Supplemental Material for the CVPR 2020 Paper "JA-POLS: a Moving-camera Background Model via Joint Alignment and Partially-overlapping Local Subspaces"

Irit Chelly
Ben-Gurion University
tohamy@post.bgu.ac.il

Vlad Winter
Ben-Gurion University
winterv@post.bgu.ac.il

Dor Litvak
Ben-Gurion University
dorlit@post.bgu.ac.il

David Rosen
Massachusetts Institute of Technology
dmrosen@mit.edu

Oren Freifeld
Ben-Gurion University
orenfr@cs.bgu.ac.il

Abstract

This document contains the following:

1. *Additional qualitative results (comparison of BG/FG separation produced by each method; comparing JA-POLS versus t-GRASTA on the Jitter dataset; performance of JA-POLS on fairly-wide scenes; performance of JA-POLS on a previously-unseen video (test data) of the same scene it was trained on).*
2. *An explanation of the automatic refinement of the predicted alignment and showing that using PoseNet would have needed a similar refinement step.*
3. *A comparison of different choices for using RPCA within our POLS model.*
4. *Additional ablation studies.*
5. *Analysis of the effect of the size of the local domain in the POLS model.*
6. *Details about the regressor net we use to predict transformation for new test images.*
7. *The procedure we use for estimating noisy relative transformations.*
8. *The specs of the machine on which we ran our experiments.*
9. *The relevant mathematical background of matrix Lie groups and Lie algebras needed to understand some of the details in our method.*

1. Additional Qualitative Results

Figure 1 shows a comparison of BG/FG separation produced by the proposed method (JA-POLS) and each of the competing methods. In Fig. 2, we compare the performance of JA-POLS and t-GRASTA on the Jitter dataset, after inserting synthetic objects (elephant and "Deadpool" figures) as foreground objects with known Ground-truth (GT) binary masks. The GT masks are used only when computing the F-measure performance index. Figure 3 visualizes JA-POLS' performance on a fairly-wide scene (GardenWideScene dataset), on a previously-unseen video (*i.e.*, *test data*). This was obtained upon learning, in an unsupervised fashion, from another video (*i.e.*, training data) of the same scene (at a different time, and using a different trajectory for the camera). Particularly, note we show that JA-POLS is capable of detecting not only moving objects but also static changes (see the figure's caption for details).

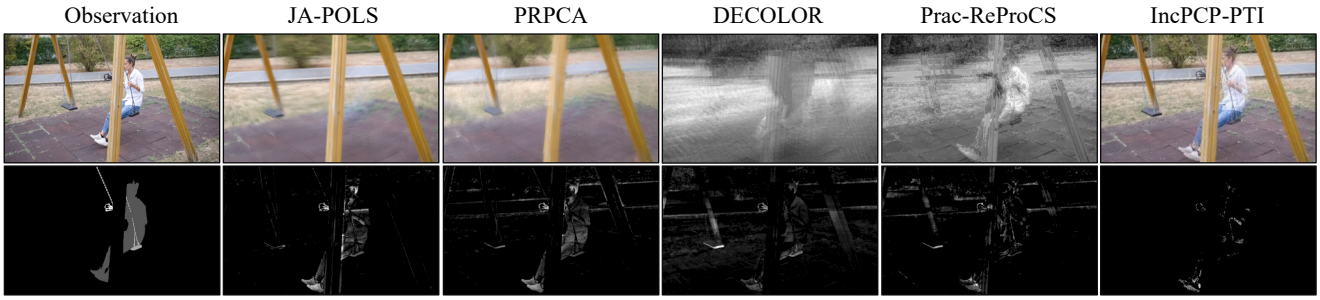

Figure 1: A typical frame from the Swing dataset. Left column: observation (top row) and ground-truth foreground (bottom). Other columns: estimated background (top) and foreground (bottom) produced by each method. Please zoom in to better see the details.

