We present the following items in the Appendix:

- The thin-plate splines warp computation (Section A) and its inverse (Section B)
- The detailed formulation for the semantics-aware refinement step (Section C)
- The layer occlusion model (Section D)
- A stochastic extension of our method (Section E)
- Detailed architectural choices for each of WALDO's modules (Section F)
- More qualitative samples on nonrigid scenes (Section G)
- The influence of the choice of the pretrained segmentation and optical flow models (Section H)
- An ablation study of our inpainting strategy (Section I)
- Further information about our implementation and the overall training process (Section J)
- A statement about the societal impact of this project (Section K)
- A qualitative study of our approach (Section L)

A. Thin-plate splines warp computation

For clarity, vectors (or points) are denoted by bold face lower case letters, and matrices are denoted by bold face upper case letters throughout this presentation. Let $\mathbf{p} = (x, y, 1)^\intercal$ be a point in homogeneous coordinates, and $\mathbf{C_1}$ and $\mathbf{C_2}$ be two sets of such points in $\mathbb{R}^{3\times L}$, which we refer to as *control points*, with L the (fixed) number of points in each set. The thin-plate splines (TPS) [9] transformation which maps $\mathbf{p}$ onto $\mathbf{p_{12}}$ writes:

$$\mathbf{p_{12}} = \mathbf{T}\mathbf{p} + \mathbf{U}\phi(\mathbf{p}, \mathbf{C_1}), \tag{5}$$

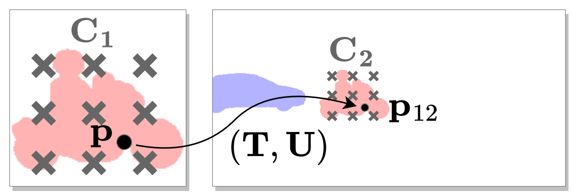

Figure A1. Thin-plate splines transformation mapping points from $\mathbf{C_1}$ onto $\mathbf{C_2}$ applied to an arbitrary point $\mathbf{p}$ using parameters ($\mathbf{T}$, $\mathbf{U}$).

where $\phi(\mathbf{p}, \mathbf{C_1}) = [k(\mathbf{p}, \mathbf{p_1})]_{\mathbf{p_1}\in\mathbf{C_1}}$ is a L-dimensional vector, and ($\mathbf{T}$, $\mathbf{U}$) are TPS parameters in the form of matrices of dimension 3×3 and $3 \times L$ respectively. The transformation is decomposed into a global affine one (through $\mathbf{T}$) and a local non-affine one (through $\mathbf{U}$ and ϕ). The kernel function k is defined by $k : (\mathbf{p}, \mathbf{q}) \to \|\mathbf{p} - \mathbf{q}\|_2^2 \log \|\mathbf{p} - \mathbf{q}\|_2$ and materializes the fact that the TPS transformation minimizes the bending energy. The $3(3 + L)$ TPS parameters are found using the constraint that points from $\mathbf{C_1}$ should be mapped onto points from $\mathbf{C_2}$ using (5). This yields a system of $3L$ equations to which Bookstein adds 9 extra ones, as explained in [9], which we formulate as:

$$\mathbf{C_1}\mathbf{U}^\intercal = \mathbf{0_{3\times 3}}. \tag{6}$$

By applying (5) on pairs of points from $\mathbf{C_1}$ and $\mathbf{C_2}$, and using extra constraints (6), we obtain the TPS parameters ($\mathbf{T}$, $\mathbf{U}$):

$$\begin{bmatrix}\mathbf{T}^\intercal \\ \mathbf{U}^\intercal\end{bmatrix} = \mathbf{\Delta}(\mathbf{C_1})^{-1}\begin{bmatrix}\mathbf{C_2}^\intercal \\ \mathbf{0_{3\times 3}}\end{bmatrix}, \qquad \mathbf{\Delta}(\mathbf{C_1}) = \begin{bmatrix}\mathbf{C_1}^\intercal & \mathbf{\Phi}(\mathbf{C_1}, \mathbf{C_1}) \\ \mathbf{0_{3\times 3}} & \mathbf{C_1}\end{bmatrix}, \tag{7}$$

where $\mathbf{\Phi}(\mathbf{C_1}, \mathbf{C_1})$ is a $L \times L$ matrix obtained by stacking $\phi(\mathbf{p_1}, \mathbf{C_1})$ for all points $\mathbf{p_1}$ in $\mathbf{C_1}$.

Hence, computing the TPS parameters amounts to inverting a $(L + 3) \times (L + 3)$ matrix, $\mathbf{\Delta}(\mathbf{C_1})$. Since doing that for each new transformation is impractical, we use a fixed grid of control points associated with each layer for $\mathbf{C_1}$, as a proxy to compute the flow between pair of frames. By fixing the value of $\mathbf{C_1}$, the corresponding matrix inversion is done only once for the whole training; and we are still able to model different deformations by setting $\mathbf{C_2}$ to different values. We note that, once $\mathbf{\Delta}(\mathbf{C_1})^{-1}$ has been computed, sampling the warp w associated with a new $\mathbf{C_2}$ is done by finding the corresponding TPS parameterization using (5), and by sampling the deformations for each point $\mathbf{p}$ in a dense grid $[\![1, H]\!] \times [\![1, W]\!]$ using (7), where the spatial resolution $H \times W$ can be arbitrarily large. Moreover, this process is fully differentiable with respect to $\mathbf{C_2}$ and each point $\mathbf{p}$ as it involves simple algebraic operations. In the main paper, $\mathbf{C_1}$ and $\mathbf{C_2}$ correspond to the regular grid of control points g^i, associated with layer i, and its deformation p^i_t at time step t respectively. The warp w^i_t, also in the main paper, corresponds to the inverse transformation to the one presented here, and associates with every point $\mathbf{p_{12}}$ corresponding to a pixel of the image (right side in Figure A1) the corresponding point $\mathbf{p}$ in object coordinates (left side). However, computing the inverse transformation cannot be achieved by simply switching the roles of $\mathbf{C_1}$ and $\mathbf{C_2}$, since $\mathbf{C_1}$ have to be kept constant, which is why we resort to warp inversion whose simple formulation is detailed in the next section.

B. Inverse warp computation

Let w be a warp in $\mathbb{R}^{2\times H\times W}$ (where $H\times W$ is a given spatial resolution), which we could also consider as a mapping from $\mathbb{R}^2$ to $\mathbb{R}^2$, where $w(p)$ is the geometric transformation of a point p sampled on the grid $[\![1,H]\!]\times[\![1,W]\!]$. Such a mapping is not surjective with respect to the grid, that is, all the cells in the grid are not necessarily reached by w. As a result, we approximate the inverse warp w^{-1} by the pixel-accurate inversion in cells for which such a direct mapping exists and use interpolation for filling others, starting from the cardinal neighbours of already inverted cells, and iteratively filling the remaining ones. The warp w^{-1} may need to point out of the grid for some cells, *e.g.*, when an object moves out of the frame. However, this cannot be extracted from w which is only defined on the grid. To avoid interpolating wrong values in these cells, we only invert w in those which are close (as per a given threshold) to the ones initially reached by w, and make others point to an arbitrary position outside of the grid layout. Finally, the same reasons which justify that training errors can backpropagate through a spatial transformer [42] also apply here.

