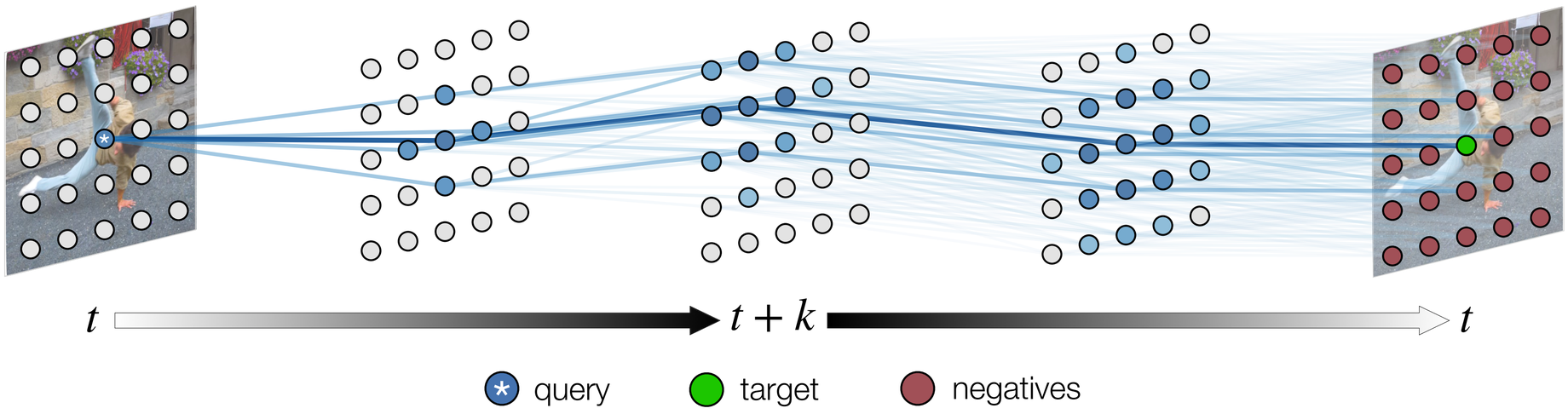
Lecture 10: Motion

1. Contrastive random walks
2. Self-supervised optical flow

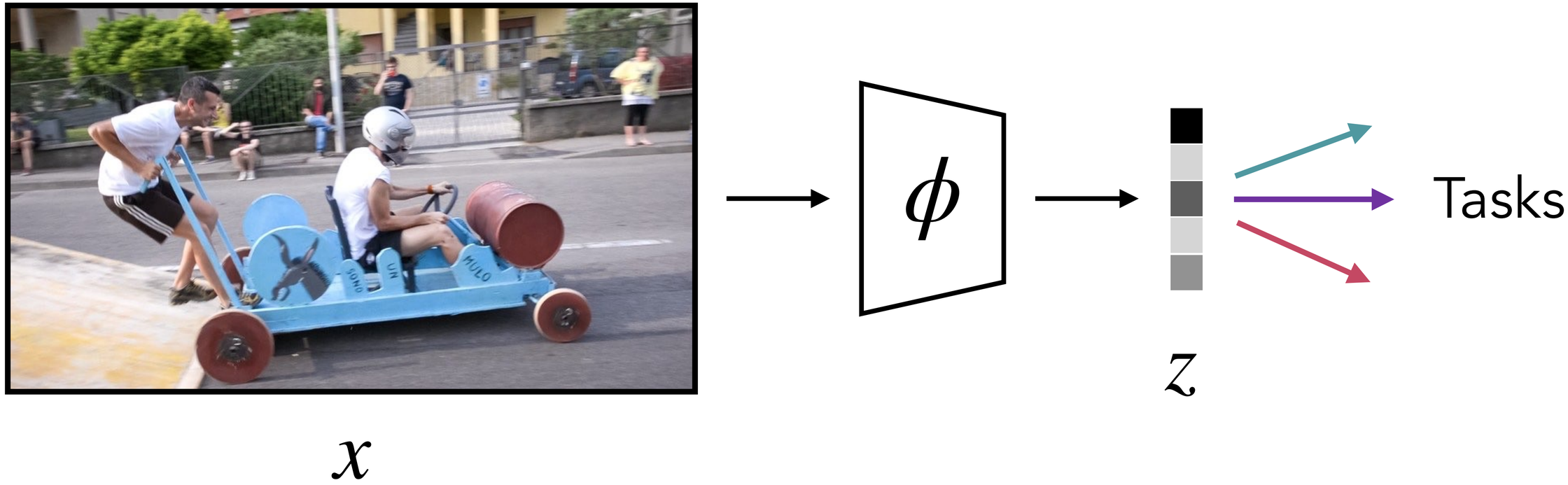
Space-Time Correspondence as a Contrastive Random Walk



Allan Jabri, Andrew Owens, Alexei A. Efros

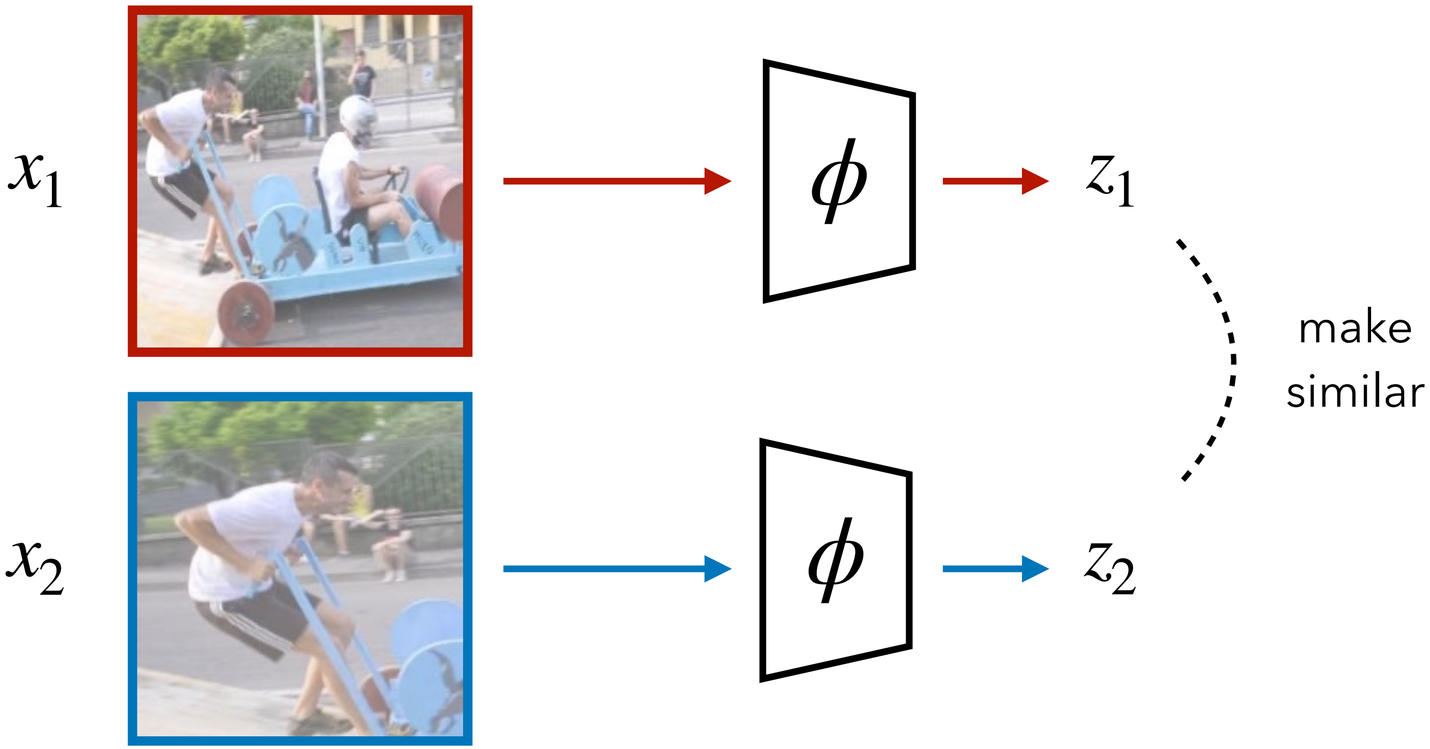
(on ArXiv next week!)

