Fully Convolutional Neural Network for Event Camera Pose Estimation

Ahmed Tabia, Fabien Bonardi and Samia Bouchafa-Bruneau
IBISC, Univ. Evry, Universite Paris-Saclay, 91025, Evry, France

Keywords: 6-DOF, Pose Estimation, Deep Learning, Event-Based Camera.

Abstract: Event cameras are bio-inspired vision sensors that record the dynamics of a scene while filtering out unnecessary data. Many classic pose estimation methods have been superseded by camera relocalization approaches based on convolutional neural networks (CNN) and long short-term memory (LSTM) in the investigation of simultaneous localization and mapping systems. However, and due to the usage of LSTM layer these methods are easy to overfit and usually take a long time to converge. In this paper, we introduce a new method to estimate the 6DOF pose of an event camera with a deep learning. Our approach starts by processing the events and generates a set of images. It then uses two CNNs to extract relevant features from the generated images. Those features are multiplied using the outer product at each location of the image and pooled across locations. The model ends with a regression layer which outputs the estimated position and orientation of the event camera. Our approach has been evaluated on different datasets. The results show its superiority compared to state-of-the-art methods.

1 INTRODUCTION

The relocalization of the camera pose, which aims at inferring the position and the orientation of the camera from an observed scene (Qu et al., 2022), is a fundamental problem in many computer vision applications, such as autonomous vehicle driving, robotics, augmented reality, and pedestrian visual positioning. Conventional computer vision relocalization methods can be categorized into two categories: (1) the geometric-based and (2) the learning-based approaches. Geometric-based approaches (Mur-Artal and Tardós, 2017) are mainly based on local feature matching. Their standard process consists of extracting a set of local features from a given image and performing a 2D-3D matching with corresponding 3D points and then calculating the camera pose of six degrees of freedom generally using Perspective-n-Point algorithms (Lepetit et al., 2009). This category of approaches highly relies on the accurate feature extraction and matching process, which is not always satisfied, particularly in the case of illumination variations (Li et al., 2020).

More recently, with the resurgence of deep learning, notably, Convolutional Neural Networks (CNN), many computer vision applications have been revisited with data-driven approaches. The new methods achieved high performance in different tasks such as object recognition (Eitel et al., 2015), image classification (Mahajan et al., 2018), and segmentation (Badrinarayanan et al., 2017). Deep learning-based approaches have shown a high ability to extract robust features (Kendall et al., 2015), however, they require a large amount of training data (usually thousands of images) and many computational resources (powerful and expensive GPUs). That is why approaches that do not require recomputing every time on such extensive data might be more interesting. For example transfer learning and integration could be compelling alternatives to alleviate this issue.

Moreover, both conventional camera relocalization categories of methods are still suffering from illumination changes, blur, and flat images in which it is difficult to extract features. These issues are mainly due to the nature of the input images captured by conventional cameras and fed into these methods.

Event cameras which are also known as neuromorphic cameras are imaging sensors that respond to local changes in brightness. Differently from conventional cameras, the raw output of these cameras is a sequence of asynchronous events (discrete pixel-wise changes in brightness) corresponding to changes in the scene illumination. They have several advantages compared to conventional cameras. They have a high temporal resolution, extensive dynamic range, and no motion blur. These advantages make the usage of the event cameras ideal in the context of robotic applications, particularly pose estimation.

Tabia, A., Bonardi, F. and Bouchafa-Bruneau, S.
Fully Convolutional Neural Network for Event Camera Pose Estimation.
DOI: 10.5220/0011641500003417
In *Proceedings of the 18th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2023) - Volume 4: VISAPP*, pages 594-599
ISBN: 978-989-758-634-7; ISSN: 2184-4321

Build upon these advantages, Rebecq et al. (Rebecq et al., 2017) presented a method which combines IMU and event information to estimate the 6DOF camera pose. More recently, a method called (SP-LSTM) has been proposed to estimate the 6DOF of an event camera by (Nguyen et al., 2019). The method uses a VGG16 achitecture (Simonyan and Zisserman, 2014) trained from scratch with stochastic gradient descent algorithm and two stacked spatial LSTM layers. The method achieves promising results in camera pose estimation, but requires an expensive training time due to the retraining of the full network model with the LSTM layer.

