An End-to-end Moving Camera Background Model

Supplemental Material

Guy Erez$^{1\,[0000-0002-4545-6664]}$, Ron Shapira Weber$^{1\,[0000-0003-4579-0678]}$, and Oren Freifeld$^{1\,[0000-0001-9816-9709]}$

Ben-Gurion University of the Negev, Be'er Sheva, Israel
`{ergu,ronsha}@post.bgu.ac.il, orenfr@cs.bgu.ac.il`

Abstract. The supplemental material includes, in addition to this document, select examples of **videos showing the results of our method next to the original videos**. Additionally, and due to space limits, additional visual comparisons between the different methods (using PDF presentations so it be will easy to browse back and forth between the frames) are available at `https://github.com/BGU-CS-VIL/DeepMCBM`.

As for **this document**, it contains the following:

1. A comparison of ROC curves of different methods on various videos. These curves correspond to the AUC values reported in Table 1 in the paper.
2. A demonstration of DeepMCBM's capability of predicting the background in previously-unseen misaligned frames from the video it was trained on (a capability lacking in most other methods, except JA-POLS [4]).
3. The details of the robust error functions that we used.
4. The technical details of the evaluation procedures.
5. The technical details of the architecture and training process.
6. An explanation how, in the affine Spatial Transformer Net, the invertibility of the affine transformations is guaranteed via the matrix exponential.

1 ROC Curves (Related to the Values from Table 1 in the Paper)

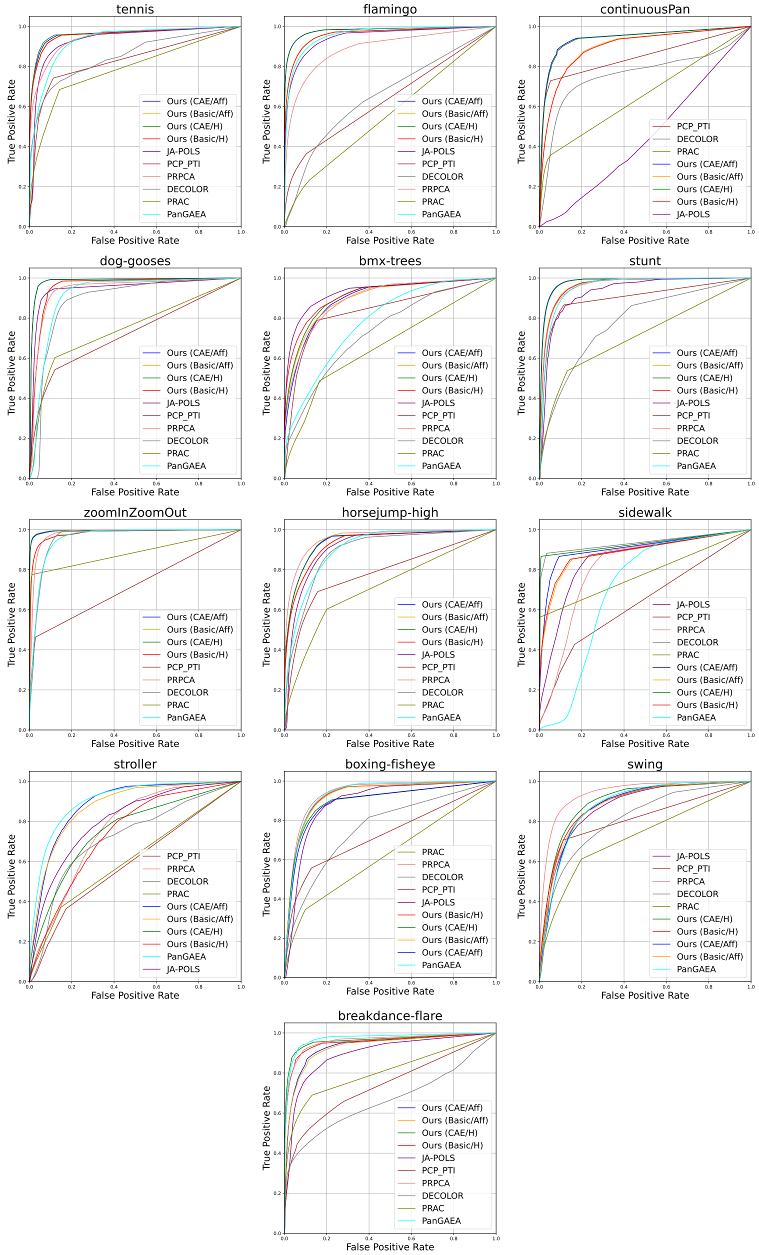

Fig. 1: A comparison of ROC curves of different methods on various videos. Note that sometimes some curves coincide (*e.g.*, our CAE/Aff and CAE/Hom on the flamingo video).