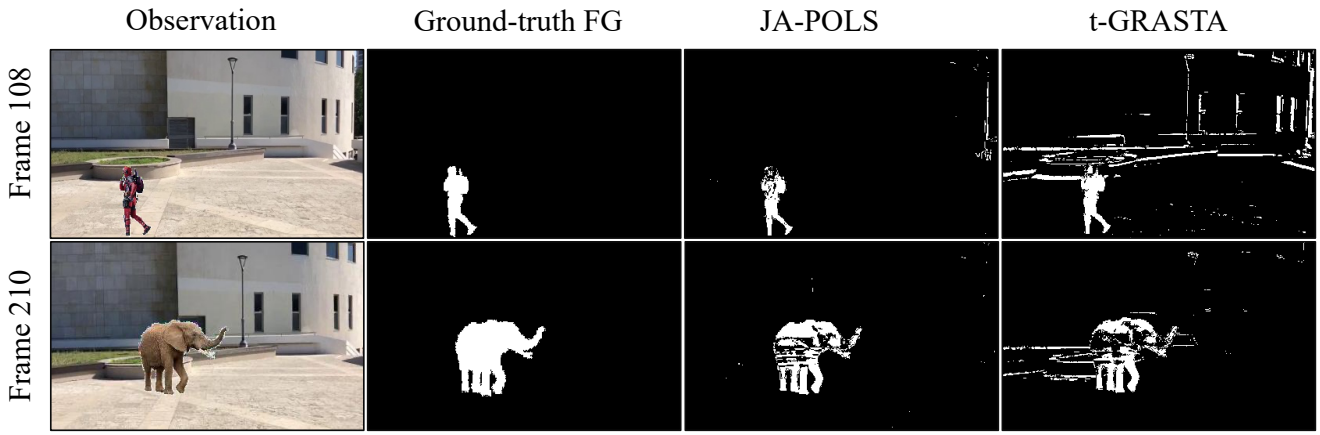

Figure 2: Two typical frames of the Jitter dataset using inserted synthetic objects, their ground-truth masks, and the estimated foreground by JA-POLS and t-GRASTA.

Figure 3: JA-POLS performance on GardenWideScene's test data. In the video, a backpack (circled in yellow in frame 244), which is part of the background, is being displaced to another location (frames 290, 461). The absence of the backpack and its new static location are detected correctly as "foreground = non-background" (red circles) in the sense they are not part of the learned background.

2. Refinement of the Predicted Alignment in the Test Phase

As discussed in the paper, in order to estimate a background/foreground for an unseen image we first align it w.r.t. the global scene, and then project it on its overlapping subspaces. In such a task, the precision of the image

alignment is crucial, as even a small deviation from the correct image location may impair the projection results drastically, and thus also hurt the BG/FG estimations. Let $T^{\boldsymbol{\theta}} \in \mathrm{Aff}(2)$ be the alignment predicted by our regression net for test image x. Thus, T locates x near the correct location in the global scene. As mentioned above, this estimation could of course be imperfect. Let $x' = x \circ T^{\boldsymbol{\theta}}$ be the warped image, Ω' be the domain of x', and Ω'_{enclose} be the domain of the smallest rectangle that encloses x'. We refine the prediction by estimating the homographic residual transformation, H, of x' toward the image received by the pixelwise averaging of the pixel stacks associated with Ω'_{enclose} (note well: Ω'_{enclose} is much smaller than Ω_{scene}, where the latter was defined in the paper). The homography estimation procedure itself is standard, and is based on SIFT (for feature extraction), FLANN (for feature matchings), and RANSAC (for robustness). As we mention elsewhere in this document, OpenCV's function, **flann.knnMatch**, conveniently provides a single wrapper to both the feature extraction and FLANN.

As the initial prediction is close enough to the desired location, this task is easily done and results in a small *final* estimation error. Next, we warp x' further using H. This results in an improved relocalization of x, which is now ready to be projected on the POLS model.

In § 4.2 in this document, we show that doing *only* the procedure described here, without using the regression net, is a bad idea. In other words, trying to directly estimate the transformation that would take the original test image x toward the scene is inferior to using our learning-based approach followed by refinement.

2.1. PoseNet's Inaccuracy in Image Alignment

If we ignore the expensive cost associated with training PoseNet (which involves an expensive 3D reconstruction as preprocessing), the latter can, at least conceptually, replace the proposed alignment-prediction module in our pipeline. However, as Figure 4 shows, this process will also require a refinement step due to similarly-imprecise results. Thus, we prefer our inexpensive approach.