In this paper, we present a new method which alleviate the problems of LSTM layers. We propose a deep learning model composed of two CNNs aimed to extract relevant features from event images. The extracted features are then aggregated using the outer product at each location of the image and pooled using a bilinear pooling operation (Lin et al., 2015). We also leverage new development made for deep learning and employ the ADAM optimizer (Kingma and Ba, 2014) along with ELU activation function (Clevert et al., 2015). We conducted experiments on different datasets and report superior results compared to state-of-art methods.

The rest of this paper is organized as follows. The proposed method is presented and detailed in Section 2. In section 3, we present extensive experimental results. Finally, we conclude this paper and discuss future work.

2 PROPOSED METHOD

We proposed a novel attention based fully convolutional neural network for pose estimation. The proposed attention mechanism helps the model focus on the motion-relevant regions in images. Given a sequence of events captured using an event camera, we first process the raw event and create a set of event images following (Nguyen et al., 2019). The image preprocessing is presented in Section 2.1. Once the events are converted into images shown in figure 1, they are fed into a convolutional neural network. The extracted features are then aggregated using bilinear pooling (Lin et al., 2015) vector for the pose estimation. The details about the model architecture are given in Section 2.2.

2.1 Image Preprocessing

Unlike normal frame-based cameras, which capture a whole image at a predetermined time interval, event cameras only capture a single event at a timestamp depending on brightness changes at a local pixel. The initial stage in this project is to solve pose relocalisation problem, inspired by (Nguyen et al., 2019) (Kendall et al., 2015) (Kendall and Cipolla, 2016), first we took the event stream and transform it to an event image $I \in \mathbb{R}^{h*w}$, where h and w are the dimension of the event image. Formally, the event e is a tuple represent by,

$$e = < e_t, (e_x, e_y), e_p >$$

where e_t is the timestamp of the event, (e_x, e_y) is the pixel coordinate and $e_p = \pm 1$ is the polarity that denotes the brightness change at the current pixel. The event image is computed from the event stream as follows:

$$I(e_x, e_y) = \begin{cases} 0 \ \ if \ e_p \ = -1 \\ 1 \ \ if \ e_p \ = \ \ 1 \end{cases} \tag{1}$$

The second step is to enlarge the image to have 224×224 pixel size, in accordance with the original image's aspect ratio and give it as input to the CNNs. Figure 1 shows an example of event images obtained after the preprocessing from event stream (Gallego and Scaramuzza, 2017). The preprocessing step plays an important role since it affects the quality of the event images, which are used to train the CNN and estimate the camera relocalisation.

2.2 The Network Model

In our method, we propose to extract different sets of features from the event image. We employ two convolutional neural networks denoted respectively A and B (see Figure 2). Two feature maps are extracted from the networks A and B which apply several pooling and non-linear transformations to the original event image. The intuition is that A and B learn different features from the input image. Then the output of both A and B are combined by a bilinear pooling layer. This layer provides a powerful representation which fuses the two sets of features by leveraging the higher-order information captured in the form of pairwise correlations between the extracted features. In our experiments, we use the pretrained MobileNetV2 (Sandler et al., 2018) model as a first feature extractor A and a VGG16 (Simonyan and Zisserman, 2014) as a second feature extractor B. The used MobileNetV2 and VGG16 have already been trained on a very large collection of images from ImageNet (Deng et al., 2009) both models achieved excellent results of relocalisation and image classification challange.

Let us denote the event camera pose by $y = [p, q]$, where $p \in \mathbb{R}^3$ represents the three dimensional camera position and the quaternion $q \in \mathbb{R}^4$ codes the cam-