2 Predicting Background to Previously-unseen Misaligned Frames

In the paper, our comparison of DeepMCBM to the other methods focused on their own playground: *i.e*., given an input sequence of video frames, the task was viewed as an optimization problem whose overarching goal is to estimate the background in those frames. However, like JA-POLS [4] but unlike the other competitors, DeepMCBM, being based on (unsupervised) learning, also possesses a generalization capability in the sense it can estimate the background in previously-unseen misaligned frames from the video it was trained on. In contrast, other methods (JA-POLS excluded), do not have a readily-available mechanism that enables them to receive such misaligned frames and predict their alignment and background estimation – at least not without solving additional optimization steps. Note that in a static camera background model this is a non-issue since, by definition, the frames (both train and test) are always aligned; however, in the moving-camera case the situation is different due to the misalignment. Remark: Note that this generalization capability should not be confused with an online-learning setting – which some competitors aim for – where each time the next consecutive frame arrives their model is being updated (using further optimization).

To showcase the generalization capability, in each of the "tennis" and "flamingo" sequences (from [10]) we partitioned the video sequence into train and test sets. We did it by letting the test set (in each of the two videos) consist of 10% of the frames (chosen at random from the original sequence) while letting the train test consist of the remaining 90% of the frames. In each video we let both DeepMCBM and its competitor, JA-POLS, train only on the train set. Next, we evaluated the models, using their prediction functionalities, on the test sets. Figure 2 and Table 1 summarize the results in terms of the corresponding ROC curve and AUC scores, respectively, and show that DeepMCBM outperforms JA-POLS in this type of evaluation as well. Note that here we intentionally used our CAE/Aff variant (and not CAE/Hom) to highlight the fact that even when using the same transformation type as JA-POLS (*i.e*., affine), DeepMCBM beats the latter.

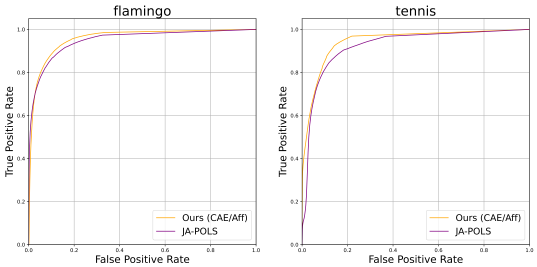

Fig. 2: A typical comparison of ROC curves, between the proposed method and JA-POLS [4], of the performance on test sets.

Sequence	DeepMCBM	JA-POLS [4]
tennis test	0.944	0.923
tennis train	0.965	0.936
flamingo test	0.957	0.949
flamingo train	0.980	0.949

Table 1: AUC scores for test and train sets. DeepMCBM in its CAE+Affine version

3 The Robust Error Functions

We have used the smoothed ℓ_1 loss (which is closely-related to Huber's function [3]) for ρ_{JA} and Geman-McClure's error function [5] for ρ_{recon}:

$$\rho_{\mathrm{JA}}(\varepsilon;\beta) = \begin{cases} 0.5\varepsilon^2/\beta & \text{if} \quad |\varepsilon| \leq \beta \\ |\varepsilon| - 0.5\beta & \text{otherwise} \end{cases} \qquad \rho_{\mathrm{recon}}(\varepsilon; s) = \frac{\varepsilon^2}{\varepsilon^2 + s^2} \tag{1}$$

where in all of our experiments we used $\beta = 0.35$ and $s = 0.05$.

4 Evaluation Details

In this section we explain how the ROC curves and AUC scores were obtained.