Representation Learning



Representation Learning via Pairwise Similarity

Metric Learning
Siamese Nets
Contrastive Learning
etc



Becker et al (1992)
de Sa (1993)
Bromley et al (1994)
Chopra et al (2005)
Dosovitsky et al (2014)
Bojanowski et al (2017)
Wu et al (2018)
van den Oord et al (2019)
Tian et al (2019)
He et al (2019)
Chen et al (2020)

1. Improved Siamese networks
2. **Training Data via Data Augmentation**

Data Augmentation

input

color

crop

flip

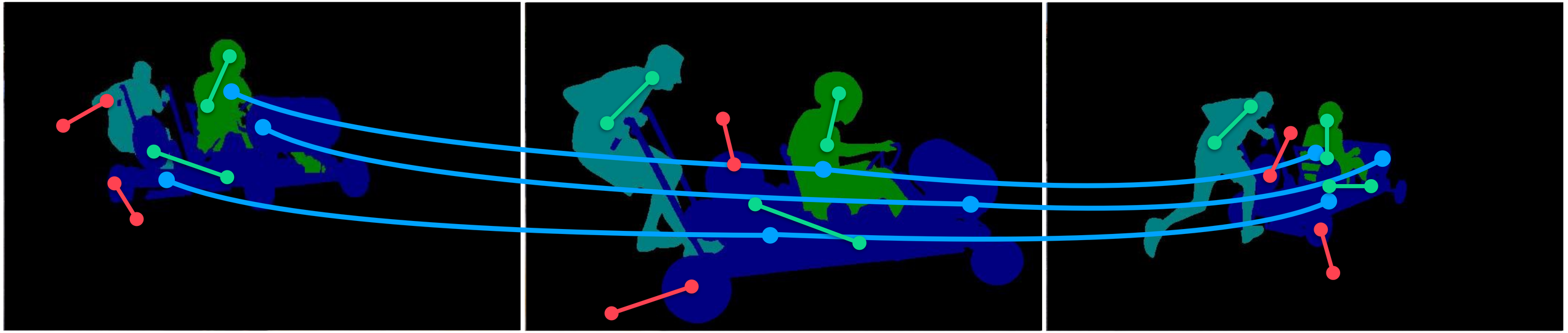
blur

Views

SimCLR augmentations (Chen et al, 2020)

Dynamics in Video

How to harness without supervision?



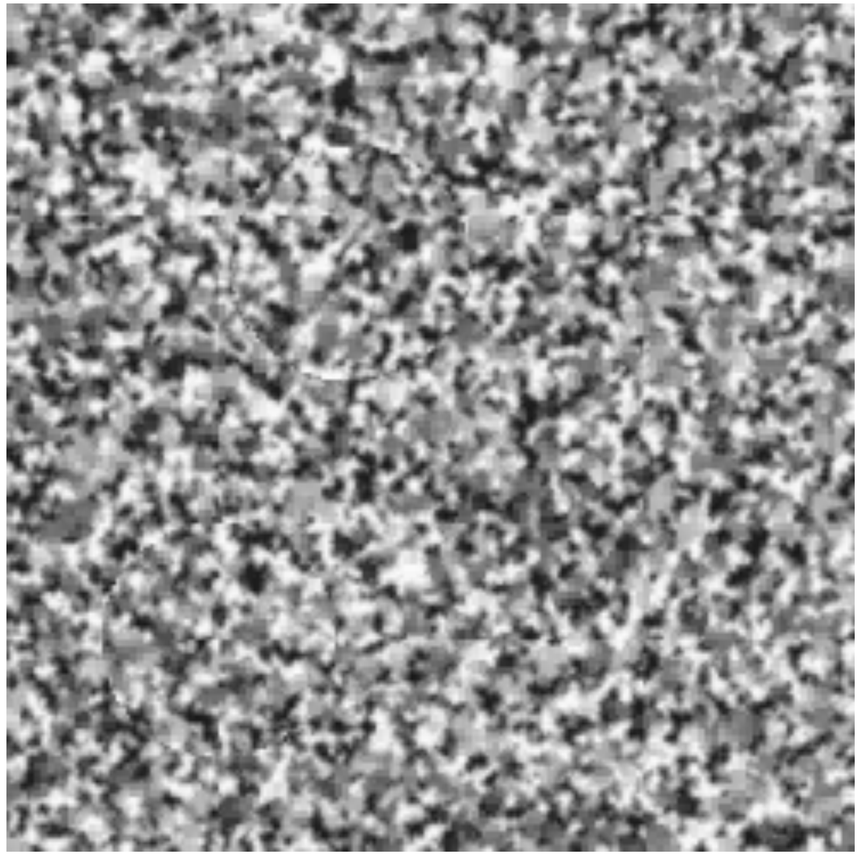
Correspondence

"Common Fate"

Wehrtheimer (1938)

Learning Representations from Video

Motion is a powerful perceptual cue



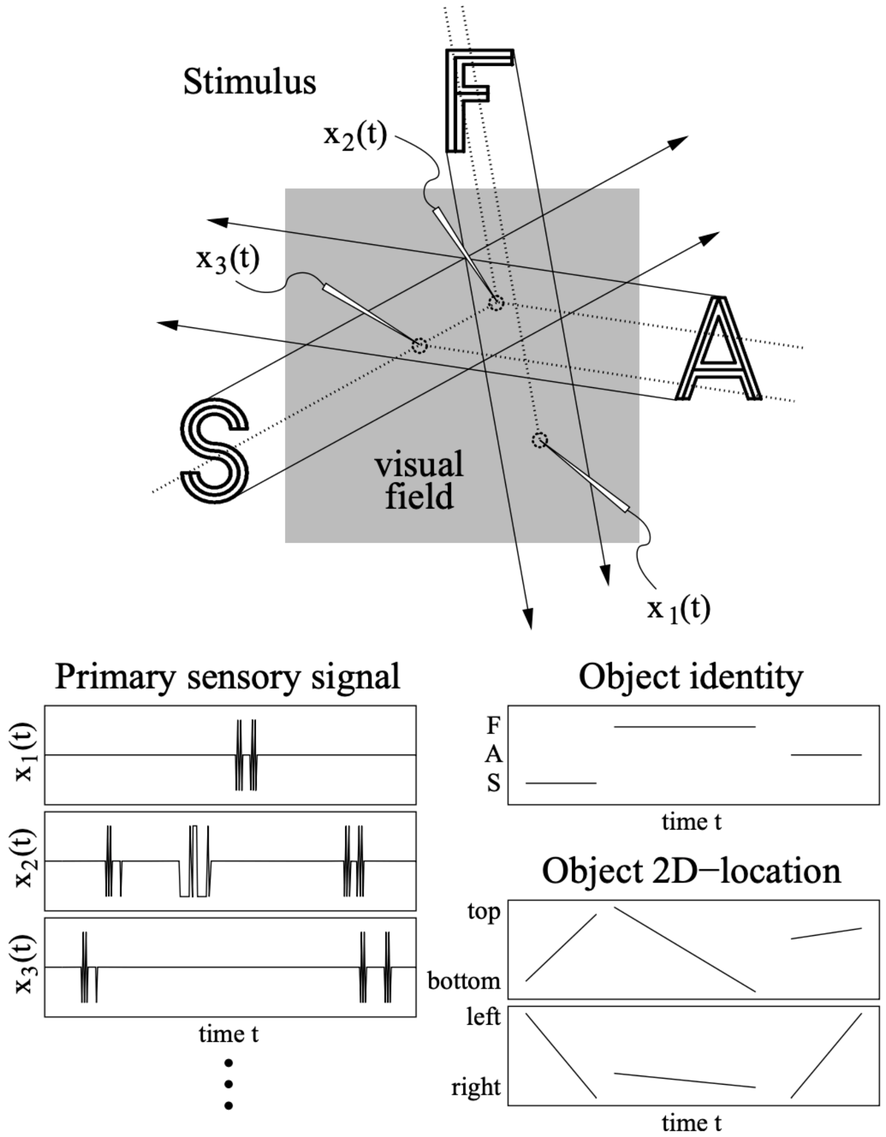
Source: S. Lazebnik

Motion is a powerful perceptual cue

G. Johansson, “Visual Perception of Biological Motion and a Model For Its Analysis",
Perception and Psychophysics 14, 201-211, 1973.