C. Semantics-aware refinement

We now describe in more details the semantics-refinement step introduced in the main body of the paper. Let m_t be a soft mask in $[0,1]^{H\times W}$ at a given time step t in $[\![1,T]\!]$, and c be a soft class assignment in $[0,1]^C$, both of them predicted for the same layer, and let s_t be an input (soft) semantic map in $[0,1]^{C\times H\times W}$ also associated with time step t. We denote by $\bar{c}$, the mean semantic class on the spatio-temporal tube defined by the masks, which, like c, is a vector in $[0,1]^C$, and writes:

$$\bar{c} = \sum_{t,h,w}[m_t \odot \mathcal{F}(s_t,c)]_{(h,w)} / \sum_{t,h,w}[m_t]_{(h,w)}, \tag{8}$$

where $\odot$ is the element-wise product, and $\mathcal{F}$ is a class-filtering function parameterized by c and applied to s_t, that is, at a given spatial location (h,w), the result of kipping dominant classes as per c in s_t:

$$[\mathcal{F}(s_t,c)]_{(h,w)} = \frac{1}{1+k_c}\sum_j \langle [s_t]_{(h,w)}, c+k_c\rangle, \tag{9}$$

with k_c a constant which defines the degree at which low scoring classes will be filtered out. One can set $k_c=0$ for full effect and greater values for filtering less. In practice, we fix the value of k_c to 0.1. Each mask m_t is updated by computing the L_1 distance between the semantic map s_t and the mean class $\bar{c}$ at every location (h,w):

$$[m_t]_{(h,w)} = (1-\|[s_t]_{(h,w)}-\bar{c}\|_1)[m_t]_{(h,w)}. \tag{10}$$

D. Layer occlusion model

Our occlusion model is rather standard, but we include it here for completeness. Ordering scores o_t are used to filter non-visible parts in a layer i due to the presence of another layer j on top of it (*i.e.*, when $o_t^i \ll o_t^j$). The transparency m_t^i of layer i at time step t is updated as follows:

$$m_t^i = m_t^i \odot \prod_{j\neq i}\Big(\mathbf{1} - \frac{o_t^j}{o_t^i+o_t^j} m_t^j\Big), \tag{11}$$

where $\odot$ is the element-wise product whose right-hand side component has values between 0 (occluded) and 1 (visible).

E. Stochastic extension of WALDO

In our original description, motions produced by WALDO are purely deterministic and they converge towards the mean of all possible future trajectories given the past. Although it is sufficient for short-term prediction, one could be interested in modeling different behaviours for longer future horizons. For this reason, we propose an extension of WALDO to account for the intrinsically uncertain nature of the future by allowing multiple predictions.

Our approach, illustrated in Figure E1, builds upon generative adversarial networks (GANs) [30]. We consider the future layer prediction module as a generator which computes the future positions of control points given ones from the past. This generator is trained jointly with a discriminator which classifies trajectories as *real* or *fake*. Both are playing a minimax game, where the discriminator is taught to correctly distinguish real from fake trajectories, while the generator tries to fool

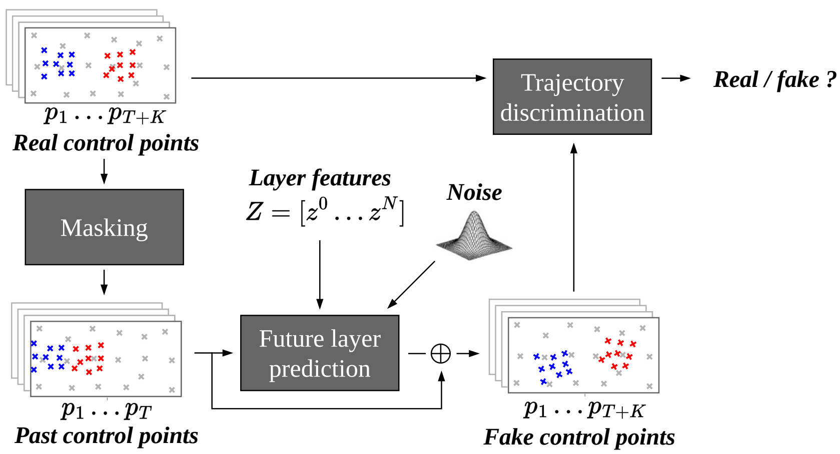

Figure E1. **Stochastic future prediction.** We extend the future layer prediction module of WALDO to allow the prediction of multiple futures. The module itself is not changed, except that it now uses noise (as input and in attention modules). The major difference is that a *trajectory discrimination* module is used at train time to assist the model in producing a realistic control point trajectory instead of the mean trajectory. At inference, only the future layer prediction module is kept. See text for details.

the discriminator. Through this process we expect synthetic trajectories to gradually improve in realism and to capture multiple modes from the underlying data distribution.

We implement the discriminator with a transformer and use the WGAN loss introduced in [3] for training. In addition to this loss, we find that keeping the initial reconstruction term ($\mathcal{L}_p$) is important for the stability of training. Moreover, we normalize gradients from both supervision signals (reconstruction and adversarial) so that they have matching contributions.

F. Detailed architectures

We detail inner operations of each of WALDO's modules (for the dimensions used on Cityscapes [17]).

F.1. Layered video decomposition

Input encoding. Semantic and flow maps (s and f) are concatenated and go through a series of 3×3 convolutional layers (Conv) with padding of 1 and stride of 2, each followed by layer norm (LN) and a GELU activation [35] (Act) to form downscaled features y for a given time step.

Table F1. Input encoding.

Stage	Operation	On	Output size
s	-	-	$20 \times 128 \times 256$
f	-	-	$2 \times 128 \times 256$
1	Concat.	s, f	$22 \times 128 \times 256$
Conv_1	$[3 \times 3]$	1	$64 \times 64 \times 128$
LN_1	Norm.	1	$64 \times 64 \times 128$
Act_1	GELU	1	$64 \times 64 \times 128$
Conv_2	$[3 \times 3]$	1	$128 \times 32 \times 64$
LN_2	Norm.	1	$128 \times 32 \times 64$
Act_2	GELU	1	$128 \times 32 \times 64$
Conv_3	$[3 \times 3]$	1	$256 \times 16 \times 32$
LN_3	Norm.	1	$256 \times 16 \times 32$
Act_3	GELU	1	$256 \times 16 \times 32$
Conv_4	$[3 \times 3]$	1	$512 \times 8 \times 16$
y	-	1	$512 \times 8 \times 16$

Layer feature extraction. We combine features encoded from different time steps into (Y), and sum them with different embeddings corresponding to their temporal ordering and spatial positioning (T and P) to form input (1). Layer features Z_{obj} and Z_{bg} are computed from input (2), which is the concatenation of object and background embeddings (2a and 2b), themselves expressed as the sum of layer ordering and spatial positioning embeddings (S, L, O and B). The next operations consists in layer norm (LN), self-attention (Att) to update (2) by computing queries, keys and values for (1) but only keys and values for (2), and multi-layer perceptrons (MLP) with GELU activations [35].

Control point positioning. We apply similar operations to predict control points (p_{obj}, p_{bg}) for object and background layers at a given time step from associated features y. A key difference with the previous module is that this one is time-independent, and that inputs (1) and (2) play the same role in the transformer blocks. A fully-connected layer (FC) outputs the 3D position of each control point in each of the $16 + 1$ layers.

Object masking. This module predicts an alpha-transparency mask a for an object from its associated features z. It is the reverse process compared to the input encoding module, that is, we replace convolutions with transposed ones for progressively upscaling feature maps z, and we decrease the size of features at each step instead of increasing it.

Object classification. This module predicts for each object represented by z a soft class assignment c, by first averaging z over its spatial dimensions, and then applying layer norm (LN), a fully-connected layer (FC), and a softmax activation (Act) to output c as a categorical distribution over classes.

F.2. Future layer prediction

Past encoding. Past control points corresponding to objects (resp. the background) are transformed into vectors, each of them paired with one layer and one time step, using a fully-connected layer FC1 (resp. FC2). We apply a pooling operation to layer representations (Z_{obg} and Z_{bg}) to reduce them to a single vector representing each layer. All these vectors are concatenated, summed with the suitable embeddings (T and P), and passed through two vanilla transformer blocks to produce encoded features E.

Future decoding. Future control points are obtained by initializing future representations (2) with some embeddings (T and P), and then alternating between transformer blocks with self-attention on future representations (2), and cross-attention from past to future ones (1 and 2).