Figure 4: Posenet's results. This figure is reproduced from the intro figure of the PoseNet paper [5]. Left: input image. Right: estimated camera pose. PoseNet results are undeniably impressive, *at least for the task it was designed to solve*. However, in the context of background models, the alignment needs to have pixel-level accuracy. Thus, any net predictions which are not providing a precise transformation, might cause large errors in the projection process on the background model. An example for such imprecise prediction can be seen in the area delineated by a blue circle: The real turret and the projected turret (as estimated by PostNet) are at least several pixels away from each other, prohibiting a successful application of a background model at this region.

3. Background Models Learned and Used in POLS

During POLS learning, we divide the scene into partially-overlapping domains and learn a local subspace model in each of them separately. As each such domain is treated as the domain of a dataset captured by a static camera, we are now free to choose any established method for subspace learning. In the context of background modeling, where the training data is contaminated by outliers (*e.g.*, people moving in the scene during the learning), Robust PCA (RPCA) models are widely used and are preferred over vanilla PCA. Thus, within our POLS model, we experimented with using either one of the following three RPCA models: TGA [4] ($k = 5$; 60% trimming), the denoising RPCA (Denoising-RPCA) which was proposed as part of the pipeline in [9], and the method by Candes

et al. [2]. Empirically, we found that using TGA within our POLS model gave the best results when the dataset is large, while using Denoising-RPCA (again, within our POLS model) was the best option when the dataset is small (*e.g*., < 80 frames). See Table 1 for a detailed comparison on several datasets. Note that when learning local (small) subspaces, even non-scalable RPCA methods (*e.g*. the Denoising-RPCA used in [9] or the method from [2]) are able to run without, *e.g*., causing memory issues. This is in sharp contrast to what happens when using a global model that should cover the entire scene.

Sequence	TGA(k=5, 80% Trimming)	Denoising-RPCA	Candes
Tennis	0.55 ± 0.03	$\mathbf{0.67} \pm 0.03$	0.61 ± 0.04
Stroller	0.41 ± 0.03	$\mathbf{0.62} \pm 0.03$	0.38 ± 0.03
ContinuousPan	$\mathbf{0.67} \pm 0.05$	0.62 ± 0.06	0.65 ± 0.03

Table 1: F-measure performance (mean±std) of JA-POLS, using different RPCA methods via POLS learning.

4. Additional Ablation Studies

4.1. The Importance of the Regularization Term in STN

In this section we present an ablation study that extends the experiment performed in the paper (see the Results section in the paper, Table 3), by providing a numerical measure of the importance of both the SE-Sync initialization and the regularization term that we use during the joint-alignment process. We performed this test on two datasets: Jitter and ContinuousPan. As Table 2 shows, the highest F-measure is achieved when using both the regularization and the SE-Sync initialization. When using regularization term but without the SE-Sync initialization, the STN can stll provide fairly-good alignment results, but only on easy alignment tasks (*e.g*. Jitter dataset, which presents only small motions) while its performance drastically deteriorates when running on wider scenes (*e.g*. ContinuousPan dataset) which present large accumulative motion. In these cases, the STN usually reaches poor local minima, as the alignment problem is too hard.

When omitting the regularization, the performance, in terms of F-measure, drops to zero, regardless whether the SE-Sync is used or not. In these cases, and due to the absence of a regularization that controls the transformation magnitudes, the STN reaches catastrophic results such as shrinking all images to a point, translating all images outside of the scene's domain, or placing the images such that none of them overlaps with another. All these situations are almost near the (bad) global minimum of zero alignment loss. Thus, the latter behavior also produces, of course, the smallest STN loss. Moreover, the resulting transformations are usually very far from the SE(2) group. To summarize, this study shows that both the SE-Sync initialization and the proposed regularization are important to our joint-alignment process.