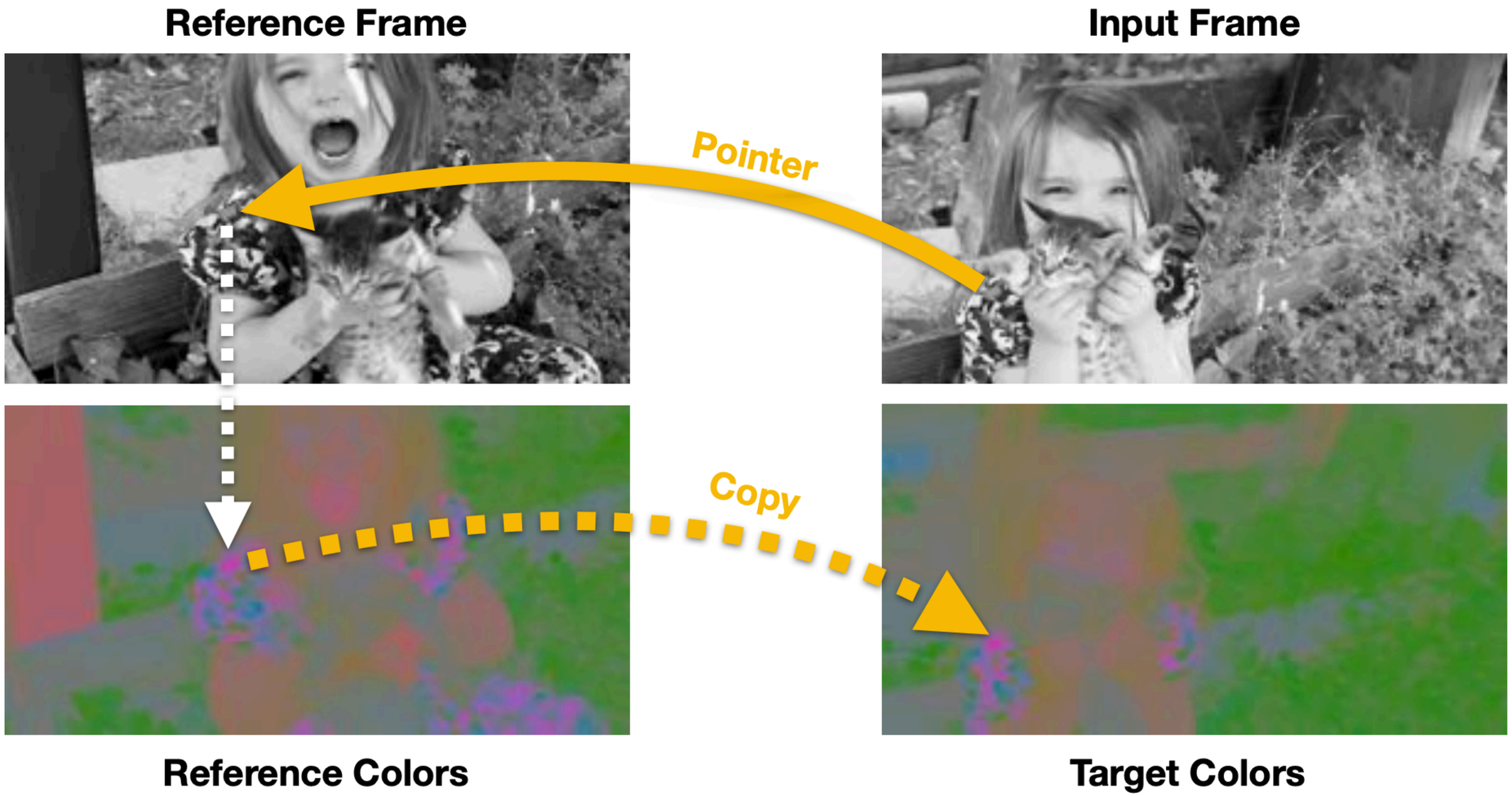
Source: S. Lazebnik

Video representation learning



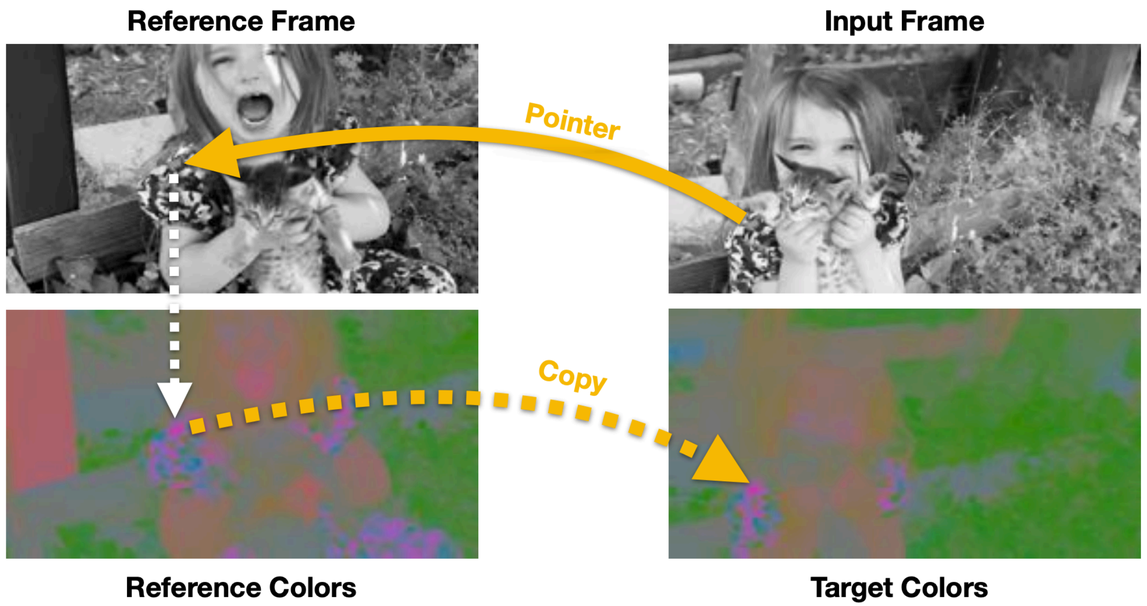
[Wiskott & Sejnowski. “Slow Feature Analysis”, 2002]

Learning temporal correspondences



[Vondrick et al., “Tracking Emerges by Colorizing Videos”, 2018]

Limitations to colorization



- Corresponding pixels don't always have the same color
  - Arguably the most important examples!
- Underconstrained: many possible valid correspondences for each pixel!
- Somewhat tied to RGB video modality

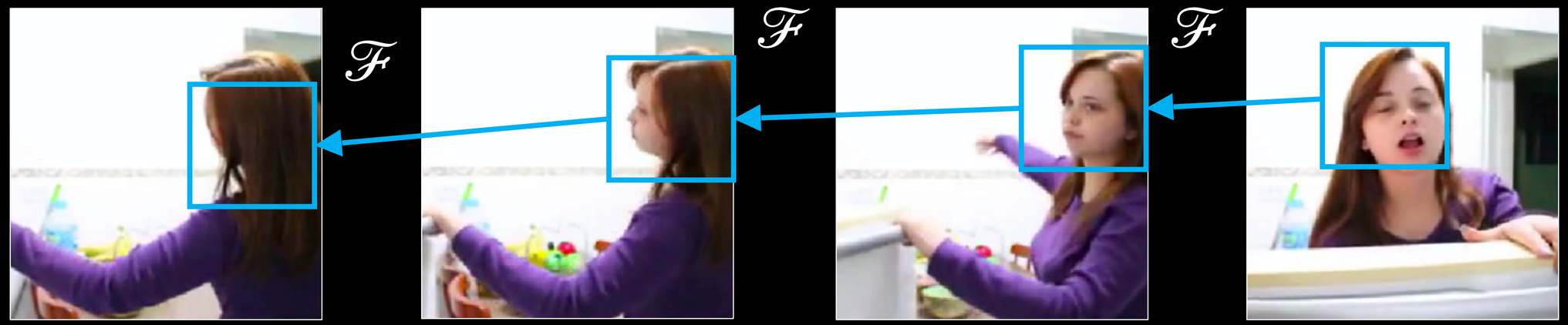
Learning Correspondence from the Cycle-consistency of Time

CVPR 2019

Xiaolong Wang, A. Jabri, Alexei A. Efros

Learning to Track

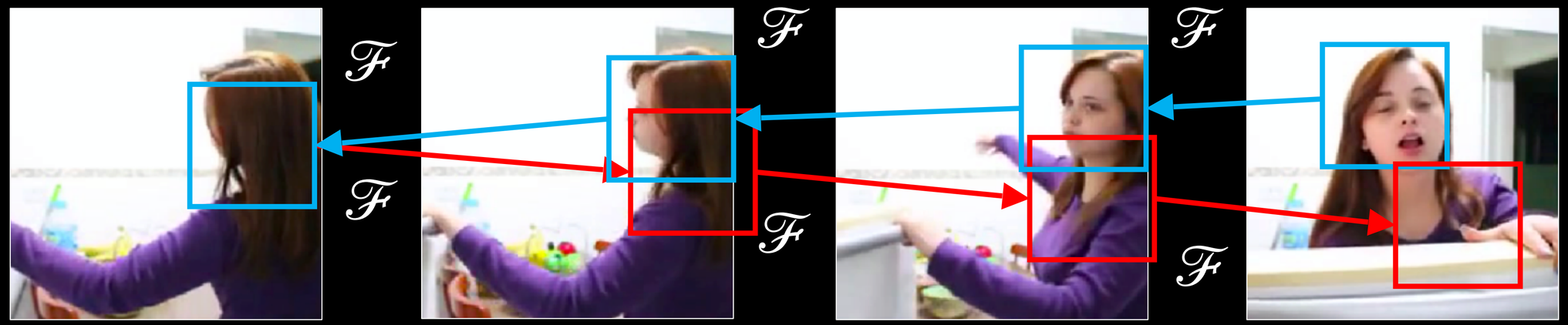
$\mathcal{F}$: a deep tracker



How to obtain supervision?

