handa, R. Benosman, S.-H. Ieng, and A. J. Davison. Simultaneous mosaicing and tracking with an event camera. In *British Machine Vision Conference*, 2014. 2

[17] H. Kim, S. Leutenegger, and A. J. Davison. Real-time 3d reconstruction and 6-dof tracking with an event camera. In *European Conference on Computer Vision*, pages 349–364. Springer, 2016. 2

[18] B. Kueng, E. Mueggler, G. Gallego, and D. Scaramuzza. Low-latency visual odometry using event-based feature tracks. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 16–23. IEEE, 2016. 2

[19] M. Liu and T. Delbruck. Block-matching optical flow for dynamic vision sensors: Algorithm and fpga implementation. In *Circuits and Systems (ISCAS), 2017 IEEE International Symposium on*, pages 1–4. IEEE, 2017. 2

[20] R. Mahjourian, M. Wicke, and A. Angelova. Unsupervised learning of depth and ego-motion from monocular video using 3d geometric constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5667–5675. 2

[21] E. Mueggler, C. Forster, N. Baumli, G. Gallego, and D. Scaramuzza. Lifetime estimation of events from dynamic vision sensors. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 4874–4881. IEEE, 2015. 2

[22] G. Orchard, R. Benosman, R. Etienne-Cummings, and N. V. Thakor. A spiking neural network architecture for visual motion estimation. In *Biomedical Circuits and Systems Conference (BioCAS), 2013 IEEE*, pages 298–301. IEEE, 2013. 2

[23] C. Reinbacher, G. Munda, and T. Pock. Real-time panoramic tracking for event cameras. *arXiv preprint arXiv:1703.05161*, 2017. 2

[24] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI (3)*, volume 9351 of *Lecture Notes in Computer Science*, pages 234–241. Springer, 2015. 3

[25] A. Saxena, S. H. Chung, and A. Y. Ng. Learning depth from single monocular images. In *Advances in neural information processing systems*, pages 1161–1168, 2006. 2

[26] K. Tateno, F. Tombari, I. Laina, and N. Navab. Cnn-slam: Real-time dense monocular slam with learned depth prediction. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 6565–6574. IEEE, 2017. 2

[27] D. Tedaldi, G. Gallego, E. Mueggler, and D. Scaramuzza. Feature detection and tracking with the dynamic and active-pixel vision sensor (davis). In *Event-based Control, Communication, and Signal Processing (EBCCSP), 2016 Second International Conference on*, pages 1–7. IEEE, 2016. 2

[28] S. Vijayanarasimhan, S. Ricco, C. Schmid, R. Sukthankar, and K. Fragkiadaki. Sfm-net: Learning of structure and motion from video, 2017. 1, 2

[29] C. Wang and J. M. Buenaposada. Learning depth from monocular videos using direct methods. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2022–2030. 2

[30] D. Weikersdorfer, R. Hoffmann, and J. Conradt. Simultaneous localization and mapping for event-based vision systems. In *International Conference on Computer Vision Systems*, pages 133–142. Springer, 2013. 2

[31] J. Xie, R. Girshick, and A. Farhadi. Deep3d: Fully automatic 2d-to-3d video conversion with deep convolutional neural networks. In *European Conference on Computer Vision*, pages 842–857. Springer, 2016. 2

[32] Z. Yang, P. Wang, Y. Wang, W. Xu, and R. Nevatia. LEGO: learning edge with geometry all at once by watching videos. In *Proc. IEEE Computer Vision and Pattern Recognition Conference*, 2018. 2

[33] C. Ye, C. Devaraj, M. Maynord, C. Fermüller, and Y. Aloimonos. Evenly cascaded convolutional networks. *CoRR*, abs/1807.00456, 2018. 4

[34] C. Ye, A. Mitrokhin, C. Parameshwara, C. Fermüller, J. A. Yorke, and Y. Aloimonos. Unsupervised learning of dense optical flow and depth from sparse event data. *arXiv preprint arXiv:1809.08625*, 2018. 2, 7

[35] Z. Yin and J. Shi. Geonet: Unsupervised learning of dense depth, optical flow and camera pose. *CoRR*, abs/1803.02276, 2018. 2

[36] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, 2017. 1, 2

[37] A. Z. Zhu, N. Atanasov, and K. Daniilidis. Event-based visual inertial odometry. 2

[38] A. Z. Zhu, L. Yuan, K. Chaney, and K. Daniilidis. Ev-flownet: Self-supervised optical flow estimation for event-based cameras. *Robotics: Science and Systems*, 2018. 2