Sequence	SE-Sync+STN				STN Only			
	w/ regularization		w/o regularization		w/ regularization		w/o regularization	
	F-measure	Loss	F-measure	Loss	F-measure	Loss	F-measure	Loss
Jitter	$\mathbf{0.83 \pm 0.03}$	0.018	0.03 ± 0.06	0.017	0.80 ± 0.03	0.021	0.00 ± 0.00	0.016
ContinuousPan	$\mathbf{0.67 \pm 0.05}$	0.064	0.01 ± 0.02	0.025	0.44 ± 0.07	0.123	0.00 ± 0.00	0.051

Table 2: F-measure performance (mean±std) and the STN loss (including the regularization loss in the relevant columns) of JA-POLS on the Jitter and ContinuousPan data, testing the impact of the SE-Sync and the regularization term individually.

4.2. The Need for a Regressor Net

In order to evaluate the importance of the regressor module, we compared JA-POLS with the following alternative pipeline: given a new test image, perform its alignment using only the procedure from the refinement step, by warping the test image toward the panoramic-size image provided by the pixel-stack average of already-aligned training images. Table 3 shows that when omitting the regressor from the pipeline, the performance degrades as the scene gets wider. This is an expected behaviour as the transformation parameters estimated by the refinement

step are likely to contain larger errors when the initial localization of the test image is far from the desired location in the global domain.

Sequence	w/ Regressor	w/o Regressor
Kitchen	**0.51** $\pm$ 0.04	0.49 $\pm$ 0.03
GardenLong	**0.59** $\pm$ 0.02	0.56 $\pm$ 0.02
ContinuousPan	**0.70** $\pm$ 0.04	0.48 $\pm$ 0.13

Table 3: F-measure performance (mean$\pm$std) of JA-POLS with and without the regressor module.

5. Analysis of the Effect of the Size of the Local Domain in the POLS Model

Recall that the POLS model uses partially-overlapping domains, where each domain is a rectangle that covers a part of the scene. In Fig. 5 we examine JA-POLS performance using different domain sizes used in POLS. If the size is too small, it hurts the performance. However, if the size is large, then the process becomes too expensive in terms of computation and memory *without improving the performance*. Thus, we merely chose the size 250×420 (in all our experiments) which is small enough to run fast and large enough to provide good results.

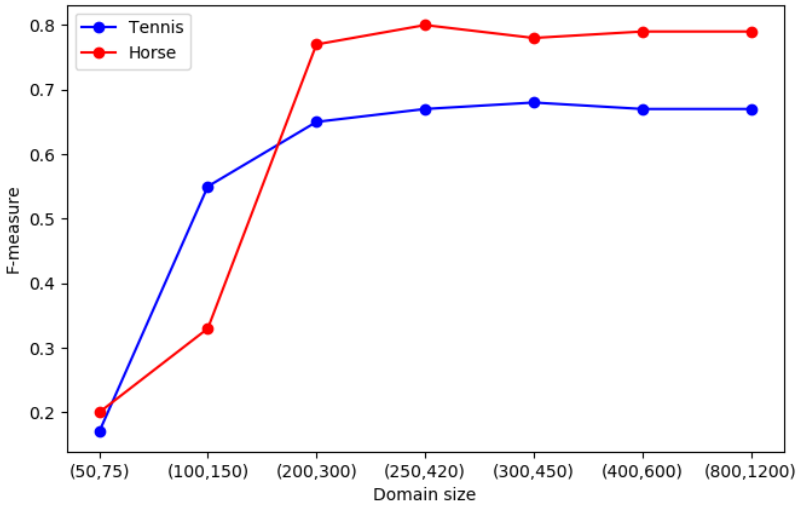

Figure 5: F-measure performance (higher is better) of JA-POLS on the Tennis and the Horse sequences using different domain sizes.