F.3. Warping, inpainting and fusion

The final component of our approach is a U-Net [68] composed of 6 downscaling layers and 6 upscaling ones, with as many skip connections between the two branches. Each downscaling (resp. upscaling) layer divides (resp. multiplies) by 2 its input resolution using a 3×3 convolution (resp. transposed convolution) with a stride of 2, and multiplies (resp. divides) by 2 the size of features so that intermediate features (between the two branches) are of size 512. This module outputs RGB values to update certain regions of an image (filling missing background or object parts, adapting light effects or shadows in other parts), a mask indicating these regions, a score (of confidence) at each pixel location to allow fusing multiple views corresponding to the same image.

Table F2. Layer feature extraction.

Stage	Operation	On	Output size
Y	-	-	$4\times512\times8\times16$
(T)	Embed.	-	$4\times512\times1\times1$
(P)	Embed.	-	$1\times512\times8\times16$
1	Sum/Reshape	Y, T, P	512×512
(S)	Embed.	-	$1\times512\times4\times4$
(L)	Embed.	-	$1\times512\times8\times16$
(O)	Embed.	-	$16\times512\times1\times1$
(B)	Embed.	-	$1\times512\times1\times1$
2a	Sum/Reshape	S, O	256×512
2b	Sum/Reshape	L, B	128×512
2	Concat.	2a, 2b	384×512
LN_1	Norm.	1	512×512
LN_2	Norm.	2	384×512
Att_1	$[512]\times3$	2/1	384×512
LN_3	Norm.	2	384×512
MLP_1	$[2048, 512]$	2	384×512
LN_4	Norm.	2	384×512
Att_2	$[512]\times3$	2/1	384×512
LN_5	Norm.	2	384×512
MLP_2	$[2048, 512]$	2	384×512
Z_{obj}	Split/Reshape	2	$16\times512\times4\times4$
Z_{bg}	Split/Reshape	2	$1\times512\times8\times16$

Table F3. Control point positioning.

Stage	Operation	On	Output size
y	-	-	$512\times8\times16$
(P)	Embed.	-	$512\times8\times16$
1	Sum/Reshape	y, P	128×512
Z_{obj}	-	-	$16\times512\times4\times4$
Z_{bg}	-	-	$1\times512\times8\times16$
(S)	Embed.	-	$1\times512\times4\times4$
(L)	Embed.	-	$1\times512\times8\times16$
(O)	Embed.	-	$16\times512\times1\times1$
(B)	Embed.	-	$1\times512\times1\times1$
2a	Sum/Reshape	Z_{obj}, S, O	$16\times512\times4\times4$
2b	Sum/Reshape	Z_{bg}, L, B	$1\times512\times8\times16$
2	Concat.	2a, 2b	384×512
3	Concat.	1, 2	512×512
LN_1	Norm.	3	512×512
Att_1	$[512]\times3$	3	512×512
LN_2	Norm.	3	512×512
MLP_1	$[2048, 512]$	3	512×512
LN_3	Norm.	3	512×512
Att_2	$[512]\times3$	3	512×512
LN_4	Norm.	3	512×512
MLP_2	$[2048, 512]$	3	512×512
2	Split	3	384×512
FC	[3]	2	384×3
p_{obj}	Split/Reshape	2	$16\times3\times4\times4$
p_{bg}	Split/Reshape	2	$1\times3\times8\times16$

Table F4. Object masking.

Stage	Operation	On	Output size
z	-	-	$512\times4\times4$
1	-	z	$512\times4\times4$
LN_1	Norm.	1	$512\times4\times4$
TConv_1	$[3\times3]$	1	$256\times8\times8$
LN_2	Norm.	1	$256\times8\times8$
Act_1	GELU	1	$256\times8\times8$
TConv_2	$[3\times3]$	1	$128\times16\times16$
LN_3	Norm.	1	$128\times16\times16$
Act_2	GELU	1	$128\times16\times16$
TConv_3	$[3\times3]$	1	$64\times32\times32$
LN_4	Norm.	1	$64\times32\times32$
Act_3	GELU	1	$64\times32\times32$
TConv_4	$[3\times3]$	1	$1\times64\times64$
Act_4	Sigmoid	1	$1\times64\times64$
a	-	1	$1\times64\times64$

Table F5. Object classification.

Stage	Operation	On	Output size
z	-	-	$512\times4\times4$
1	Spatial mean	z	512
LN	Norm.	1	512
FC	[20]	1	20
Act	Softmax	1	20
c	-	1	20

Table F6. Past encoding.

Stage	Operation	On	Output size
P_{obj}	-	-	4×16×3×4×4
P_{bg}	-	-	4×1×3×8×16
1a	Reshape	P_{obj}	$4\times16\times48$
1b	Reshape	P_{bg}	$4\times1\times128$
FC1	[512]	1a	$4\times16\times512$
FC2	[512]	1b	$4\times1\times512$
Z_{obj}	-	-	16×512×4×4
Z_{bg}	-	-	1×512×8×16
2a	Pool/Reshape	Z_{obj}	$1\times16\times512$
2b	Pool/Reshape	Z_{bg}	$1\times1\times512$
3	Concat.	1a, 1b, 2a, 2b	$5\times17\times512$
(T)	Embed.	-	$5\times1\times512$
(P)	Embed.	-	$1\times17\times512$
3	Sum/Reshape	3, T, P	85×512
LN_1	Norm.	3	85×512
Att_1	$[512]\times3$	3	85×512
LN_2	Norm.	3	85×512
MLP_1	$[2048, 512]$	3	85×512
LN_3	Norm.	3	85×512
Att_2	$[512]\times3$	3	85×512
LN_4	Norm.	3	85×512
MLP_2	$[2048, 512]$	3	85×512
E	Reshape	3	$5\times17\times512$

Table F7. Future decoding.

Stage	Operation	On	Output size
E	-	-	$5\times17\times512$
1	Reshape	E	85×512
(T)	Embed.	-	$10\times1\times512$
(P)	Embed.	-	$1\times17\times512$
2	Sum/Reshape	T, P	170×512
LN_1	Norm.	1	170×512
LN_2	Norm.	2	170×512
Att_1	$[512]\times3$	2	170×512
LN_3	Norm.	2	170×512
MLP_1	$[2048, 512]$	2	170×512
LN_4	Norm.	2	170×512
Att_2	$[512]\times3$	2/1	170×512
LN_5	Norm.	2	170×512
MLP_2	$[2048, 512]$	2	170×512
LN_6	Norm.	2	170×512
Att_3	$[512]\times3$	2	170×512
LN_7	Norm.	2	170×512
MLP_3	$[2048, 512]$	2	170×512
LN_8	Norm.	2	170×512
Att_4	$[512]\times3$	2/1	170×512
LN_9	Norm.	2	170×512
MLP_4	$[2048, 512]$	2	170×512
LN_{10}	Norm.	2	170×512
2a	Split	2	160×512
2b	Split	2	10×512
FC1	[48]	2a	160×48
FC2	[384]	2b	10×384
P_{obj}	Reshape	2a	10×16×3×4×4
P_{bg}	Reshape	2b	10×1×3×8×16

G. Additional visual results on nonrigid scenes

$T{+}1$ $T{+}10$ Warp $T{+}1$ $T{+}10$ Warp

Figure G2. Future prediction from $T{=}4$ frames on Taichi-HD (256×256). Nonrigid motions can be visualized by the associated warps, predicted from the control points between T and $T{+}10$ (colors represent different directions).

We further illustrate (Figure G2) that the motion representation proposed in WALDO allows us to represent nonrigid object deformations by training on Taichi-HD [73] dataset. Our approach can handle complex human motions such as leaning forward / backward, moving one leg while keeping the other on the ground, raising individual arms...

H. Influence of the choice of the pretrained segmentation and optical flow models

The use of pretrained networks in prior works vary widely according to the level of information required by each method, and depending on what was available at the time to extract these information. We summarize the differences in Table H1.