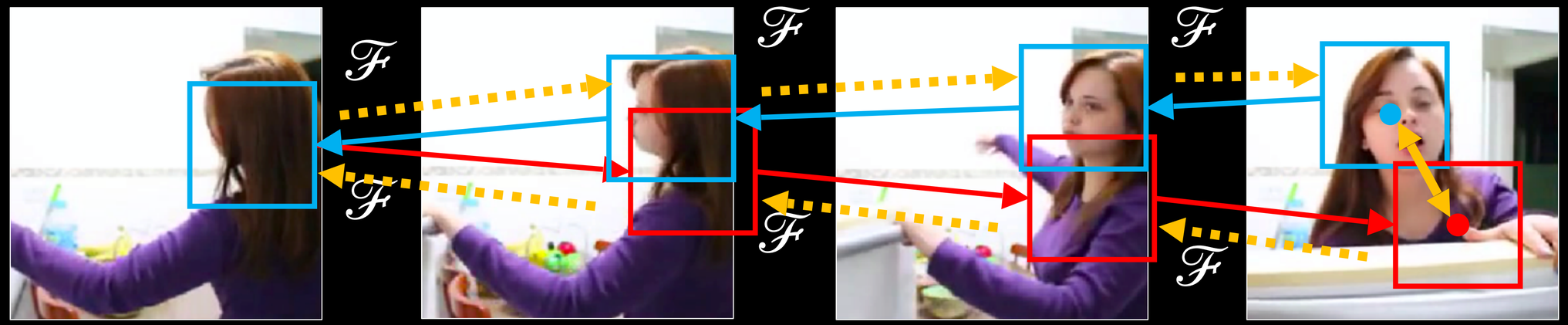
Supervision: Cycle-Consistency in Time

Track backwards



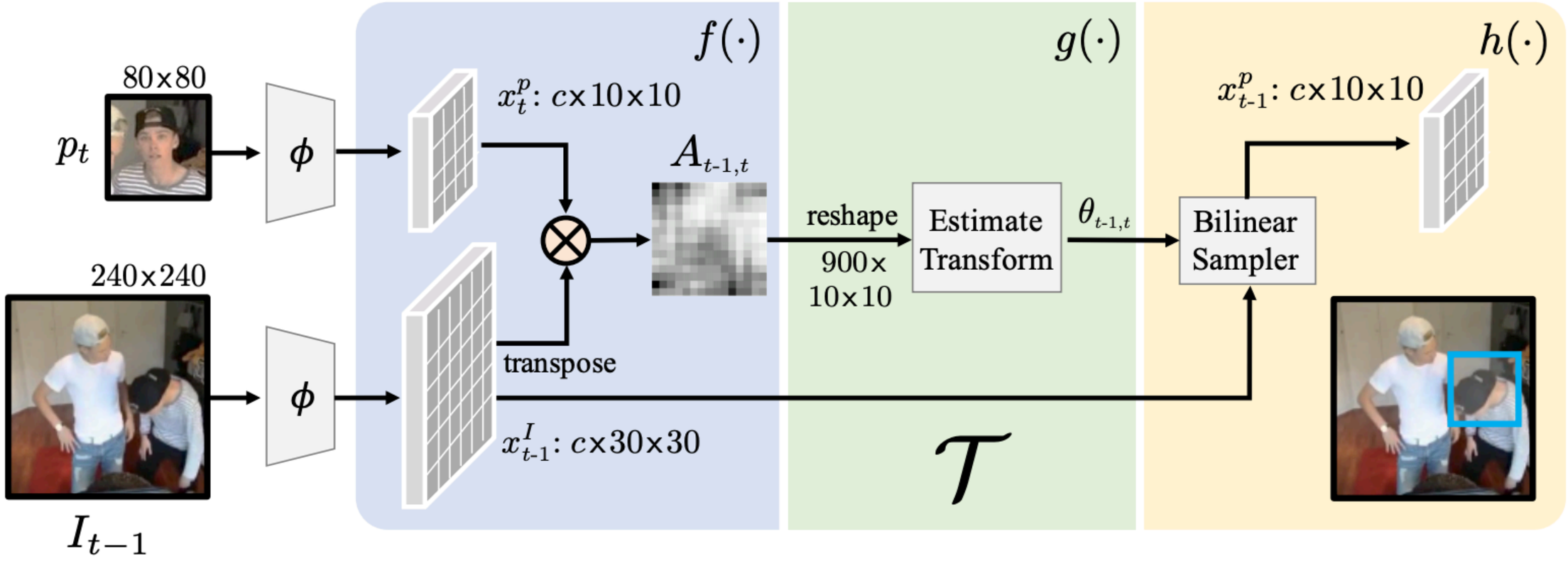
Track forwards, back to the future

Supervision: Cycle-Consistency in Time



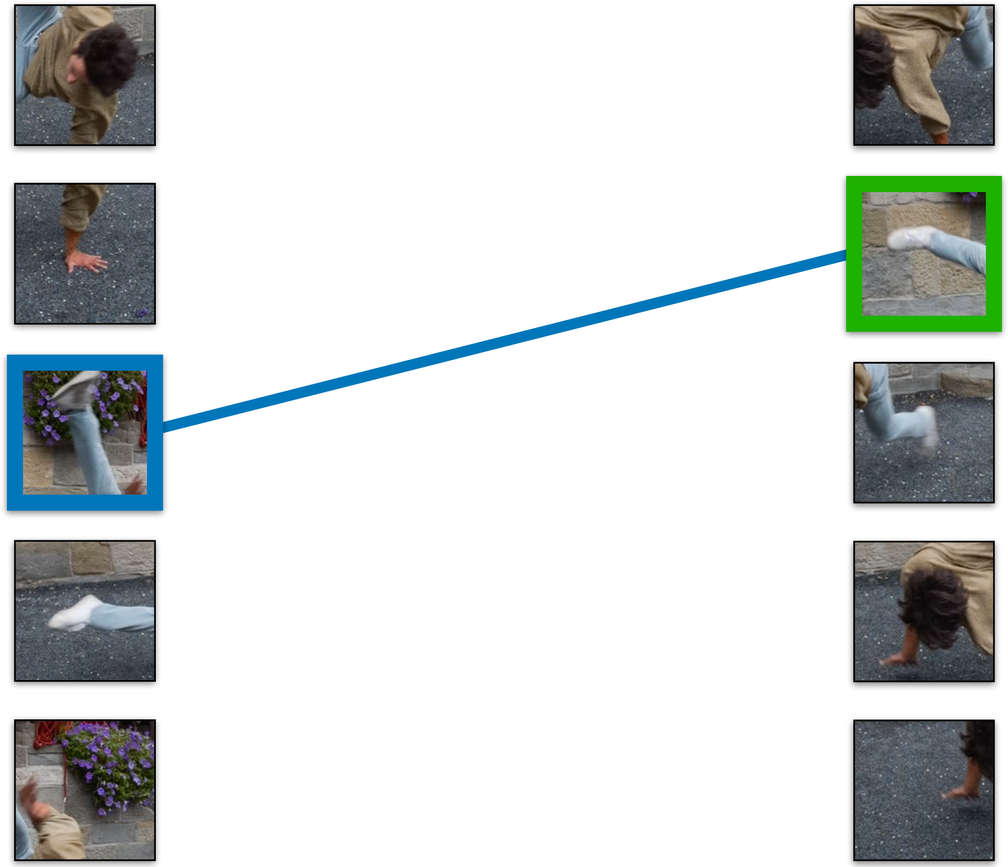
Backpropagation through time, along the cycle