6. The Regression Net Used to Predict Transformation of New Test Frames

We use a slightly-modified version of GoogleNet [11]. The original net consists of 22 parameter layers and 5 average and max pooling layers. Most of the layers are located in the Inception modules, which were first introduced in GoogleNet. Inception modules were designed to improve the problems of over-fitting and computational costs. There are 3 classifier networks in total, connected to the 4th, 7th and the 9th (the last) Inception modules. These auxiliary networks are thought to combat vanishing gradients and provide some regulrization. However, later experiments [11] showed that the effect of the auxiliary network was seemingly minor ($\sim 0.5\%$). Thus, we discard the first 2 auxiliary classifiers and modify the last one with a fully-connected layer of size 256×6 to allow the regression of $\boldsymbol{\theta}_i \in \mathbb{R}^6$. As discussed in the paper, we initialize the network with parameters that were pre-trained on ImageNet to facilitate transfer learning. As the training set is relatively small, we perform data augmentation by creating N_{aug} additional synthesized data points for each original data point in the training set, where N_{aug} is pre-defined. The augmentation is done as follows: given a pair of $(x, T^{\boldsymbol{\theta}})$, where x is an input frame, $T^{\boldsymbol{\theta}} \in \text{Aff}(2)$

is its ground-truth invertible affine transformation parameterized by $\boldsymbol{\theta} \in \mathbb{R}^6$, we sample $\boldsymbol{\theta}^{\text{aug}} \in \mathbb{R}^6$ in the matrix group (*i.e.* $T^{\boldsymbol{\theta}^{\text{aug}}} \in \text{Aff}(2)$), which forms a synthesized residual affine-transformation matrix. In order to have a correctly-augmented data, we use the composition $(T^{\boldsymbol{\theta}_{\text{aug}}})^{-1} \circ T^{\boldsymbol{\theta}}$ as the generated ground-truth transformation. We then have the new synthestized pair $(x \circ T^{\boldsymbol{\theta}^{\text{aug}}}, (T^{\boldsymbol{\theta}_{\text{aug}}})^{-1} \circ T^{\boldsymbol{\theta}})$ as an additional example in the training set. When sampling, we use a sufficiently-small standard deviation to keep the transformation realistic enough to mimic possible camera movements.

7. Estimation of Relative Transformations

We now discuss how we extract the relative SE transformations that are used as noisy measurements in the SE-Sync module. The estimation of a relative transformation, $\widetilde{g}_{ij} \in \text{SE}(2)$, between a pair of images, x_i and x_j, is done via standard computer-vision tools. Particularly, first we use the OpenCV library [1] for both the extraction of SIFT features [8] in each image and the feature matching [10] (namely, the estimation of correspondences between the two feature sets) via a Fast Library for Approximate Nearest Neighbors (FLANN). Conveniently, OpenCV's function, `flann.knnMatch`, provides a single wrapper to these two operations. The result is a set of r matching-point pairs

$$\boldsymbol{P} = \begin{bmatrix} \boldsymbol{p}_1 & \dots & \boldsymbol{p}_r \end{bmatrix} \quad \boldsymbol{Q} = \begin{bmatrix} \boldsymbol{q}_1 & \dots & \boldsymbol{q}_r \end{bmatrix} \qquad \boldsymbol{p}_k, \boldsymbol{q}_k \in \mathbb{R}^2, \quad \forall k \in \{1, \dots, r\} \tag{1}$$

where r is the number of obtained matches, and $(\boldsymbol{p}_k, \boldsymbol{q}_k)$ are the pixel coordinates of the k-th pair of matching features in frames x_j and x_i, respectively. As the correspondences are noisy and usually also contain some errors (due to wrong matching of features), we formulate an optimization problem over SE(2):

$$(\widetilde{\boldsymbol{t}}_{ij}, \widetilde{\boldsymbol{R}}_{ij}) = \underset{\boldsymbol{R} \in \text{SO}(2),\, \boldsymbol{t} \in \mathbb{R}^2}{\arg\min} \sum_{k=1}^{r} \|(\boldsymbol{R}\boldsymbol{p}_k + \boldsymbol{t}) - \boldsymbol{q}_k\|_{\ell_2}^2 . \tag{2}$$

This is a linear least squares (LS) problem with *nonlinear constraints*. Fortunately, the problem in Eq. (2) still admits a closed-form solution [12]; we employ it together with RANSAC [3], by keeping only a subset of $r' \leq r$ points that yield a minimal error using the LS solution, in order to robustify the estimate. The result of the overall estimation procedure is not only the relative transformation, $\widetilde{g}_{ij} = \begin{bmatrix} \widetilde{\boldsymbol{R}}_{ij} & \widetilde{\boldsymbol{t}}_{ij} \\ \boldsymbol{0}_{1\times 2} & 1 \end{bmatrix} \in \text{SE}(2)$, but also (estimated) precisions, τ_{ij}, κ_{ij} of $\widetilde{\boldsymbol{t}}_{ij}$ and $\widetilde{\boldsymbol{R}}_{ij}$, respectively.