Table H1. Pretrained models used by different methods.

Method	Optical flow estimation	Semantic segmentation	Instance segmentation	Depth estimation	Object tracking	Video Frame Interpolation
VPVFI [100]	RAFT [77]	-	-	-	-	RIFE [39]
VPCL [29]	RAFT [77]	-	-	-	-	-
Vid2vid [90]	FlowNet2 [40]	DeepLabV3 [14]	-	-	-	-
OMP [99]	PWCNet [76]	VPLR [113]	UPSNet [102]	GeoNet [107]	SiamRPN++ [50]	-
SADM [7]	PWCNet [76]	DeepLabV3 [14]	-	-	-	-
WALDO	RAFT [77]	DeepLabV3 [14]	-	-	-	-

We thus evaluate the influence of the choice of the pretrained segmentation and optical flow models to WALDO's performance. Results on Cityscapes and KITTI test set, obtained by substituting segmentation model DeepLabV3 [14] with MobileNetV2 [70] or ViT-Adapter [15], and optical flow model RAFT [77] with PWCNet [76], are presented in Table H2.

Table H2. Ablation studies of optical flow estimation and semantic segmentation methods on the Cityscapes and KITTI test sets. Like in the main paper, we compute multi-scale SSIM ($\times 10^3$) and LPIPS ($\times 10^3$) for the k^{th} frame and report the average for k in $[\![1, K]\!]$.

Flow		Segmentation			(a) Cityscapes $K=1$		$K=5$		$K=10$		(b) KITTI $K=1$		$K=3$		$K=5$	
[76]	[77]	[70]	[14]	[15]	SSIM ↑	LPIPS ↓	SSIM ↑	LPIPS ↓	SSIM ↑	LPIPS ↓	SSIM ↑	LPIPS ↓	SSIM ↑	LPIPS ↓	SSIM ↑	LPIPS ↓
✓			✓		947	062	836	122	753	175	856	116	760	171	697	214
	✓	✓			954	055	849	111	768	165	859	112	756	166	692	209
	✓		✓		**957**	**049**	**854**	**105**	**771**	**158**	**867**	**108**	766	**163**	702	206
	✓			✓	**957**	050	853	**105**	770	159	866	109	**767**	**163**	**703**	**205**

Despite PWCNet [76] having approximately twice the average end-point-error of RAFT [77], using it instead of RAFT only results in a small performance drop for video prediction with WALDO. This is in line with the conclusions of Geng *et al.* in [29] who conducted a similar comparison. Moreover, replacing DeepLabV3 [14] with MobileNetV2 [70], with respective segmentation performance of 81 and 75 in terms of test set mIoU on Cityscapes, yields only little loss of video prediction quality. Conversely, using a more advanced segmentation model like ViT-Adapter [15], with 85 test set mIoU on Cityscapes, does not change the results substantially. Our interpretation is that the segmentation quality is not the limiting factor for WALDO's performance once it has reached a sufficient level. To conclude, WALDO is quite robust to the choice of the pretrained segmentation and optical flow models.

I. Ablation study of the inpainting strategy

Table I1. Ablation of our inpainting strategy on the Cityscapes test set.

Adversarial inpainting [51]	Temporal consistency	$K=1$		$K=5$		$K=10$		
		SSIM ↑	LPIPS ↓	SSIM ↑	LPIPS ↓	SSIM ↑	LPIPS ↓	FVD ↓
		957	**049**	853	**105**	770	**158**	061
✓		**957**	**049**	**854**	**105**	**771**	**158**	<u>057</u>
✓	✓	**957**	**049**	**854**	**105**	**771**	**158**	**055**

Figure I3. Visual ablation of our inpainting strategy. We start off with a method that fills in empty regions with the sole objective to minimize the reconstruction error (∅), we then propose to leverage an off-the-shelf adversarial inpainting (adv. inp.) method [51] to improve realism, and we finally illustrate our full strategy where we use the predicted flow to ensure the temporal consistency (temp. const.) of inpainted regions. For clarity, these regions are indicated in the last row. Please zoom in for details.

Given that we use an off-the-shelf inpainting method [51] trained on external data [111], we assess the impact of the use of such a model in our approach on quantitative measurements.

Results are presented in Table I1 and show that although gains in SSIM and FVD are possible, these gains remain small. In addition, no perceptible change is observed on LPIPS. Our temporal consistency strategy, which consists in filling frames one by one and using the predicted flow to propagate new contents into subsequent frames, allow small extra gains on the FVD metric.

The benefits from our inpainting strategy are more visible in qualitative samples presented in Figure I3. Without adversarial inpainting (∅), empty image regions are filled using a model trained to minimize the reconstruction error. We observe that this results in blurry image parts with important artefacts. Using adversarial inpainting produces much more realistic images when considering frames individually, but filling each of them independently is still not very natural (best viewed in the videos included in the project webpage). Our approach for producing temporally-consistent outputs is able to solve this issue. For example, in the left-most sample sequence of the third row of Figure I3, we see that inpainted image parts match between different time steps although the camera is moving.

J. Further implementation details

We follow [99] and group semantic classes which form a consistent entity together, *e.g.*, riders with their bicycle, traffic lights/signs with the poles, which allows us to represent those within a single object layer. We use horizontal flips, cropping, and color jittering as data augmentation. Although we encounter signs of over-fitting in some experiments, validation curves do not increase during training, nor after convergence. So best model selection is not necessary, and we always save the last checkpoint. We find that a good initialization of object regions helps the layer decomposition module to reach a better optimum and to converge faster. Hence, we add a warmup period during which the module is trained without flow reconstruction ($\lambda_f = 0$), and then progressively increase the associated parameter during training ($\lambda_f > 0$) once we start having good object proposals. Without doing so, the module tends to rely on the background layer alone to reconstruct the scene motion, which, as a result, leads to an under-use of the object layers. We also find that removing data-augmentation in the last few epochs when training the warping, inpainting and fusing module slightly improves the performance at inference time.

K. Societal impact

The total cost for this project, including architecture and hyperparameter search, training, testing and comparisons with baselines has been around 25K GPU hours, with an associated environmental cost of course. On the other hand, we strive to minimize this cost by decomposing video prediction into efficient lightweight modules, and our approach will hopefully contribute to eventually improve the safety of autonomous vehicles, by, say, predicting the motion of nearby agents.

L. Qualitative study of WALDO

In this section, we provide qualitative samples for each of the three modules which compose WALDO.

Layered video decomposition. We illustrate in Figure L1 our strategy to decompose videos into layers as a way to build inter-frame connections using a compact representation of motion from which we recover the dense scene flow.

We observe that objects are predicted in regions which match our pseudo ground truth, constructed from input segmentation and flow maps, even in difficult regions like the poles. Although they share the same semantic class, the three cars in the left-most example in Figure L1 are correctly segmented into different objects. Still, it may happen that multiple objects are merged into the same layer, or that an object is over-segmented into multiple layers (like the ego vehicle in these examples). This is due, in part, to the limitations of our approach which indicates regions of interest for the objects, but does not impose how they should be split among the different layers. When computing video decompositions, we also position a set of control points associated with each layer. The delta of control points between pairs of time steps produces sparse motion vectors for the background and the objects. We show that, although we use a small number of points, we are able to accurately recover the dense scene flow using TPS transformations, and that motion discontinuities occur at layer boundaries as expected.

Future layer prediction. We compare in Figure L2 the scene dynamics extracted from our decomposition strategy to the one inferred via the future layer prediction module.

We accurately reconstruct complex motions under various scenarios: when the background is static, moves towards the camera due to the ego motion, or sideways when the car is turning; in the presence of different kinds of objects such as trucks, cars, bikes or various road elements; and whether these objects move in the same direction or not.

Warping, fusion and inpainting. We illustrate in Figure L3 how future frames are finally synthesized.

