Deformable Sprites for Unsupervised Video Decomposition

Supplementary Material

Parameterization of rigid transforms (Section 3.3). We parameterize the homography as a decomposed chain of transformations, as in [1]:

$$\mathbf{H} = \begin{bmatrix} s\mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{bmatrix} \mathbf{K} & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{v}^\top & v \end{bmatrix}, \quad (1)$$

where $\mathbf{R} \in SO(2)$ and $\mathbf{K}$ is an upper triangular with $\det \mathbf{K} = 1$. The parameters we optimize over are then $\eta = [s\mathbf{R}_{11}, s\mathbf{R}_{12}, \mathbf{t}_1, \mathbf{t}_2, \mathbf{K}_{11}, \mathbf{K}_{12}, \mathbf{v}_1, \mathbf{v}_2] \in \mathbb{R}^8$. We find this parameterization to be more stable, and to favor explaining the motion with rigid scaling and translation rather than perspective and skew effects.

B-spline details (Section 3.3, Equations 2 and 3). The 1D B-spline basis functions of degree d, $B_d^{\text{1d}}i(z) \in [0,1]^{d+1}$ are defined for over the unit interval $[0,1]$. We use the degree $d = 2$ vector basis function $B_2^{\text{1d}}(z) \in [0,1]^3$:

$$B_2^{\text{1d}}(z) = \left[\tfrac{(1-z)^2}{2}, \quad -z^2 + z + 0.5, \quad \tfrac{z^2}{2}\right]^\top. \quad (2)$$

In 2D, the degree $d = 2$ basis functions $B_2^{\text{2d}}(u,v)$ are matrices in $[0,1]^{3\times 3}$ defined for $u, v \in [0,1]$.

$$B_2^{\text{2d}}(u,v) = B_2^{\text{1d}}(u) \cdot B_2^{\text{1d}}(v)^\top. \quad (3)$$

For each interpolated query point, we blend the $d+1 = 3$ neighboring knot values (in each dimension) using the weights given by the respective basis functions.

Determining transform scales (Section 4). The transformations are optimized to be consistent with the optical flow in the coordinate space of the input images, but the scale factor $s^{(\ell)}$ between each layer's sprite and the input images is not well-determined. We determine the scale of each layer with the predicted masks after 5 epochs of training. For each frame t and layer ℓ, we compute the mean flow, $\mu_t^{(\ell)} \in \mathbb{R}^2$ as in Equation 8, and the bounding box coordinates, $a_t^{(\ell)}, b_t^{(\ell)} \in \mathbb{R}^2$, of the points $\{x;\ M_t^{(\ell)(x)} \geq 0.5\}$. The displacement of layer ℓ at time t is $\delta_t^{(\ell)} = \sum_{\tau=1}^{t} \mu_\tau^{(\ell)}$. We estimate the total area covered by each layer from the start frame as a bounding box with minimum coordinate $\alpha^{(\ell)}$ and maximum coordinate $\beta_t^{(\ell)}$

$$\alpha^{(\ell)} = \min_t \{a_t^{(\ell)} - \delta_t^{(\ell)}\}, \quad \beta^{(\ell)} = \max_t \{a_t^{(\ell)} - \delta_t^{(\ell)}\}. \quad (4)$$

We set the scale factor for layer ℓ to be $s^{(\ell)} = \frac{2}{\beta^{(\ell)} - \alpha^{(\ell)}}$. We also update the rigid translation of layer ℓ at time t to be $s^{(\ell)} \cdot (\delta_t^{(\ell)} - \alpha^{(\ell)}) - 1$. $\alpha^{(\ell)}$ and $\beta^{(\ell)}$ are thus mapped to $(-1,-1)$ and $(1,1)$, respectively.

Scale invariance in transform loss (Section 4, Equation 10). We divide the transform loss by the scale of the transform in Equation 10 to prevent the transforms from becoming degenerate (scaled to 0). Our loss results in transforms with more stable scales. Given the homography parameterization of $\eta = [s\mathbf{R}_{11}, s\mathbf{R}_{12}, \mathbf{t}_1, \mathbf{t}_2, \mathbf{K}_{11}, \mathbf{K}_{12}, \mathbf{v}_1, \mathbf{v}_2] \in \mathbb{R}^8$, we use $s = \sqrt{\eta_1^2 + \eta_2^2}$ as the scale.