8. Machine Specifications

All our experiments were done on a machine with 2 Intel(R) Xeon(R) CPU E5-2630 v4, 2.20GHz processors, each with 10 cores (20 hyperthreads), and 256GB RAM.

9. Lie Groups and Lie Algebra

Definition 1 (Matrix Lie groups) *Let n be a fixed positive integer. A* matrix Lie group *is a set G of $n \times n$ matrices together with the binary operation of matrix product on G (that is, the domain is $G \times G$) such that:*

(G1) $\boldsymbol{I}_{n\times n} \in G$;

(G2) $\boldsymbol{A}, \boldsymbol{B} \in G \Rightarrow \boldsymbol{A}\boldsymbol{B} \in G$;

(G3) $\boldsymbol{A} \in G \Rightarrow \boldsymbol{A}$ *is an invertible matrix,* $\boldsymbol{A}^{-1} \in G$.

Matrix Lie groups are also called *matrix groups*, the terms being identical. It is possible to use a similar definition for matrix Lie groups whose elements take complex values; in this work, however, the discussion is restricted to real-valued matrix Lie groups. Let G satisfy the conditions in Definition 1. Basic linear algebra shows that:

(G4) $\boldsymbol{A} \in G \Rightarrow \boldsymbol{A}\boldsymbol{I}_{n\times n} = \boldsymbol{I}_{n\times n}\boldsymbol{A} = \boldsymbol{A}$

(G5) $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C} \in G \Rightarrow (\boldsymbol{A}\boldsymbol{B})\boldsymbol{C} = \boldsymbol{A}(\boldsymbol{B}\boldsymbol{C}) = \boldsymbol{A}\boldsymbol{B}\boldsymbol{C}$

(G6) $\boldsymbol{A} \in G \Rightarrow \boldsymbol{A}\boldsymbol{A}^{-1} = \boldsymbol{A}^{-1}\boldsymbol{A} = \boldsymbol{I}_{n\times n}$.

We now discuss Lie algebras, confining the discussion to Lie algebras of real-valued matrix Lie groups.

Definition 2 (Lie algebra) *Let G be a real-valued matrix Lie group. Its Lie algebra, $\mathfrak{g}$, is given by the linear space*

$$\mathfrak{g} = \exp^{-1}(G), \tag{3}$$

(where exp *is the matrix exponential) and $[\cdot,\cdot] : \mathfrak{g} \times \mathfrak{g} \to \mathfrak{g}$, the* Lie bracket *of $\mathfrak{g}$, is given by $\boldsymbol{A}, \boldsymbol{B} \mapsto \boldsymbol{AB} - \boldsymbol{BA}$.*

Remark 1 *Note that the notation $\exp^{-1}(G)$ does not imply that the map $\exp : \mathfrak{g} \to G$ is invertible; rather, it is merely the standard set-theoretic notation for the preimage of the set G under the map $\exp : \mathfrak{g} \to G$, which means all those elements of $\mathbb{R}^{n\times n}$ such that when we exponentiate them, we end up in G. In other words, $\exp^{-1}(G) \triangleq \{A : \exp(A) \in G\}$. In particular, depending on G, the map $\exp : \mathfrak{g} \to G$ might not be surjective;* e.g.*, since it can be shown that $\det(\exp(A)) = e^{\mathrm{tr}(A)}$ (note the RHS is always positive), it follows that $\exp(A)$ always has a positive determinant. Consequently, if B has a negative determinant, then there does not exist an $A \in \mathbb{R}^{n\times n}$ such that $\exp(A) = B$. Moreover, depending on G, the map $\exp : \mathfrak{g} \to G$ might not be injective.*