Warping past frames to obtain future ones is not enough, as some regions may not be recovered from the past. In particular, we see that shadows may not always be consistent with the new position of objects, *e.g.*, for the vehicle in the bottom-right example in Figure L3. Our fusion and inpainting strategy is able to fill empty regions with realistic and temporal-consistent content (see also Sec I), and handles shadows reasonably well. Finally, we show that by fusing multiple views from the past context, WALDO is able to reduce disocclusions significantly (from the grey + black to only black regions in the last row of the figure).

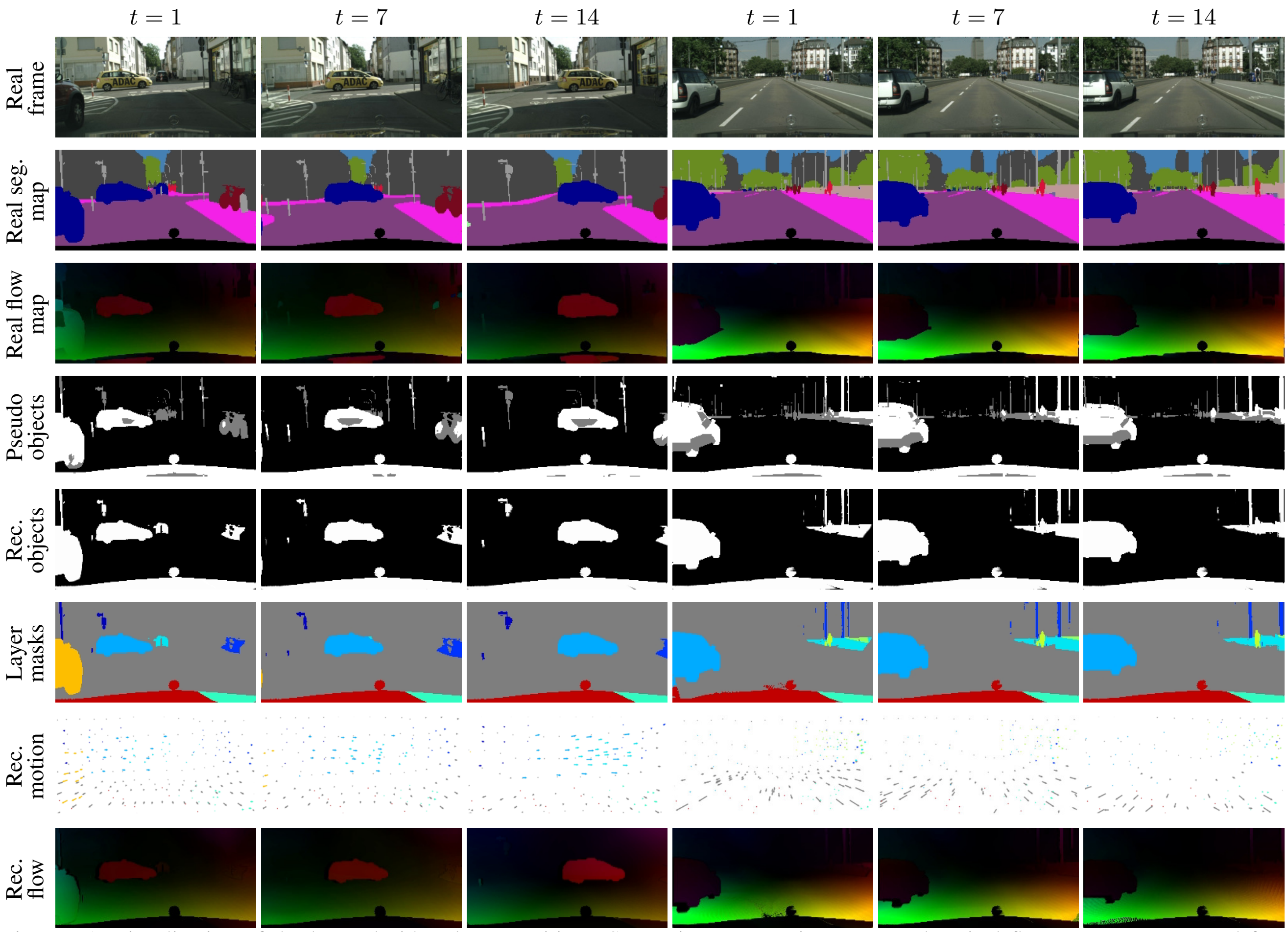

Figure L1. Visualization of the layered video decomposition. Semantic segmentation maps and optical flow maps are extracted from RGB frames using off-the-shelf methods [14, 77]. We combine both to construct pseudo ground truths for object discovery: in $\boxed{\text{white}}$, moving foreground regions towards which objects are attracted; in grey, static foreground ones which remain neutral; and in black, the background which repulses objects. We then show predicted object regions and their decomposition into layers. Each layer is tracked over time using a small set of control points. We compute motion vectors between points in pairs of frames (here, between consecutive time steps), and reconstruct from these and the layer masks the complex scene flow. Please zoom in for details.

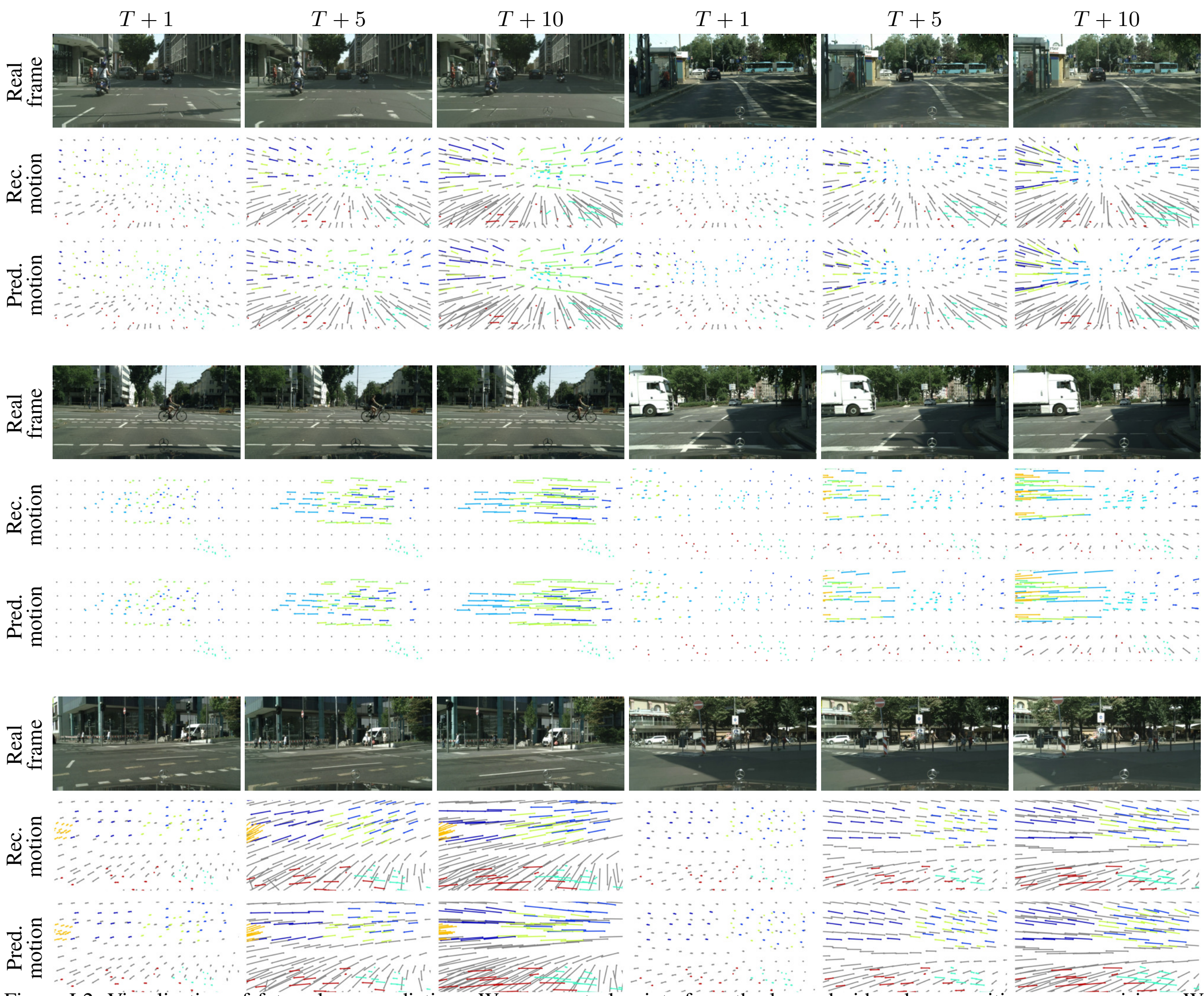

Figure L2. Visualization of future layer prediction. We use control points from the layered video decomposition as supervision. We compare motion vectors reconstructed from these points to the ones predicted for up to time step $T+10$ from a context of $T{=}4$ past frames. The motion vectors are computed between time step T and time step t in $\{T+1, T+10\}$. Different colors correspond to different layers. Please zoom in for details.