Differentiable tracking

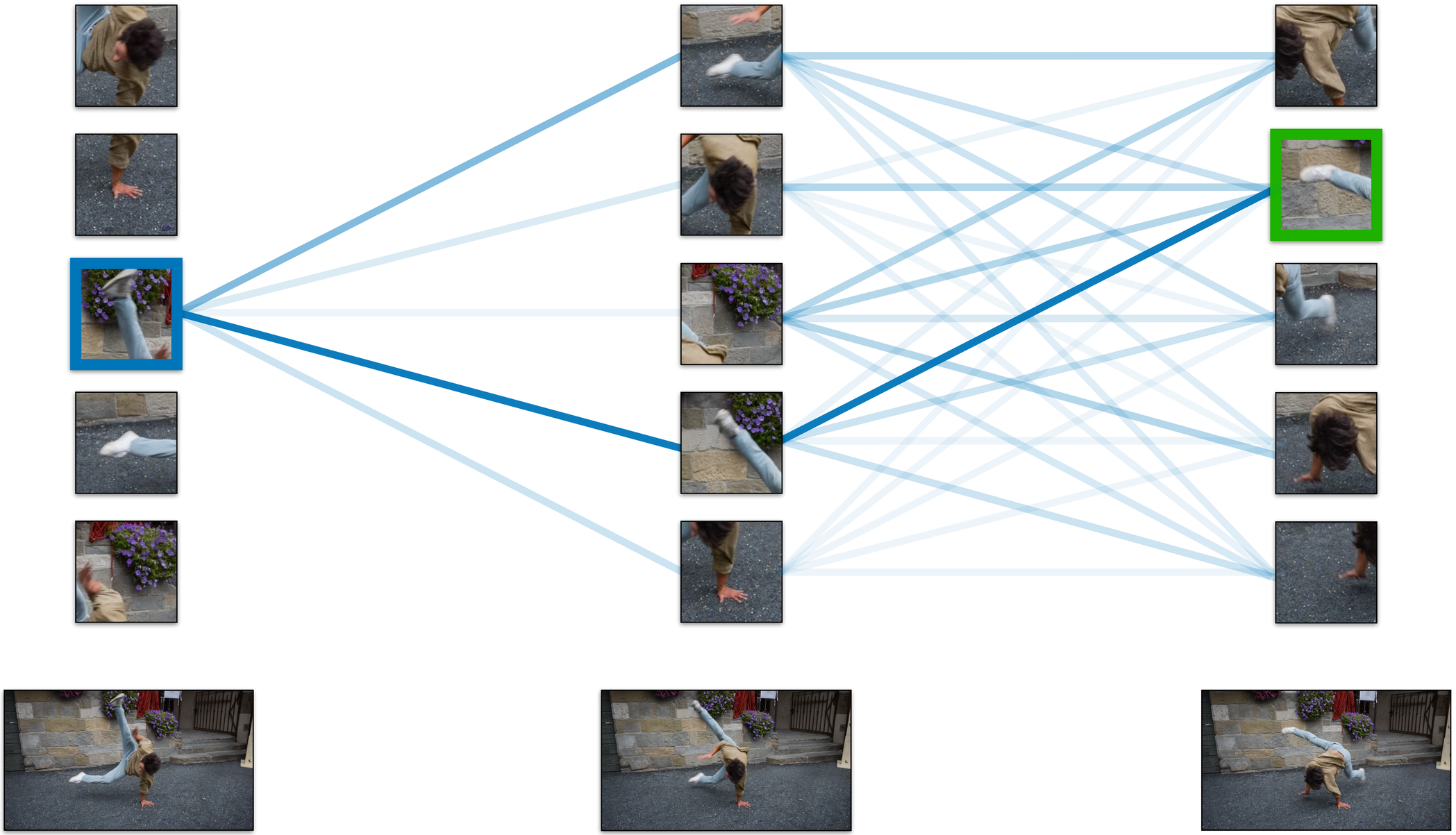


Supervised learning

Supervised Learning

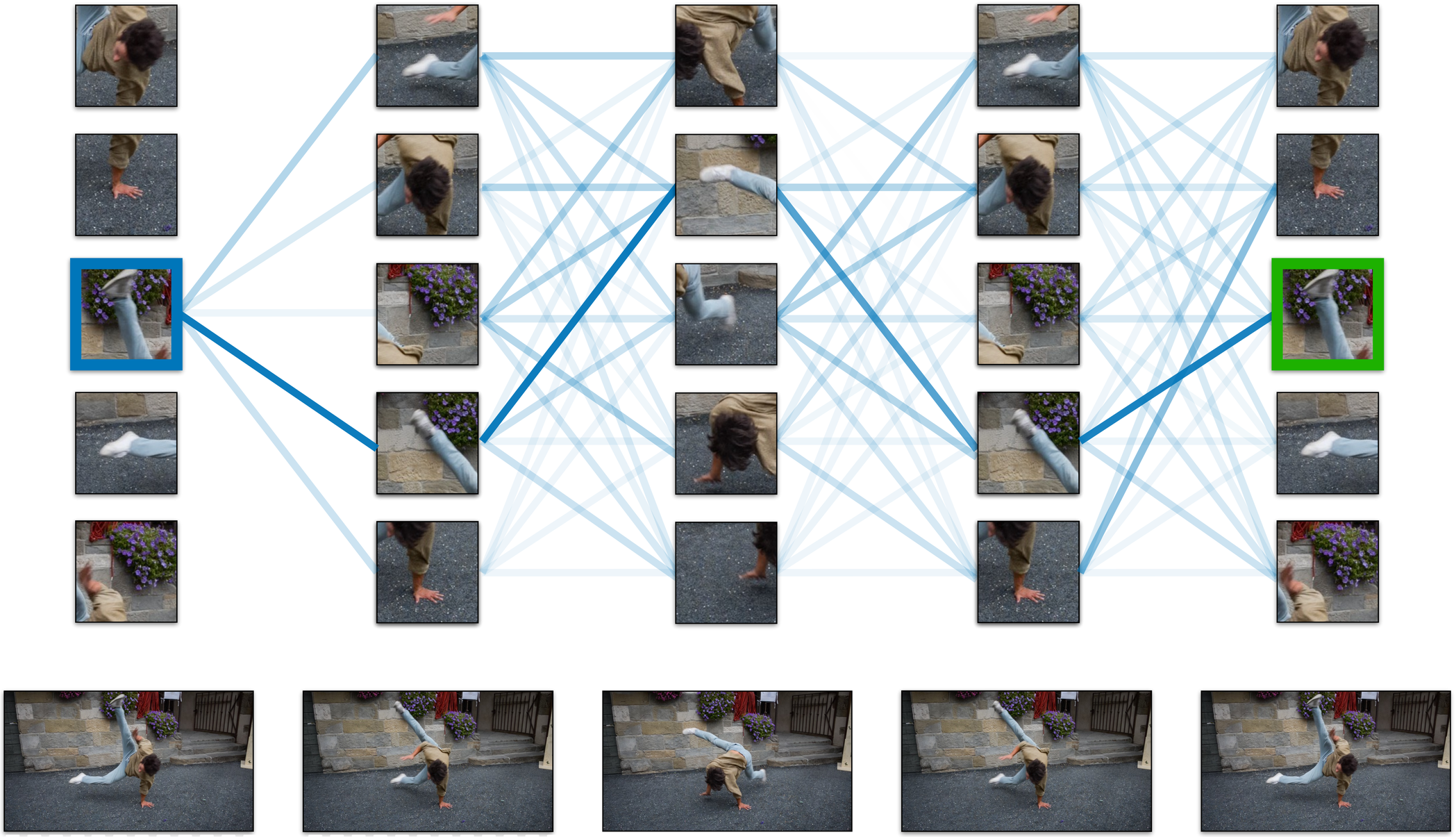


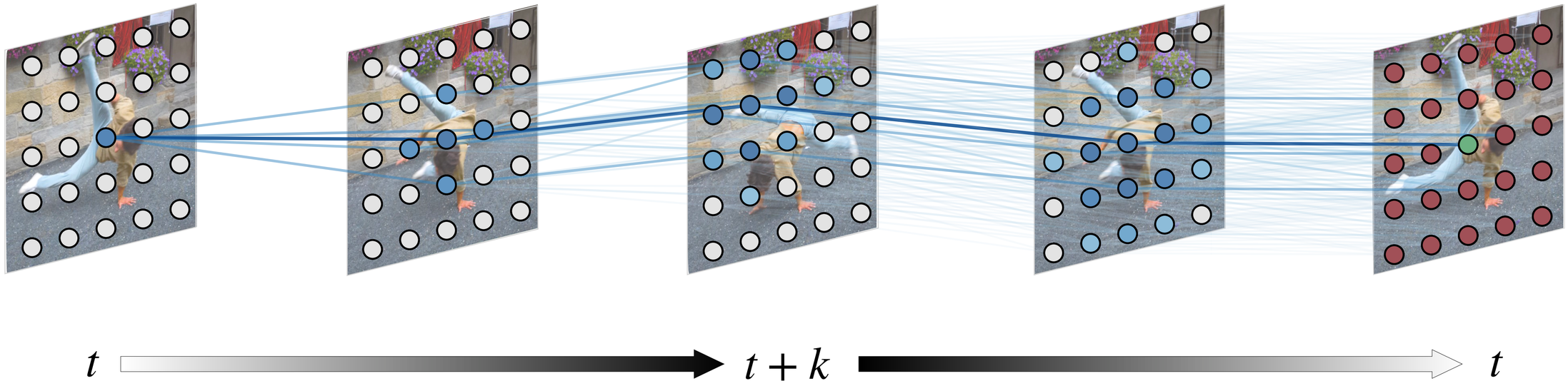
Latent views

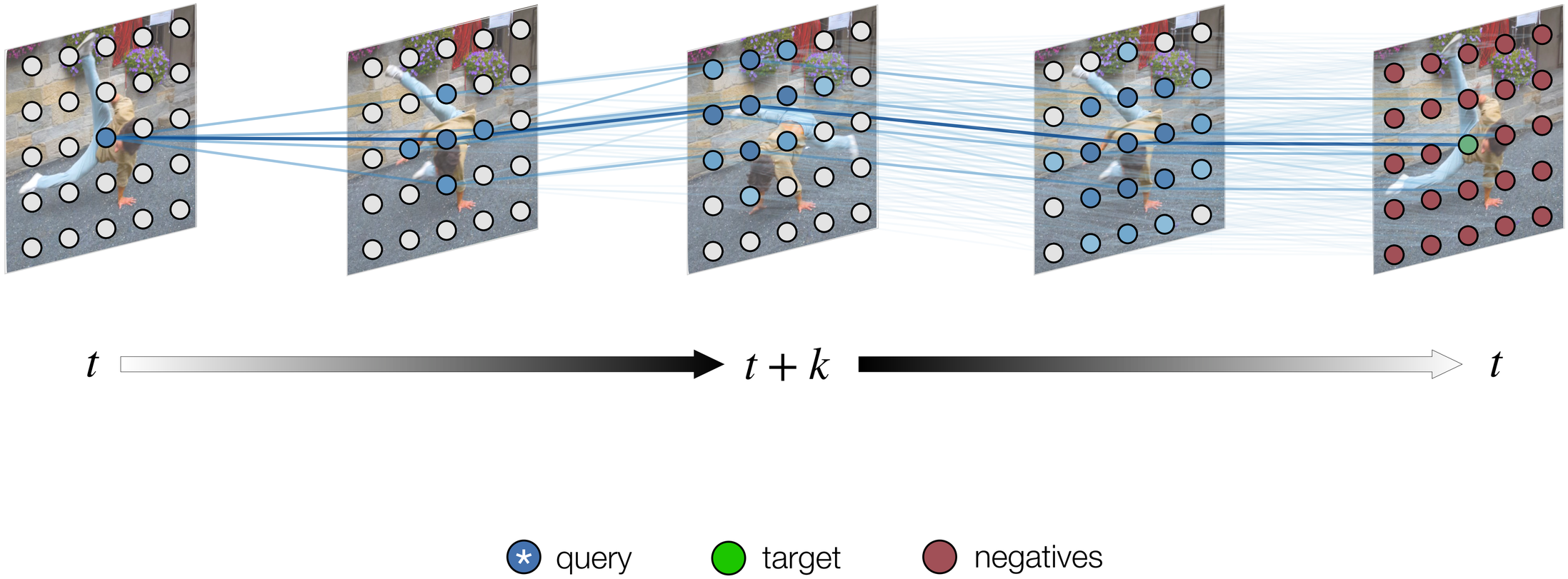


Palindromes

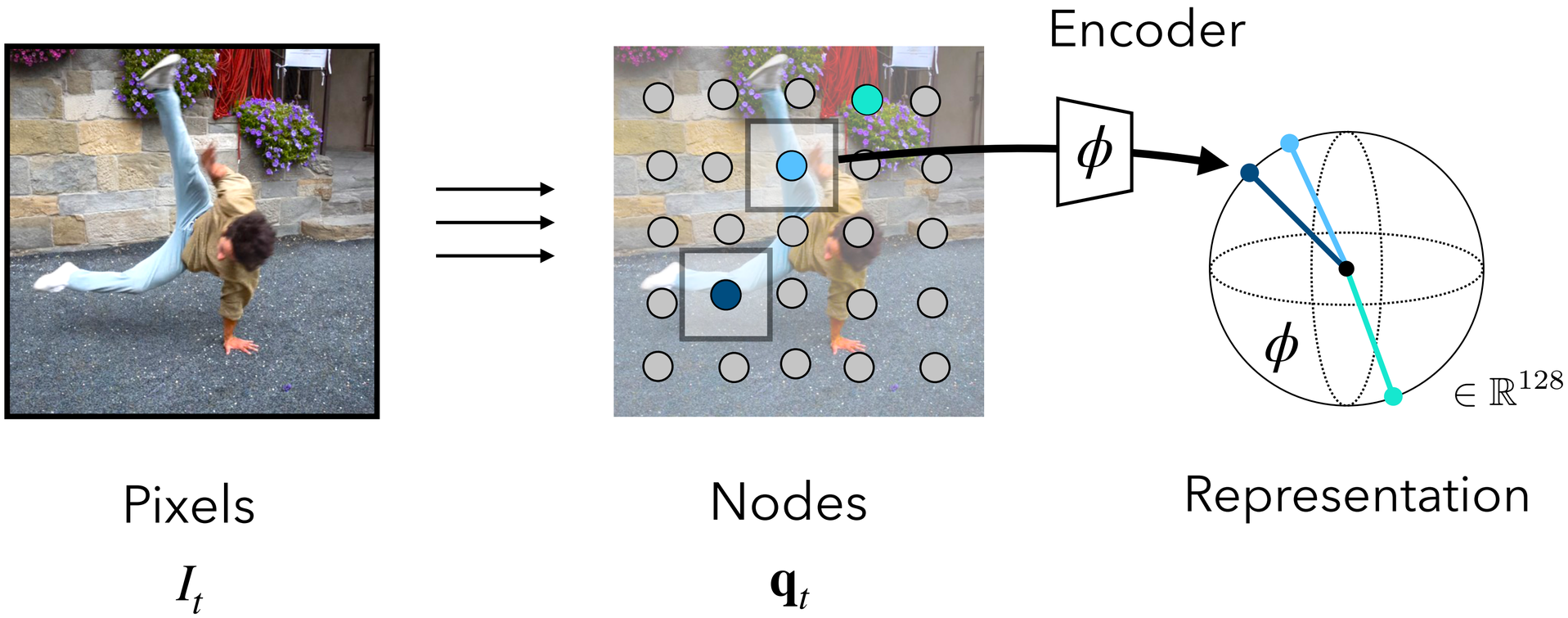
Self-supervised Learning