All Lie groups in our paper are matrix Lie groups. Let $\mathcal{G}_{\text{align}}$ be a matrix Lie group whose elements act on $\mathbb{R}^2$. We refer to $\mathcal{G}_{\text{align}}$ as the *alignment group*. Let $\mathcal{G}_{\text{sync}}$ be a subgroup of $\mathcal{G}_{\text{align}}$, represented notationally as $\mathcal{G}_{\text{sync}} \leq \mathcal{G}_{\text{align}}$. We refer to $\mathcal{G}_{\text{sync}}$ as the *synchronization group*. Let $\mathfrak{g}_{\mathfrak{align}}$ and $\mathfrak{g}_{\mathfrak{sync}}$ denote the corresponding Lie algebras of $\mathcal{G}_{\text{align}}$ and $\mathcal{G}_{\text{sync}}$. The alignment group may be chosen to be Aff(2) (namely, the group of invertible affine transformations in 2D), while the synchronization group may be chosen to be SE(2) (namely, the group of special Euclidean transformations in 2D). In this work, we use $\mathcal{G}_{\text{align}} = \text{Aff}(2)$ and $\mathcal{G}_{\text{sync}} = \text{SE}(2)$. Thus, for concreteness and simplicity, we will focus on this case and avoid needless generality that will complicate the presentation. That said, it is useful to keep in mind that the formulation presented in this work can be discussed in more general terms as this enables a cleaner view that goes beyond certain specific details and opens the door for new research directions on how to generalize the method proposed here. For $\mathcal{G}_{\text{align}}$, it would be relatively easy to use a more flexible group than Aff(2). For example, homographies would be a natural choice. Empirically, however, we found that Aff(2) sufficed for producing good results in various settings. For $\mathcal{G}_{\text{sync}}$, it would be considerably harder to go beyond SE(2) as there are currently no available solvers that generalize SE-Sync.

The Lie algebras of Aff(2) and SE(2) are denoted by $\mathfrak{aff}(2)$ and $\mathfrak{se}(2)$. While $\mathfrak{aff}(2)$ and $\mathfrak{se}(2)$ are linear subspaces, the latter being a subspace of the former, Aff(2) and SE(2) are nonlinear (smooth) manifolds. Thus, dim(Aff(2)) and dim(SE(2)) are well defined despite of the nonlinearity [6, 7]. Their dimensions are dim(SE(2)) = 3 and dim(Aff(2))=6. The generic elements of $\mathfrak{aff}(2)$ and $\mathfrak{se}(2)$, respectively, have the forms

$$\begin{bmatrix} a & b & c \\ e & f & h \\ 0 & 0 & 0 \end{bmatrix} \in \mathfrak{aff}(2) \qquad \begin{bmatrix} 0 & b & c \\ -b & 0 & h \\ 0 & 0 & 0 \end{bmatrix} \in \mathfrak{se}(2) \subset \mathfrak{aff}(2)\,. \tag{4}$$

Let $\mathbf{vec} : \mathfrak{aff}(2) \to \mathbb{R}^6$ be *a* linear bijection between elements of $\mathfrak{aff}(2)$ and their representation as vectors. Reusing the symbol **vec**, let $\mathbf{vec} : \mathfrak{se}(2) \to \mathbb{R}^3$ denote a similar linear bijection which is the restriction of $\mathbf{vec} : \mathfrak{aff}(2) \to \mathbb{R}^6$ to $\mathfrak{se}(2)$. Particularly, we will use the following linear bijections:

$$(a,b,c,d,e,f) = \mathbf{vec}\begin{bmatrix} a & b & c \\ e & f & h \\ 0 & 0 & 0 \end{bmatrix} \qquad \mathbf{vec}^{-1}(a,b,c,d,e,f) = \begin{bmatrix} a & b & c \\ e & f & h \\ 0 & 0 & 0 \end{bmatrix} \tag{5}$$

$$(b,c,h) = \mathbf{vec}\begin{bmatrix} 0 & b & c \\ -b & 0 & h \\ 0 & 0 & 0 \end{bmatrix} \qquad \mathbf{vec}^{-1}(b,c,h) = \begin{bmatrix} 0 & b & c \\ -b & 0 & h \\ 0 & 0 & 0 \end{bmatrix} \tag{6}$$