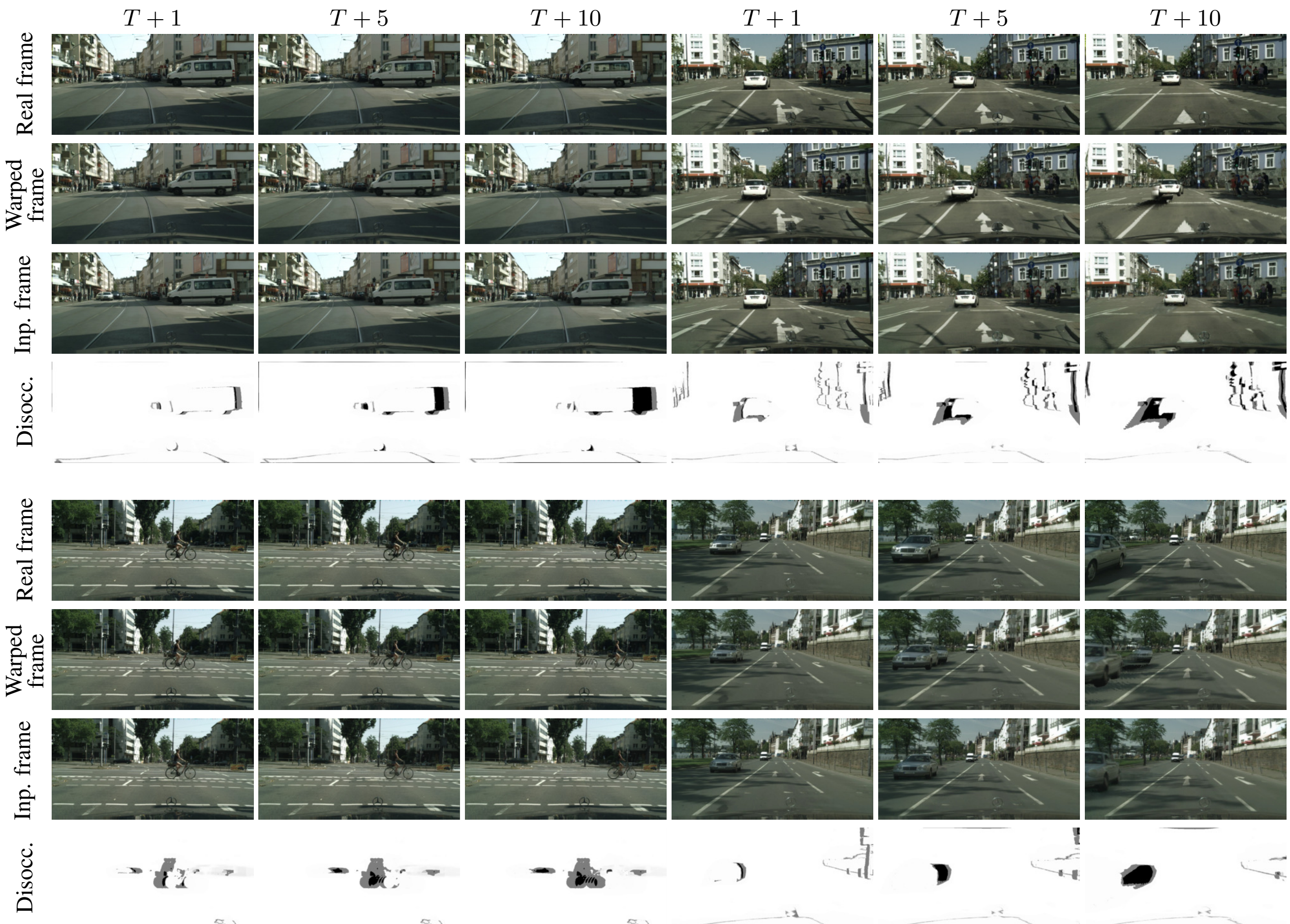

Figure L3. Visualization of the warping, fusion and inpainting module. We use the layer decomposition computed on a whole video to reconstruct the last 10 frames using the $T = 4$ first ones as context. This is done by warping, inpainting and fusing different views from the context. We compare real frames, warped ones, and fused/inpainted ones. We also illustrate the effect of disocclusion, by showing in the last row, in black, regions which are not visible in any frame of the context, and, in grey, those which are not visible in some frames but visible in others. Please zoom in for details.

References

[1] Adil Kaan Akan, Erkut Erdem, Aykut Erdem, and Fatma Güney. SLAMP: Stochastic latent appearance and motion prediction. In *ICCV*, 2021. 1, 2, 5, 6, 7

[2] Adil Kaan Akan, Sadra Safadoust, Erkut Erdem, Aykut Erdem, and Fatma Güney. Stochastic video prediction with structure and motion. *arXiv preprint*, 2022. 1, 2, 6

[3] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *ICML*, 2017. 16

[4] Mohammad Babaeizadeh, Chelsea Finn, Dumitru Erhan, Roy H. Campbell, and Sergey Levine. Stochastic variational video prediction. In *ICLR*, 2018. 2, 7

[5] Zhipeng Bao, Pavel Tokmakov, Allan Jabri, Yu-Xiong Wang, Adrien Gaidon, and Martial Hebert. Discovering objects that can move. In *CVPR*, 2022. 2, 3

[6] Amir Bar, Roei Herzig, Xiaolong Wang, Anna Rohrbach, Gal Chechik, Trevor Darrell, and Amir Globerson. Compositional video synthesis with action graphs. In *ICML*, 2021. 2

[7] Xinzhu Bei, Yanchao Yang, and Stefano Soatto. Learning semantic-aware dynamics for video prediction. In *CVPR*, 2021. 1, 2, 5, 6, 19

[8] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *TPAMI*, 2002. 3

[9] Fred L. Bookstein. Principal warps: Thin-plate splines and the decomposition of deformations. *TPAMI*, 1989. 1, 3, 7, 14

[10] Christopher P Burgess, Loic Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matt Botvinick, and Alexander Lerchner. Monet: Unsupervised scene decomposition and representation. *arXiv preprint*, 2019. 2

[11] Lluis Castrejon, Nicolas Ballas, and Aaron Courville. Improved conditional VRNNs for video prediction. In *ICCV*, 2019. 6

[12] Michael B Chang, Tomer Ullman, Antonio Torralba, and Joshua B Tenenbaum. A compositional object-based approach to learning physical dynamics. In *ICLR*, 2016. 2

[13] Zheng Chang, Xinfeng Zhang, Shanshe Wang, Siwei Ma, and Wen Gao. STRPM: A spatiotemporal residual predictive model for high-resolution video prediction. In *CVPR*, 2022. 1, 2, 5, 7