Given an input sequence of N frames, $(f^n)_{n=1}^N$, **ground-truth annotations** of foreground pixels, $(a^n)_{n=1}^N$, and a background estimation sequence, $(\widehat{f^n})_{n=1}^N$, we compute the pixelwise squared error $(E^n)_{n=1}^N$, where $E^n_{\boldsymbol{x}} = \frac{1}{C}\sum_{c=1}^C (f^n_{\boldsymbol{x},c} - \widehat{f}^n_{\boldsymbol{x},c})^2$ is the error in pixel $\boldsymbol{x} \in \Omega$ (recall that Ω, a rectangle of height h and width w, was defined in the paper as the common domain of the original input frames), averaged across C channels. To have a unified comparison measure applicable in all videos and for all methods, we scale to the $[0,1]$ interval where the scaling is done (for the method under consideration) w.r.t. the entire video sequence. That is, we define the scaled error of that method in the specific video as

$$\widetilde{E}^n_{\boldsymbol{x}} = \frac{E^n_{\boldsymbol{x}} - \min_{n,\boldsymbol{x}'}(E^n_{\boldsymbol{x}'})}{\max_{n,\boldsymbol{x}'}(E^n_{\boldsymbol{x}'}) - \min_{n,\boldsymbol{x}'}(E^n_{\boldsymbol{x}'})} \,. \tag{2}$$

Next, for each threshold value α (from a discrete set of evenly-spaced points between 0 and 1) we computed the True Positive Rate (TPR) and False Positive Rate (FPR):

$$\mathrm{TPR} = \frac{1}{N \cdot h \cdot w} \sum_{n=1}^N \sum_{\boldsymbol{x}\in\Omega} \mathbb{1}(a_{\boldsymbol{x}} = 1 \wedge \widetilde{E}^n_{\boldsymbol{x}} \geq \alpha)\,; \tag{3}$$

$$\mathrm{FPR} = \frac{1}{N \cdot h \cdot w} \sum_{n=1}^N \sum_{\boldsymbol{x}\in\Omega} \mathbb{1}(a_{\boldsymbol{x}} = 0 \wedge \widetilde{E}^n_{\boldsymbol{x}} \geq \alpha)\,. \tag{4}$$

Computing the TPR and FPR for multiple threshold values lets us obtain the Receiver Operating Characteristic (ROC) curve for this (method,sequence) pair. Figure 1 shows the resulted ROC curves while Area Under the (ROC) Curve (AUC) scores are summarized in Table 1 in the paper.

5 Architecture and Training Details

DeepMCBM's pipeline consists of two main parts:

1. a Spatial Transformer Net (STN);
2. a Conditional Autoencoder (CAE).

First, an STN is trained to learn an input-dependent function, that predicts the transformation parameter vector $\boldsymbol{\theta}$. These predictions are used to create the panoramic central moments as well as, later, to unwarp those moments back towards the input image. Once the STN training process has converged, the STN module is frozen and the CAE is trained to learn a background model in the input domain, where the conditioning is on the aforementioned unwapred moments. Below we describe the architecture and training details of the STN and CAE modules.

5.1 The STN

The STN we used consisted of a backbone and tailored regression heads for each available transformation type (in our case, either affine of homographies). The backbone we used was ResNet18 [6] whose output size was 1000. Each regression head consisted of two dense layers of size 1000×32 and $32 \times d$, where d is the dimension of the transformation parameters vector $\boldsymbol{\theta}$ ($d = 6$ for affine transformations and $d = 8$ for homographies). In both the regressor heads we used a ReLU [1] activation function. In all our experiments, we trained the backbone and the Affine head for the first 3000 epochs. In the variants that used homographies, we first did repeat the process above (that is, 3000 epochs for training the backbone and the affine head) and then switched to train the homographic head for additional 3000 epochs (recall that our affine transformations are invertible and that such transformations are a particular case of homographies; thus, the results from the affine head provide a good initialization for the homographic head). Thus, in total, the STN module was trained for about 6000 epochs with a step learning rate scheduler, decreasing the learning rate every 1000 epochs with an initial learning rate of 0.005.

5.2 The CAE

The second module in the DeepMCBM pipeline is a CAE, conditioned on the unwarped panoramic central moments. In our experiments, we conditioned the CAE on the first two such moments, but more generally, any number of moments can be used as well. Theoretically, using more moments implies that the distribution of the pixel stack is better captured.