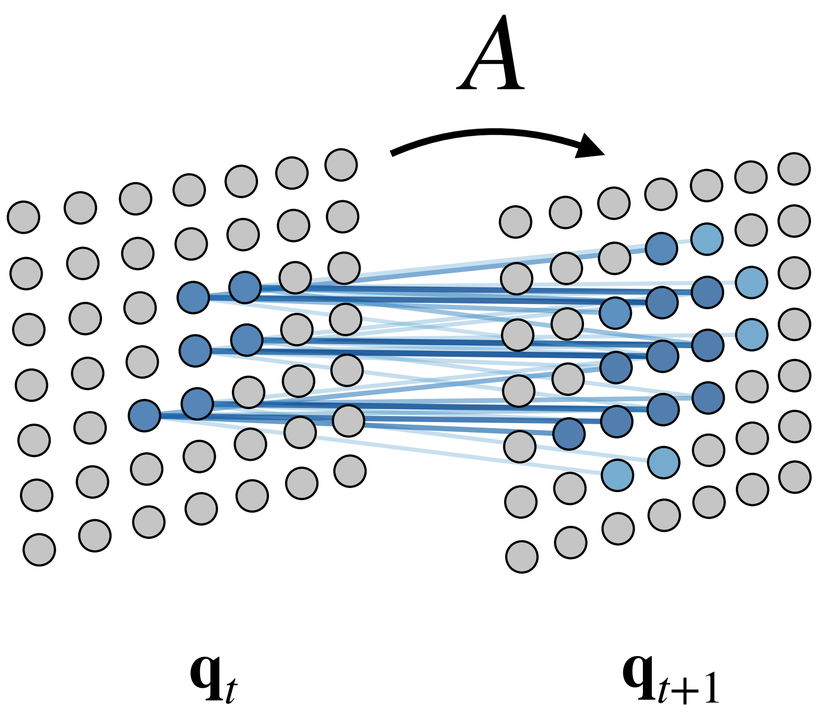

t
$t+k$
t


t
$t+k$
t
query
target
negatives

Video as a Graph



Video as a Graph

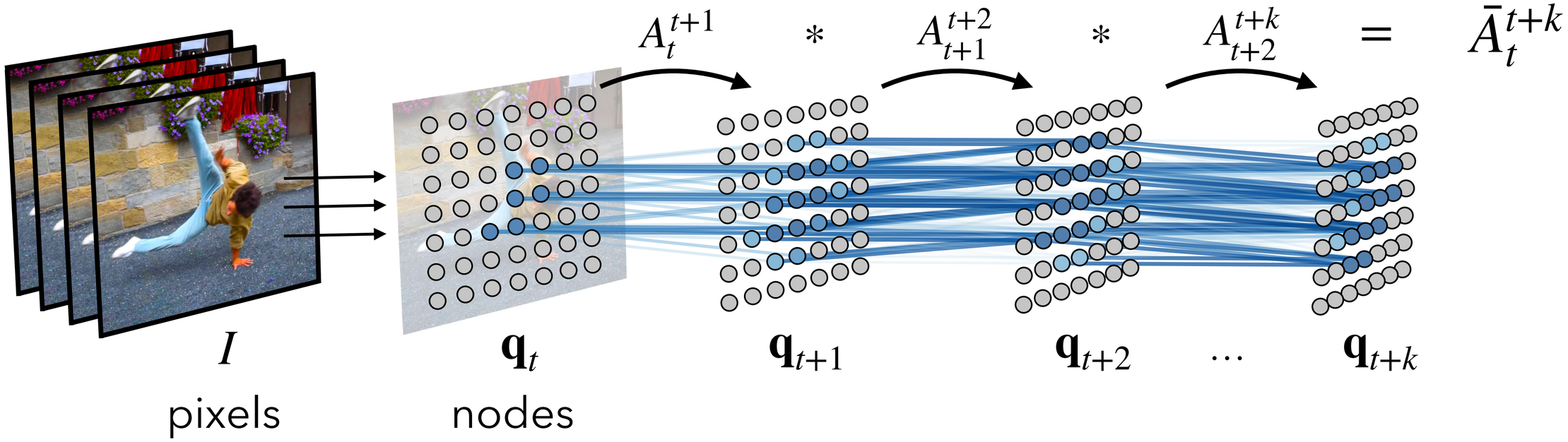


$$A_{ij} = \frac{e^{d_\phi(q_t^i,\ q_{t+1}^j)/\tau}}{\sum_l e^{d_\phi(q_t^i,\ q_{t+1}^l)/\tau}}$$

$$= P(X_{t+1} = j \mid X_t = i)$$

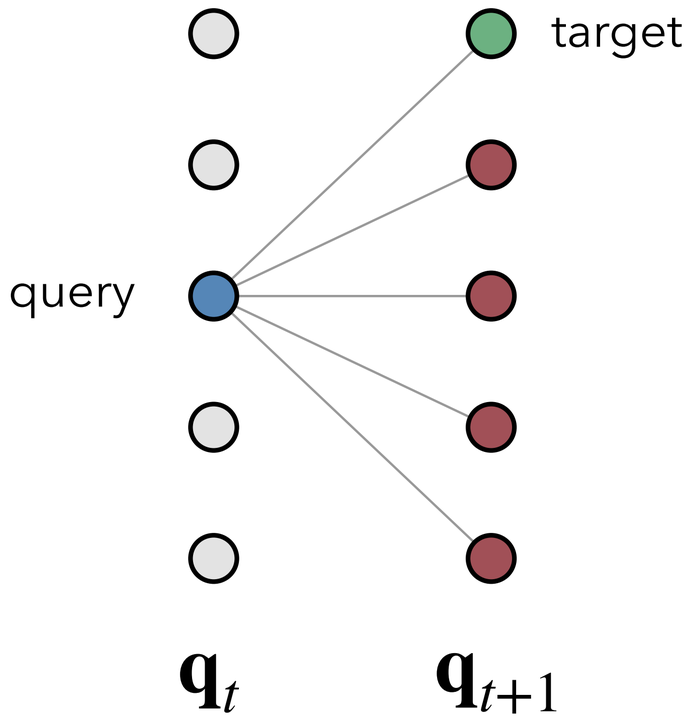