[14] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018. 2, 3, 5, 9, 19, 22

[15] Zhe Chen, Yuchen Duan, Wenhai Wang, Junjun He, Tong Lu, Jifeng Dai, and Yu Qiao. Vision transformer adapter for dense predictions. *arXiv preprint*, 2022. 9, 19

[16] Aidan Clark, Jeff Donahue, and Karen Simonyan. Adversarial video generation on complex datasets. *arXiv preprint*, 2019. 2

[17] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 2, 5, 16

[18] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009. 5

[19] Emily Denton and Rob Fergus. Stochastic video generation with a learned prior. In *ICML*, 2018. 2, 6

[20] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat GANs on image synthesis. In *NeurIPS*, 2021. 1

[21] Sébastien Ehrhardt, Oliver Groth, Aron Monszpart, Martin Engelcke, Ingmar Posner, Niloy J. Mitra, and Andrea Vedaldi. RELATE: Physically plausible multi-object scene synthesis using structured latent spaces. In *NeurIPS*, 2020. 2

[22] Martin Engelcke, Adam R Kosiorek, Oiwi Parker Jones, and Ingmar Posner. Genesis: Generative scene inference and sampling with object-centric latent representations. In *ICLR*, 2020. 2

[23] Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis. In *CVPR*, 2021. 1

[24] Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. In *NeurIPS*, 2016. 2

[25] Jean-Yves Franceschi, Edouard Delasalles, Mickaël Chen, Sylvain Lamprier, and Patrick Gallinari. Stochastic latent residual video prediction. In *ICML*, 2020. 2, 6

[26] Aditya Ganeshan, Alexis Vallet, Yasunori Kudo, Shin-ichi Maeda, Tommi Kerola, Rares Ambrus, Dennis Park, and Adrien Gaidon. Warp-refine propagation: Semi-supervised auto-labeling via cycle-consistency. In *ICCV*, 2021. 2

[27] Hang Gao, Huazhe Xu, Qi-Zhi Cai, Ruth Wang, Fisher Yu, and Trevor Darrell. Disentangling propagation and generation for video prediction. In *ICCV*, 2019. 2

[28] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The KITTI dataset. *IJRR*, 2013. 2, 5

[29] Daniel Geng, Max Hamilton, and Andrew Owens. Comparing correspondences: Video prediction with correspondence-wise losses. In *CVPR*, 2022. 1, 2, 6, 7, 19

[30] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014. 5, 15

[31] Klaus Greff, Raphaël Lopez Kaufman, Rishabh Kabra, Nick Watters, Christopher Burgess, Daniel Zoran, Loic Matthey, Matthew Botvinick, and Alexander Lerchner. Multi-object representation learning with iterative variational inference. In *ICML*, 2019. 2

[32] Ligong Han, Jian Ren, Hsin-Ying Lee, Francesco Barbieri, Kyle Olszewski, Shervin Minaee, Dimitris Metaxas, and Sergey Tulyakov. Show me what and tell me how: Video synthesis via multimodal conditioning. In *CVPR*, 2022. 2

[33] Zekun Hao, Xun Huang, and Serge Belongie. Controllable video generation with sparse trajectories. In *CVPR*, 2018. 2

[34] Paul Henderson and Christoph H. Lampert. Unsupervised object-centric video generation and decomposition in 3D. In *NeurIPS*, 2020. 2

[35] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs). *arXiv preprint*, 2016. 16

[36] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *arXiv preprint*, 2022. 1

[37] Jun-Ting Hsieh, Bingbin Liu, De-An Huang, Li F Fei-Fei, and Juan Carlos Niebles. Learning to decompose and disentangle representations for video prediction. In *NeurIPS*, 2018. 2

[38] Yaosi Hu, Chong Luo, and Zhenzhong Chen. Make it move: controllable image-to-video generation with text descriptions. In *CVPR*, 2022. 2

[39] Zhewei Huang, Tianyuan Zhang, Wen Heng, Boxin Shi, and Shuchang Zhou. Real-time intermediate flow estimation for video frame interpolation. In *ECCV*, 2022. 19

[40] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *CVPR*, 2017. 19

[41] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3.6M: Large scale datasets and predictive methods for 3d human sensing in natural environments. *TPAMI*, 2013. 2, 5

[42] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *NeurIPS*, 2015. 2, 15

[43] Nebojsa Jojic and Brendan J Frey. Learning flexible sprites in video layers. In *CVPR*, 2001. 2, 4

[44] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. In *NeurIPS*, 2021. 1

[45] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 5

[46] Thomas Kipf, Gamaleldin F. Elsayed, Aravindh Mahendran, Austin Stone, Sara Sabour, Georg Heigold, Rico Jonschkowski, Alexey Dosovitskiy, and Klaus Greff. Conditional Object-Centric Learning from Video. In *ICLR*, 2022. 2

[47] Manoj Kumar, Mohammad Babaeizadeh, Dumitru Erhan, Chelsea Finn, Sergey Levine, Laurent Dinh, and Durk Kingma. VideoFlow: A conditional flow-based model for stochastic video generation. In *ICLR*, 2020. 2

[48] Yong-Hoon Kwon and Min-Gyu Park. Predicting future frames using retrospective Cycle GAN. In *CVPR*, 2019. 7

[49] Alex X Lee, Richard Zhang, Frederik Ebert, Pieter Abbeel, Chelsea Finn, and Sergey Levine. Stochastic adversarial video prediction. *arXiv preprint*, 2018. 7

[50] Bo Li, Wei Wu, Qiang Wang, Fangyi Zhang, Junliang Xing, and Junjie Yan. SiamRPN++: Evolution of siamese visual tracking with very deep networks. In *CVPR*, 2019. 19

[51] Wenbo Li, Zhe Lin, Kun Zhou, Lu Qi, Yi Wang, and Jiaya Jia. MAT: Mask-aware transformer for large hole image inpainting. In *CVPR*, 2022. 5, 9, 20

[52] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Flow-grounded spatial-temporal video prediction from still images. In *ECCV*, 2018. 2

[53] Ziwei Liu, Raymond A Yeh, Xiaoou Tang, Yiming Liu, and Aseem Agarwala. Video frame synthesis using deep voxel flow. In *ICCV*, 2017. 6

[54] Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. In *NeurIPS*, 2020. 2, 4

[55] William Lotter, Gabriel Kreiman, and David Cox. Deep predictive coding networks for video prediction and unsupervised learning. In *ICLR*, 2017. 6

[56] Erika Lu, Forrester Cole, Tali Dekel, Andrew Zisserman, William T Freeman, and Michael Rubinstein. Omnimatte: associating objects and their effects in video. In *CVPR*, 2021. 2

[57] Pauline Luc, Aidan Clark, Sander Dieleman, Diego de Las Casas, Yotam Doron, Albin Cassirer, and Karen Simonyan. Transformation-based adversarial video prediction on large-scale data. *arXiv preprint*, 2020. 2

[58] Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. In *ICLR*, 2016. 7

[59] Jacob Menick and Nal Kalchbrenner. Generating high fidelity images with subscale pixel networks and multidimensional upscaling. In *ICLR*, 2019. 1

[60] Matthias Minderer, Chen Sun, Ruben Villegas, Forrester Cole, Kevin P Murphy, and Honglak Lee. Unsupervised learning of object structure and dynamics from videos. In *NeurIPS*, 2019. 2

[61] Guillaume Le Moing, Jean Ponce, and Cordelia Schmid. CCVS: Context-aware controllable video synthesis. In *NeurIPS*, 2021. 1, 2

[62] Tom Monnier, Elliot Vincent, Jean Ponce, and Mathieu Aubry. Unsupervised layered image decomposition into object prototypes. In *ICCV*, 2021. 2, 4

[63] Charlie Nash, João Carreira, Jacob Walker, Iain Barr, Andrew Jaegle, Mateusz Malinowski, and Peter Battaglia. Transframer: Arbitrary frame prediction with generative models. *arXiv preprint*, 2022. 1