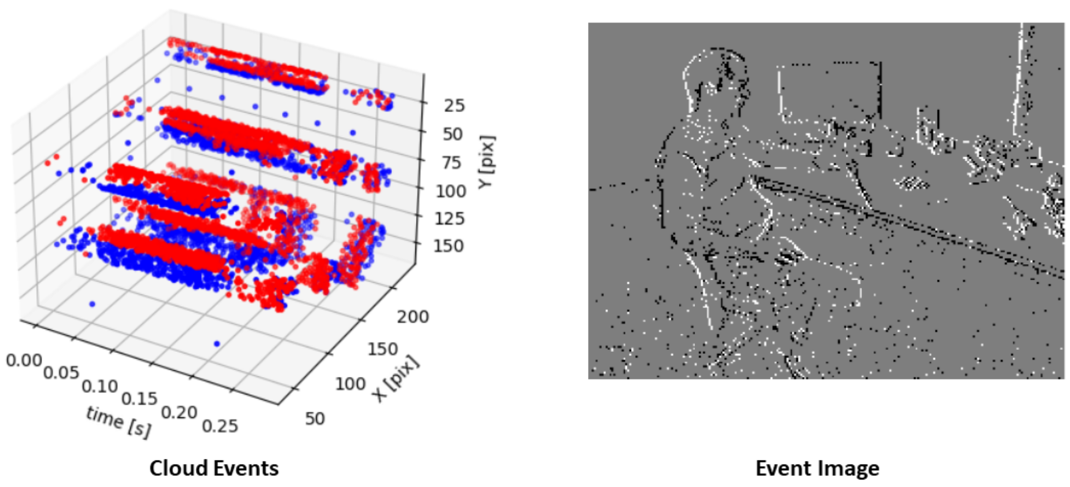

Figure 1: Image preprocessing from point cloud events to event image.

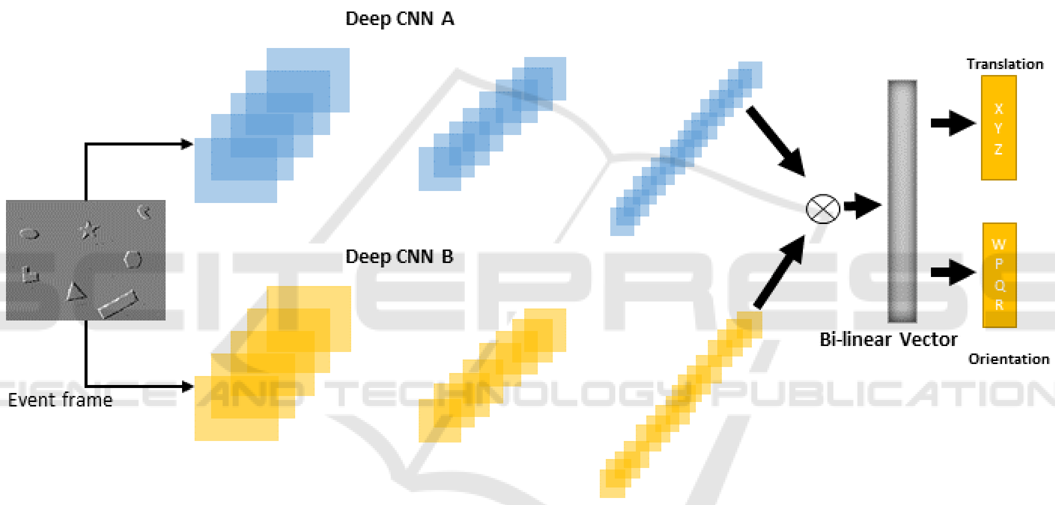

Figure 2: An overview of our 6DOF pose relocalization method for event cameras . We first create an event image from stream of events. Then we extract features from the created event image using a Bilinear pooling. Then the feature vector is then given to a fully connected layer of seven neurons is used to regress the camera pose vector.

era orientation. In our experiments, the used CNNs have been pretrained on the ImageNet dataset (Deng et al., 2009) with input dimensions of $224 \times 224 \times 3$. Deep features are learned from the input event images obtained from the preprocessing step. Both network A and B outputs feature maps represented respectively by the matrix V of dimensionality $n \times d$, and the matrix U of size $m \times d$. Here, n and m are the number of kernels in the output layers of the networks A and B, respectively. The dimensionality of each filter is d; it is obtained by flattening the 2-dimensional feature map, i.e., the output image that has undergone several kernel convolutions and pooling transformations. The bilinear pooling operation is then defined as:

$$X = UV^T, U \in \mathbb{R}^{m*d}, V \in \mathbb{R}^{n*d}, X \in \mathbb{R}^{m*n} \qquad (2)$$

The connection between the CNN outputs and the bilinear pooling is preceded by an ELU (Clevert et al., 2015) activation function $F(x)$. It is defined as:

$$F(x) = \begin{cases} x & x > 0 \\ \alpha(e^x - 1) & x <= 0 \end{cases} \qquad (3)$$

In order to regress the seven-dimensional pose vector, a linear regression layer is added at the end of the model (see Figure 2 the output layer).