where $d_\phi(x, y) = \phi(x)^\top \phi(y)$

X_t is the position of walker at time t



Learn representation ϕ = Fit transition probabilities $\bar{A}_t^{t+k}$

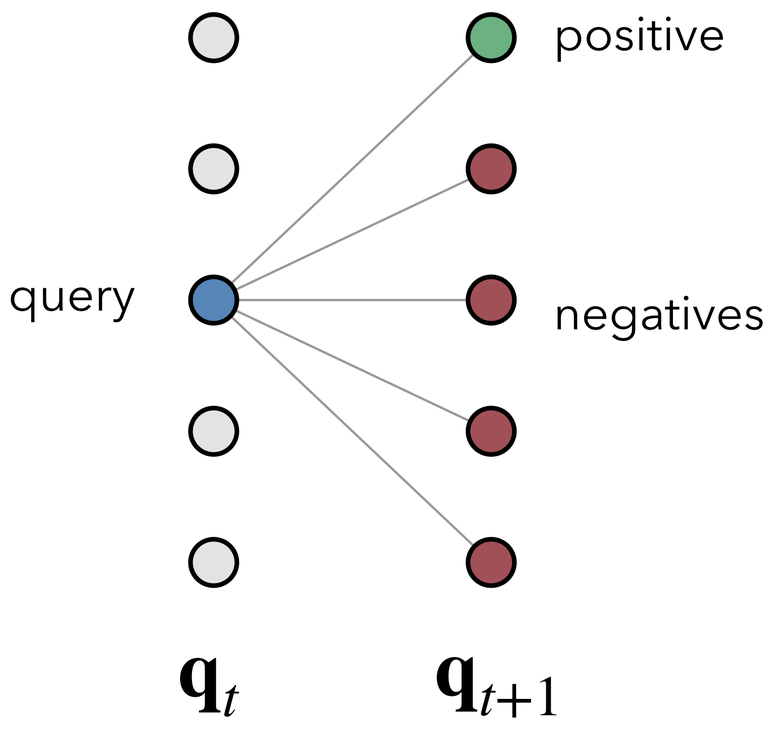
One Step



Maximize

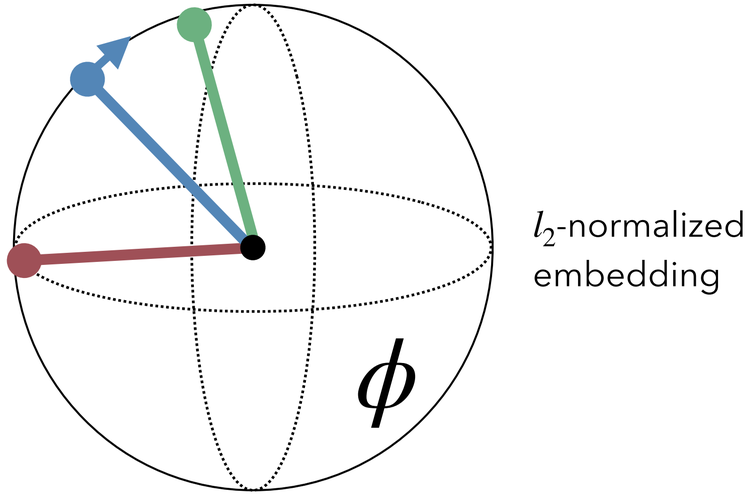
$$P(X_{t+1} = target \,|\, X_t = query)$$