[64] Charlie Nash, Jacob Menick, Sander Dieleman, and Peter W Battaglia. Generating images with sparse representations. In *ICML*, 2021. 1

[65] Eunbyung Park, Jimei Yang, Ersin Yumer, Duygu Ceylan, and Alexander C Berg. Transformation-grounded image generation network for novel 3D view synthesis. In *CVPR*, 2017. 2

[66] M Pawan Kumar, Philip HS Torr, and Andrew Zisserman. Learning layered motion segmentations of video. In *ICCV*, 2008. 2

[67] Mikel D Rodriguez, Javed Ahmed, and Mubarak Shah. Action MACH a spatio-temporal maximum average correlation height filter for action recognition. In *CVPR*, 2008. 2, 5

[68] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 5, 17

[69] Masaki Saito, Shunta Saito, Masanori Koyama, and Sosuke Kobayashi. Train sparsely, generate densely: Memory-efficient unsupervised training of high-resolution temporal GAN. *IJCV*, 2020. 2

[70] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018. 9, 19

[71] Karl Schmeckpeper, Georgios Georgakis, and Kostas Daniilidis. Object-centric video prediction without annotation. In *ICRA*, 2021. 2

[72] Baoguang Shi, Xinggang Wang, Pengyuan Lyu, Cong Yao, and Xiang Bai. Robust scene text recognition with automatic rectification. In *CVPR*, 2016. 2

[73] Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. First order motion model for image animation. In *NeurIPS*, 2019. 19

[74] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 5

[75] Deqing Sun, Jonas Wulff, Erik B Sudderth, Hanspeter Pfister, and Michael J Black. A fully-connected layered model of foreground and background flow. In *CVPR*, 2013. 2

[76] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In *CVPR*, 2018. 9, 19

[77] Zachary Teed and Jia Deng. RAFT: Recurrent all-pairs field transforms for optical flow. In *ECCV*, 2020. 2, 3, 5, 9, 19, 22

[78] D'Arcy Wentworth Thompson. On growth and form. *Nature*, 1917. 3

[79] Yu Tian, Jian Ren, Menglei Chai, Kyle Olszewski, Xi Peng, Dimitris N. Metaxas, and Sergey Tulyakov. A good image generator is what you need for high-resolution video synthesis. In *ICLR*, 2021. 1, 2

[80] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. MoCoGAN: Decomposing motion and content for video generation. In *CVPR*, 2018. 2

[81] Thomas Unterthiner, Sjoerd van Steenkiste, Karol Kurach, Raphaël Marinier, Marcin Michalski, and Sylvain Gelly. FVD: A new metric for video generation. In *ICLRw*, 2019. 5

[82] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *NeurIPS*, 2017. 1

[83] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 4, 8

[84] Ruben Villegas, Arkanath Pathak, Harini Kannan, Dumitru Erhan, Quoc V Le, and Honglak Lee. High fidelity video prediction with large stochastic recurrent neural networks. In *NeurIPS*, 2019. 7

[85] Ruben Villegas, Jimei Yang, Seunghoon Hong, Xunyu Lin, and Honglak Lee. Decomposing motion and content for natural video sequence prediction. In *ICLR*, 2017. 6

[86] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. In *NeurIPS*, 2016. 2

[87] Carl Vondrick and Antonio Torralba. Generating the future with adversarial transformers. In *CVPR*, 2017. 2

[88] Jacob Walker, Kenneth Marino, Abhinav Gupta, and Martial Hebert. The pose knows: Video forecasting by generating pose futures. In *ICCV*, 2017. 2

[89] John YA Wang and Edward H Adelson. Representing moving images with layers. *TIP*, 1994. 2

[90] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Video-to-video synthesis. In *NeurIPS*, 2018. 2, 6, 19

[91] Yunbo Wang, Zhifeng Gao, Mingsheng Long, Jianmin Wang, and S Yu Philip. PredRNN++: Towards a resolution of the deep-in-time dilemma in spatiotemporal predictive learning. In *ICML*, 2018. 7

[92] Yunbo Wang, Lu Jiang, Ming-Hsuan Yang, Li-Jia Li, Mingsheng Long, and Li Fei-Fei. Eidetic 3D LSTM: A model for video prediction and beyond. In *ICLR*, 2019. 7

[93] Yunbo Wang, Mingsheng Long, Jianmin Wang, Zhifeng Gao, and Philip S Yu. PredRNN: Recurrent neural networks for predictive learning using spatiotemporal lstms. In *NeurIPS*, 2017. 7

[94] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *TIP*, 2004. 5

[95] Dirk Weissenborn, Oscar Täckström, and Jakob Uszkoreit. Scaling autoregressive video models. In *ICLR*, 2020. 1

[96] Bohan Wu, Suraj Nair, Roberto Martin-Martin, Li Fei-Fei, and Chelsea Finn. Greedy hierarchical variational autoencoders for large-scale video prediction. In *CVPR*, 2021. 1

[97] Chenfei Wu, Jian Liang, Lei Ji, Fan Yang, Yuejian Fang, Daxin Jiang, and Nan Duan. Nüwa: Visual synthesis pre-training for neural visual world creation. *arXiv preprint*, 2021. 1, 2

[98] Haixu Wu, Zhiyu Yao, Jianmin Wang, and Mingsheng Long. MotionRNN: A flexible model for video prediction with spacetime-varying motions. In *CVPR*, 2021. 7

[99] Yue Wu, Rongrong Gao, Jaesik Park, and Qifeng Chen. Future video synthesis with object motion prediction. In *CVPR*, 2020. 1, 2, 6, 7, 19, 21

[100] Yue Wu, Qiang Wen, and Qifeng Chen. Optimizing video prediction via video frame interpolation. In *CVPR*, 2022. 1, 2, 6, 7, 19

[101] Jonas Wulff and Michael J Black. Efficient sparse-to-dense optical flow estimation using a learned basis and layers. In *CVPR*, 2015. 2

[102] Yuwen Xiong, Renjie Liao, Hengshuang Zhao, Rui Hu, Min Bai, Ersin Yumer, and Raquel Urtasun. UPSNet: A unified panoptic segmentation network. In *CVPR*, 2019. 19

[103] Wilson Yan, Yunzhi Zhang, Pieter Abbeel, and Aravind Srinivas. VideoGPT: Video generation using VQ-VAE and transformers. *arXiv preprint*, 2021. 1

[104] Charig Yang, Hala Lamdouar, Erika Lu, Andrew Zisserman, and Weidi Xie. Self-supervised video object segmentation by motion grouping. In *ICCV*, 2021. 2, 3

[105] Vickie Ye, Zhengqi Li, Richard Tucker, Angjoo Kanazawa, and Noah Snavely. Deformable sprites for unsupervised video decomposition. In *CVPR*, 2022. 2

[106] Yufei Ye, Maneesh Singh, Abhinav Gupta, and Shubham Tulsiani. Compositional video prediction. In *ICCV*, 2019. 2

[107] Zhichao Yin and Jianping Shi. Geonet: Unsupervised learning of dense depth, optical flow and camera pose. In *CVPR*, 2018. 19

[108] Wei Yu, Yichao Lu, Steve Easterbrook, and Sanja Fidler. Efficient and information-preserving future frame prediction and beyond. In *ICLR*, 2020. 7

[109] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 5

[110] Yunzhi Zhang and Jiajun Wu. Video extrapolation in space and time. In *ECCV*, 2022. 6

[111] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *TPAMI*, 2017. 5, 9, 20

[112] Tinghui Zhou, Shubham Tulsiani, Weilun Sun, Jitendra Malik, and Alexei A Efros. View synthesis by appearance flow. In *ECCV*, 2016. 2

[113] Yi Zhu, Karan Sapra, Fitsum A Reda, Kevin J Shih, Shawn Newsam, Andrew Tao, and Bryan Catanzaro. Improving semantic segmentation via video propagation and label relaxation. In *CVPR*, 2019. 19