Training Settings and Loss Function. In our method, we train our model using Adam (Kingma and Ba, 2014) optimizer (with parameters $\beta_1 = 0.9$, and $\beta_2 = 0.999$) to obtain high performance by calculating the adaptive learning rate of each hyper parameter, and to prevent redundancy and get faster gradient update.

We choose the smooth $l1$ loss instead of the mean square loss function in our implementation. The smooth $l1$ loss over n samples is defined as:

$$smoothl1(x,y) = \frac{1}{n}\sum_{i=1}^{n} z_i \tag{4}$$

where z_i is given by :

$$z_i = \begin{cases} 0.5(x_i - y_i)^2/\beta, & \text{if } |x_i - y_i| < \beta \\ |x_i - y_i| - 0.5 * \beta, & \text{otherwise} \end{cases} \tag{5}$$

where x and y are the ground truth and the target camera pose vectors, respectively. β is an optional parameter which specifies the threshold at which to change between l1 and l2 loss. As β varies, the l1 segment of the loss has a constant slope of 1. In our implementation we set β equals to 1.

Following Kendall et al. (Kendall et al., 2015) work, at the test phase we normalize the quaternion to unit length, and utilize Euclidean distance to assess the difference between two quaternions. The distance should be measured in spherical space in theory, but in reality, the deep network produces a predicted quaternion $\widehat{q}$ that is close enough to the groundtruth quaternion q. This makes the difference between the spherical and Euclidean distance insignificant.

3 EXPERIMENTAL RESULTS

The proposed method has been evaluated on a collection of six real event data. In this section, we present the datasets used for the method evaluation, the training environment, and the experimental results.

3.1 Dataset

We conducted experiments on the event camera dataset that was collected by (Mueggler et al., 2017). The dataset includes a collection of scenes captured by a DAVIS240C from minilabs. They contain the cloud of events, images, IMU measurements, and camera calibration from the DAVIS. The groundtruth camera poses are collected from a motion-capture system with sub-millimeter precision at 200Hz. We adopt the timestamp of the motion-capture system to build event frames. All the events with the timestamps between t and $t+1$ of the motion-capture system is grouped as one event image. Without using the loss of generality, we consider the ground-truth pose of this event image as the camera pose shot by the motion capture system at instant $t+1$. This method technically limits the speed of the event camera to the speed of the motion capture system. We Follow the same evaluation protocol as in (Nguyen et al., 2019). The protocol includes two type of splits:

- **The Random Split** where we select 70% of the event images for training and the remaining 30% for testing.
- **The Novel Split** in which we select the first 70% of each event for the training, then the rest 30% of the event for the test. In this way, we have two independent sequences on the same scene. Following Nguyen et al. (Nguyen et al., 2019) the training sequence is selected from timestamp t_0 to t_{70}, and the testing sequence is from timestamp t_{71} to t_{100}).

3.2 Training Environment

Once the preprocessing stage is performed, patches of size 224×224 pixels are taken from each frame and fed into the CNNs in a patch-level dataset. To evaluate the effectiveness of our proposed method in this paper, we conduct several experiments and compare our results with those of deep learning architectures with state of the art models using LSTM. We used Pytorch (Paszke et al., 2019) to implement the proposed method. All our experiments have been conducted on a platform composed of a processor Intel(R) Xeon(R) CPU @ 2.00GHz, a CPU memory of size 24GB, and a single Tesla T4 GPU. The networks have been trained with 350 epochs with a learning rate equals to $2\exp-3$ with momentum-decay equals to $4\exp-3$ and a weight decay set to 0.

3.3 Results

We use the same protocol of comparison reported in (Nguyen et al., 2019) and used in PoseNet (Kendall et al., 2015) and Bayesian PoseNet (Kendall and Cipolla, 2016). As quantitative evaluation, we choose to calculate the median and average error of the predicted pose in position and orientation. The Euclidean distance is used to compare the predicted position to the groundtruth, and the anticipated orientation is normalized to unit length before being compared to the groundtruth. For location and orientation, the median and average error are recorded in m and deg(°), respectively.

3.3.1 Comparison with State-of-the-Art Methods

we report the comparison results between our method explained on the section 2.2 and the state of the art models namely PoseNet(Kendall et al., 2015),

Table 1: Comparison between our method results and the results of PoseNet (Kendall et al., 2015), Bayesian PoseNet (Kendall and Cipolla, 2016) and SP-LSTM (Nguyen et al., 2019). The evaluation is performed using the random split protocol.