Recall that a generic element of SO(n), the special orthogonal group in $\mathbb{R}^n$, is an n-by-n matrix $\boldsymbol{R}$ such that $\boldsymbol{R}\boldsymbol{R}^T = \boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{I}_{n\times n}$ and $\det \boldsymbol{R} = 1$. Elements of SO(n) are called rotation matrices. The generic elements of Aff(2) and SE(2), respectively, have the forms

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{t} \\ \boldsymbol{0}_{1\times 2} & 1 \end{bmatrix} \in \text{Aff}(2), \quad \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}_{1\times 2} & 1 \end{bmatrix} \in \text{SE}(2) \subset \text{Aff}(2)\,, \quad \det \boldsymbol{A} \neq 0,\ \boldsymbol{R} \in \text{SO}(2),\ \boldsymbol{t} \in \mathbb{R}^2\,. \tag{7}$$

The matrix exponential maps $\mathfrak{aff}(2)$ into Aff(2), and $\mathfrak{se}(2) \subset \mathfrak{aff}(2)$ onto SE(2) $\subset$ Aff(2).

Let $\boldsymbol{B} = \begin{bmatrix} \boldsymbol{b}_1 & \boldsymbol{b}_2 & \boldsymbol{b}_3 \end{bmatrix}$ be an orthonormal 6-by-3 matrix (*i.e.*, $\boldsymbol{B}^T\boldsymbol{B} = \boldsymbol{I}_{3\times 3}$) such that $(\mathbf{vec}^{-1}(\boldsymbol{b}_i))_{i=1}^3$ is an orthonormal basis of $\mathfrak{g}_{\mathfrak{sync}}$. An example is

$$\boldsymbol{B} = \begin{bmatrix} 0 & 0 & 0 \\ \sqrt{.5} & 0 & 0 \\ 0 & 0 & 1 \\ -\sqrt{0.5} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}\,. \tag{8}$$

It can be shown that the 3-rank 6-by-6 projection matrix, $\boldsymbol{B}\boldsymbol{B}^T$, does not depend on the choice of $\boldsymbol{B}$. Let $T \in \mathrm{Aff}(2)$. We define the distance between T and $\mathrm{SE}(2)$ (*i.e.*, a measure for how far T is from being a rigid-body transformation) in terms of the orthogonal projection of $\log(T)$ on $\mathfrak{se}(2)$:

$$T \in \mathrm{Aff}(2) \quad d(T, \mathrm{SE}(2)) = \left\| \boldsymbol{\theta} - \boldsymbol{B}\boldsymbol{B}^T\boldsymbol{\theta} \right\|_{\ell_2} \quad \text{where } \boldsymbol{\theta} = \mathbf{vec}^{-1}(\log(T)) \in \mathfrak{se}(2) \ . \tag{9}$$

It is this distance that is used in the regularization term in our paper.

References

[1] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. 6

[2] Emmanuel J Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *JACM*, pages 1–37, 2011. 4

[3] M.A. Fischler and R.C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, pages 381–395, 1981. 6

[4] Soren Hauberg, Aasa Feragen, and Michael J Black. Grassmann averages for scalable robust pca. In *CVPR*, pages 3810–3817, 2014. 3

[5] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *ICCV*, pages 2938–2946, 2015. 3

[6] J.M. Lee. *Introduction to smooth manifolds*. Springer Verlag, 2003. 7

[7] J.M. Lee. *Introduction to topological manifolds*, volume 202. Springer, 2010. 7

[8] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, pages 91–110, 2004. 6

[9] Brian E Moore, Chen Gao, and Raj Rao Nadakuditi. Panoramic robust pca for foreground–background separation on noisy, free-motion camera video. *IEEE Transactions on Computational Imaging*, pages 195–211, 2019. 3, 4

[10] Marius Muja and David G Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP*, page 2, 2009. 6

[11] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, pages 1–9, 2015. 5

[12] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *TPAMI*, pages 376–380, 1991. 6