Rejection of poorly-estimated fundamental matrices (Section 4). We use the optical flow vectors of all pixels in the input frame to estimate the fundamental matrix required for $\mathcal{L}_{\text{static}}$. Because least median of squares regression can handle outliers consisting up to 50% of the data, the estimated fundamental matrix captures the static geometry in the cases when background pixels comprise at least half of the frame. However, we will encounter outlier percentages greater than 50% when they do not.

To distinguish the two cases, we observe a difference in the distribution of residuals in the two cases. Fitting a fundamental matrix to a moving object in its local reference frame will result in a fairly even residual distribution: we observe a modest residual on all points. In comparison, fitting a fundamental matrix to a static element with moving outliers will result in a bimodal residual distribution: the static points have low residuals, while outlier moving points have high residuals. We implement this distinction by rejecting estimates with a median error greater than 1 pixel.

Computing occlusions for optical flow consistency (Section 4). We enforce optical flow consistency, using the losses in equations 10 and 11, for all pixels that are not occluded. We compute occlusion using the method described in [4]. We consider all points x in I_t. Under the forward flow field $F_{t\to t+1}$, x is taken to $x' = F_{t\to t+1}(x)$. Under the backward flow field $F_{t+1\to t}$, x' is taken to $x'' = F_{t+1\to t}(x')$. We consider all points x such that

$$\|x'' - x\|^2 > 0.01 \cdot \left(F_{t\to t+1}(x)^2 + F_{t+1\to t}(x')^2\right) + 1.5 \quad (5)$$

to be occluded, and the corresponding points x'' to be the occluders. We call $\omega(I_t)$ the set of such occluded points x in I_t; we call $\Omega(I_t)$ the set of such occluded points x'' in I_t.

Network architectures. We implement our model in PyTorch [2]. We use a standard UNet architecture [3] for both the grouping model and the texture generator. We provide the architecture we used for the results reported in the paper in Tables 1 and 2. All convolutions are 3×3 with a stride and padding of 1. All layers use a ReLU activation function. All upsampling and pooling layers use a scale factor of 2.

Optimization. We optimize our losses with a schedule. Initially the relative weight of the grouping loss is greater than those of the transform and recon losses. After 5 epochs, we reduce the grouping loss weight to 0.1 of the original weight. We use $\lambda_{\text{recon}} = 1$, $\lambda_{\text{group}} = 1$, $\lambda_{\text{transform}} = 1$. We use the Adam optimizer with learning rate 10^{-3}. We train each video with batch size 8 on an NVIDIA RTX 2080, for videos resized to 240×426 spatial resolution, for a total of 200 epochs. For a video with 80 frames of this size, optimization takes around 30 minutes.

	layer	in channels	out channels
1	conv, BN, avg pool	16	32
2	conv, BN, avg pool	32	64
3	conv, BN, avg pool	64	64
4	conv, BN, UP	64	64
5	skip3, conv, BN, UP	128	32
6	skip2, conv, BN, UP	64	16

Table 1. Network architecture for the grouping network. BN is batch norm, UP is bilinear upsample.

	layer	in channels	out channels
1	conv, BN, avg pool	16	32
2	conv, BN, avg pool	32	64
3	conv, BN, avg pool	64	128
4	conv, BN, avg pool	128	128
5	conv, BN, UP	128	128
6	skip4, BN, UP	256	64
7	skip3, conv, BN, UP	128	32
8	skip2, conv, BN, UP	64	16

Table 2. Network architecture for the texture generator. BN is batch norm, UP is bilinear upsample.

References

[1] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. 1

[2] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 1

[3] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015. 1

[4] Narayanan Sundaram, Thomas Brox, and Kurt Keutzer. Dense point trajectories by gpu-accelerated large displacement optical flow. In *European conference on computer vision*, pages 438–451. Springer, 2010. 1