	PoseNet (Kendall et al., 2015)		**Bayesian PoseNet** (Kendall and Cipolla, 2016)		**SP-LSTM** (Nguyen et al., 2019)		**Ours**	
	Median Error	Average Error	Median Error	Average Error	Median Error	Average Error	Median Error	Average Error
shapes rotation	0.109m, 7.388°	0.137m, 8.812°	0.142m, 9.557°	0.164m, 11.312°	0.025m, 2.256°	0.028m, 2.946°	0.018m, 1.753°	0.020m, 2.551°
shapes translation	0.238m, 6.001°	0.252m, 7.519°	0.264m, 6.235°	0.269m, 7.585°	0.035m, 2.117°	0.039m, 2.809°	0.033m, 2.211°	0.036m, 2.717°
box translation	0.193m, 6.977°	0.212m, 8.184°	0.190m, 6.636°	0.213m, 7.995°	0.036m, 2.195°	0.042m, 2.486°	0.029m, 1.507°	0.032m, 1.693°
dynamic 6dof	0.297m, 9.332°	0.298m, 11.242°	0.296m, 8.963°	0.293m, 11.069°	0.031m, 2.047°	0.036m, 2.576°	0.027m, 1.802°	0.029m, 2.394°
hdr poster	0.282m, 8.513°	0.296m, 10.919°	0.290m, 8.710°	0.308m, 11.293°	0.051m, 3.354°	0.060m, 4.220°	0.040m, 2.937°	0.051m, 3.783°
poster translation	0.266m, 6.516°	0.282m, 8.066°	0.264m, 5.459°	0.274m, 7.232°	0.036m, 2.074°	0.041m, 2.564°	0.036m, 2.045°	0.039m, 2.315°
Average	0.231m, 7.455°	0.246m, 9.124°	0.241m, 7.593°	0.254m, 9.414°	0.036m, 2.341°	0.041m, 2.934°	**0,030m, 2.204°**	**0.034m, 2.708°**

Table 2: Comparison between our method results and the results of PoseNet (Kendall et al., 2015), Bayesian PoseNet (Kendall and Cipolla, 2016) and SP-LSTM (Nguyen et al., 2019). The evaluation is performed using the novel split protocol.

	PoseNet (Kendall et al., 2015)		**Bayesian PoseNet** (Kendall and Cipolla, 2016)		**SP-LSTM** (Nguyen et al., 2019)		**Ours**	
	Median Error	Average Error	Median Error	Average Error	Median Error	Average Error	Median Error	Average Error
shapes rotation	0.201m, 12.499°	0.214m, 13.993°	0.164m, 12.188°	0.191m, 14.213°	0.045m, 5.017°	0.049m, 11.414°	0.050, 3.681°	0.053m, 6.823°
shapes translation	0.198m, 6.969°	0.222m, 8.866°	0.213m, 7.441°	0.228m, 10.142°	0.072m, 4.496°	0.081m, 5.336°	0.062m, 4.554°	0.068m, 5.854°
shapes 6dof	0.320m, 13.733°	0.330m, 18.801°	0.326m, 13.296°	0.329m, 18.594°	0.078m, 5.524°	0.095m, 9.532°	0.071m, 5.787°	0.091m, 7.550°
Average	0.240m, 11.067°	0.255m, 13.887°	0.234m, 10.975°	0.249m, 14.316°	0.065m, 5.012°	0.075m, 8.761°	**0.061m, 3.448°**	**0.070m, 6.742°**

Bayesian PoseNet (Kendall and Cipolla, 2016) and SP-LSTM(Nguyen et al., 2019) using CNN and LSTM.

3.3.2 Random Split

The results reported in this table 1 have been obtained using the random split strategy. We use 6 sequences (**shapes rotation, box translation, shapes translation, dynamic 6dof, hdr poster, poster translation**) for this experiment. In all this sequences, our model obtains the lowest mean and average errors. It achieves 0.030m and 2.204° in average median error of all sequence of the real dataset while the most recent method SP-LSTM result 0.036m, 2.341°, PoseNet and Bayesian PoseNet results are 0.231m, 7.455° and 0.241m, 7.593°, respectively

3.3.3 Novel Split

Table 2 presents comparison results from the novel split presented in Section 3.1. One can notice from this table that the novel split is more difficult to handle than the random split the errors from all methods are bigger than errors reported with the random split. We use three sequences from the shapes scene (**shapes rotation, shapes translation, shapes 6dof**) in this novel split. The results of our method are superior to the results obtained with state of the art methods. It achieves 0.061m and 3.448° in average median error of all sequence of the real dataset while the most recently method SP-LSTM result 0.065m, 5.012°, and 0.240m, 11.067° and 0.234m, 10.975° from PoseNet and Bayesian PoseNet errors, respectively.