Our CAE was based on the (unconditional) Autoencoder from [2]. Using a similar architecture, we used 4 channels in each hidden convolutional layer and 2 channels for the last convolutional layer. In between the convolutional encoder and decoder we used dense layers of size 256×4 and 4×256 (thus, the code size was 4) with a ReLU [1] activation function. All convolutional layers used a 5×5 kernel with stride size 2.

Our conditioning is done in both the encoder and decoder parts. As the conditioning (the unwarped central moments) is in the same spatial dimensions as the input image, conditioning on the encoder size was done by a simple concatenation, along the channel dimension, of the input and the unwarped moments. This results in $(N_{\text{moments}} + 1) \cdot C$ input channels for the encoder where N_{moments} is the number of the moments used ($N_{\text{moments}} = 2$ in our experiments) and C is the number of channels of an input image ($C = 3$ in the RGB case). To condition the decoder, we concatenate the unwaped moments to the (classic) decoder output and then use a small Convolutional Neural Net (CNN) to integrate the decoder's output and the conditioning. This last CNN is composed of 3 convolutional layers with 50 channels for the hidden layesr and C channels for the output layer. All three layers use a 3×3 kernel, and a ReLU [1] activation function.

Using the robust reconstruction loss mentioned in the paper, we trained the CAE for 2000 epochs with a learning rate scheduler and a step size of 500 epochs and initial learning rate of 0.001.

6 Using the Matrix Exponential within an STN

Let $\boldsymbol{A}$ be a 3×3 real-valued matrix such that its last row is all zeros:

$$\boldsymbol{A} = \begin{bmatrix} A_1 & A_2 & A_3 \\ A_4 & A_5 & A_6 \\ 0 & 0 & 0 \end{bmatrix} . \tag{5}$$

Then, a known result (which is also widely-used in computer vision; see, *e.g.*, [8,9]) from the theory of matrix groups is that

$$\boldsymbol{T} \triangleq \exp(\boldsymbol{A}) \tag{6}$$

(*i.e.*, the matrix exponential of $\boldsymbol{A}$) has the following form,

$$\boldsymbol{T} = \begin{bmatrix} T_1 & T_2 & T_3 \\ T_4 & T_5 & T_6 \\ 0 & 0 & 1 \end{bmatrix} \tag{7}$$

where, in addition, we have that $\det \boldsymbol{T} > 0$ (in particular, $\boldsymbol{T}$ is invertible). Consequently, the matrix exponential provides a mapping from the unconstrained linear space $\mathbb{R}^6$ into the Affine Group (namely, the space of invertible affine transformations – in this case from $\mathbb{R}^2$ to $\mathbb{R}^2$). Taken together with the fact that the matrix exponential is a differentiable function, this means that an STN [7] can be easily set to produce only invertible transformations [11,4].

References

1. Agarap, A.F.: Deep learning using rectified linear units (relu). arXiv preprint arXiv:1803.08375 (2018) 6, 7
2. Ballé, J., Laparra, V., Simoncelli, E.P.: End-to-end optimized image compression. In: ICLR (2017) 7
3. Black, M.J., Rangarajan, A.: On the unification of line processes, outlier rejection, and robust statistics with applications in early vision. IJCV (1996) 5
4. Chelly, I., Winter, V., Litvak, D., Rosen, D., Freifeld, O.: JA-POLS: a moving-camera background model via joint alignment and partially-overlapping local subspaces. In: CVPR (2020) 1, 4, 5, 7
5. Geman, S., McClure, D.E.: Statistical methods for tomographic image reconstruction. In: BISI (1987) 5
6. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016) 6
7. Jaderberg, M., Simonyan, K., Zisserman, A., et al.: Spatial transformer networks. In: NIPS (2015) 7
8. Lin, D., Grimson, E., Fisher III, J.: Learning visual flows: A Lie algebraic approach. In: CVPR (2009) 7
9. Lin, D., Grimson, E., Fisher III, J.: Modeling and estimating persistent motion with geometric flows. In: CVPR (2010) 7
10. Perazzi, F., Pont-Tuset, J., McWilliams, B., Van Gool, L., Gross, M., Sorkine-Hornung, A.: A benchmark dataset and evaluation methodology for video object segmentation. In: CVPR (2016) 4
11. Skafte Detlefsen, N., Freifeld, O., Hauberg, S.: Deep diffeomorphic transformer networks. In: CVPR (2018) 7