We recall that in the novel split, the testing set is selected from the last 30% of the event images. This means we do not have the "neighborhood" relationship between the testing and training images. In the random split, the testing images can be very close to the training images since we select the images randomly from the whole sequence for training/testing.

To conclude, the extensive experimental results from both the random split and novel split setup show that our method successfully relocalizes the event camera pose using only the event image coming from the cloud of polarity. The critical reason for the improvement is using bilinear pooling to learn the spatial relationship features in the event image . The experiments using the novel split setup also confirm that our approach successfully encodes the scene's geometry during the training and generalizes well during the testing. Furthermore, our network also has a speedy inference time and requires only the event image as the input to relocalize the camera pose.

4 CONCLUSION

In this paper, we introduce new method to estimate the 6DOF pose of an event camera with a deep learning. First, the events are preprocessed to build event images. Then a set of features are extracted using a deep convolutional neural network bilinear pooling. These extracted features are aggregated and fed into a single layer which is connected to a fully connected layer for the pose regression. In our training, we used the adam optimizer instead of conventional stochastic gradient descent. We also used ELU activation functions. Furthermore, our method has fast inference time and needs only the event image to relocalize the camera pose. The results on publicly available datasets show that our approach generalizes well and outperforms recent works including LSTM based architectures.

REFERENCES

Badrinarayanan, V., Kendall, A., and Cipolla, R. (2017). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495.

Clevert, D.-A., Unterthiner, T., and Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

Eitel, A., Springenberg, J. T., Spinello, L., Riedmiller, M., and Burgard, W. (2015). Multimodal deep learning for robust rgb-d object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 681–687. IEEE.

Gallego, G. and Scaramuzza, D. (2017). Accurate angular velocity estimation with an event camera. *IEEE Robotics and Automation Letters*, 2(2):632–639.

Kendall, A. and Cipolla, R. (2016). Modelling uncertainty in deep learning for camera relocalization. In *2016 IEEE international conference on Robotics and Automation (ICRA)*, pages 4762–4769. IEEE.

Kendall, A., Grimes, M., and Cipolla, R. (2015). Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE international conference on computer vision*, pages 2938–2946.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Lepetit, V., Moreno-Noguer, F., and Fua, P. (2009). Epnp: An accurate o (n) solution to the pnp problem. *International journal of computer vision*, 81(2):155–166.

Li, M., Chen, R., Liao, X., Guo, B., Zhang, W., and Guo, G. (2020). A precise indoor visual positioning approach using a built image feature database and single user image from smartphone cameras. *Remote Sensing*, 12(5):869.

Lin, T.-Y., RoyChowdhury, A., and Maji, S. (2015). Bilinear cnn models for fine-grained visual recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 1449–1457.

Mahajan, D., Girshick, R., Ramanathan, V., He, K., Paluri, M., Li, Y., Bharambe, A., and Van Der Maaten, L. (2018). Exploring the limits of weakly supervised pretraining. In *Proceedings of the European conference on computer vision (ECCV)*, pages 181–196.

Mueggler, E., Rebecq, H., Gallego, G., Delbruck, T., and Scaramuzza, D. (2017). The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and slam. *The International Journal of Robotics Research*, 36(2):142–149.

Mur-Artal, R. and Tardós, J. D. (2017). Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE transactions on robotics*, 33(5):1255–1262.

Nguyen, A., Do, T.-T., Caldwell, D. G., and Tsagarakis, N. G. (2019). Real-time 6dof pose relocalization for event cameras with stacked spatial lstm networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Qu, C., Shivakumar, S. S., Miller, I. D., and Taylor, C. J. (2022). Dsol: A fast direct sparse odometry scheme. *arXiv preprint arXiv:2203.08182*.

Rebecq, H., Horstschaefer, T., and Scaramuzza, D. (2017). Real-time visual-inertial odometry for event cameras using keyframe-based nonlinear optimization.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.