One Step

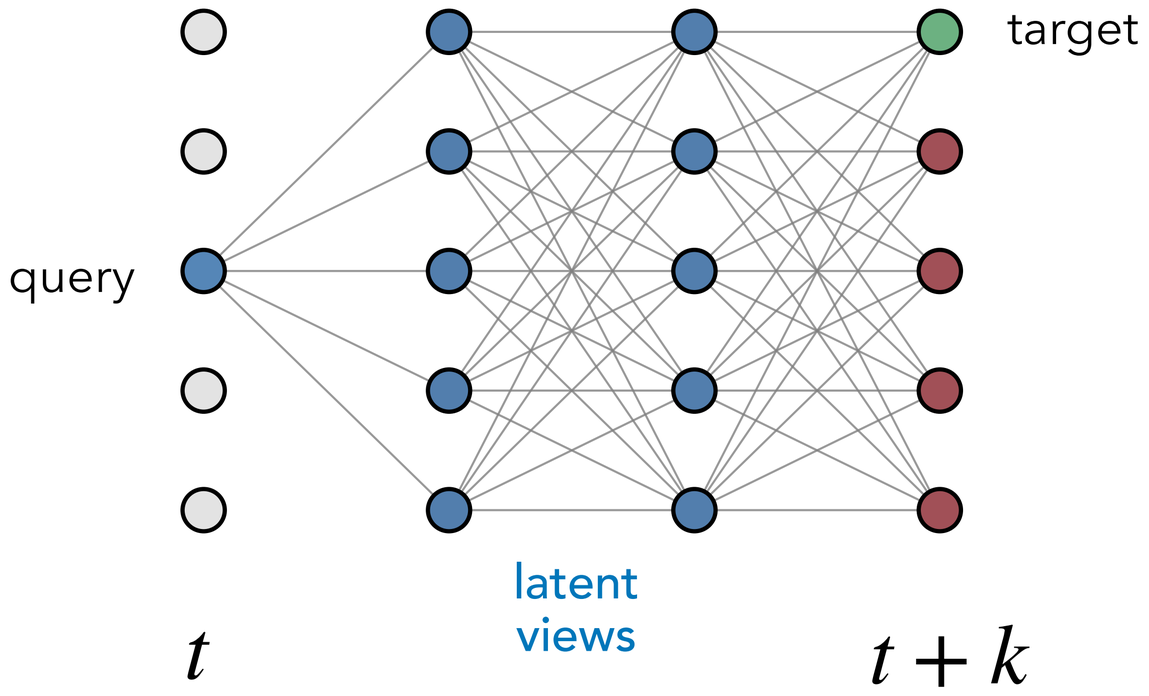


Maximize

$$P(X_{t+1} = \text{pos} \mid X_t = \text{query}) = \frac{e^{\phi(\text{query})^\top \phi(\text{pos})}}{\sum_{q_l} e^{\phi(\text{query})^\top \phi(q_l)}}$$



Chaining Correspondences



k-step Transition Matrix

$$\bar{A}_t^{t+k} = \prod_{i=0}^{k-1} A_{t+i}^{t+i+1}$$

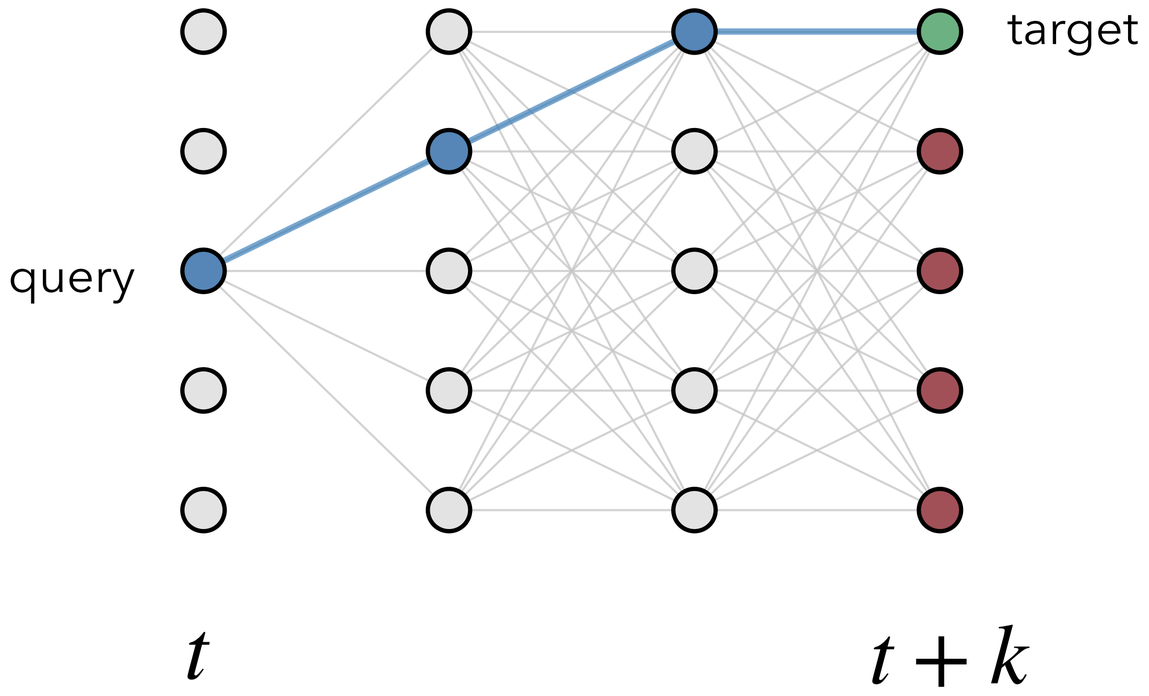
$$= P(X_{t+k} \mid X_t)$$

← Sum over intermediate time steps

Fitting $\bar{A}_t^{t+k}$ provides supervision

for $A_t^{t+1}, A_{t+1}^{t+2} \ldots, A_{t+k-1}^{t+k}$

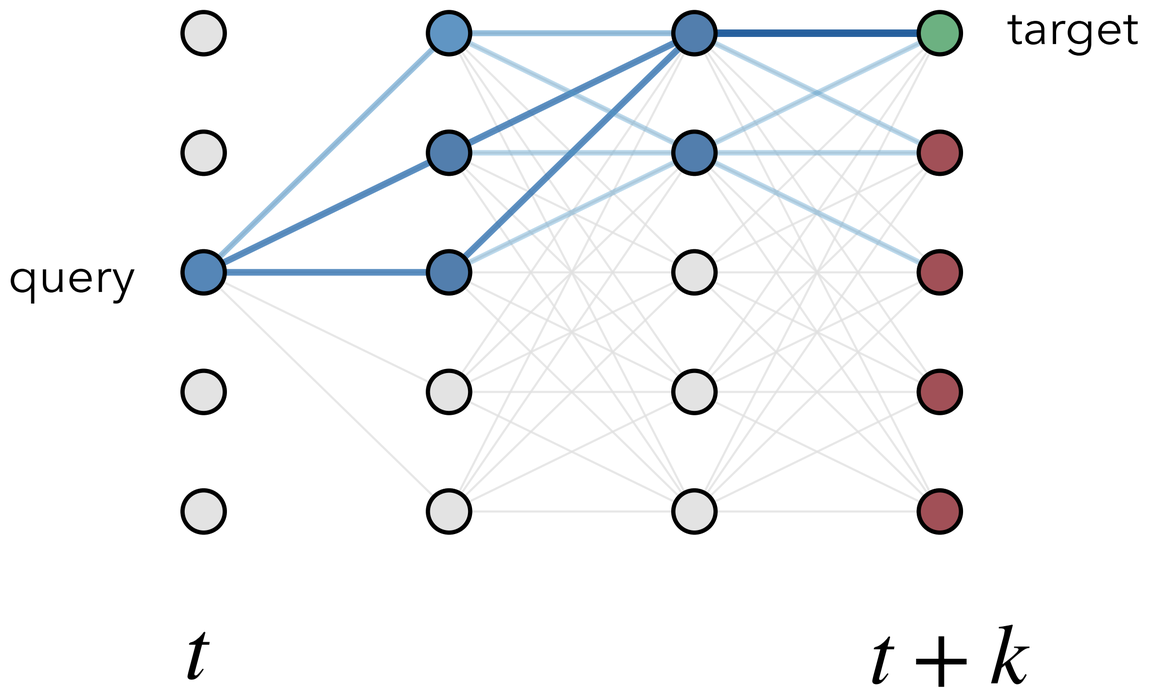
Chaining Correspondences



Easy case:

Obvious path

Chaining Correspondences

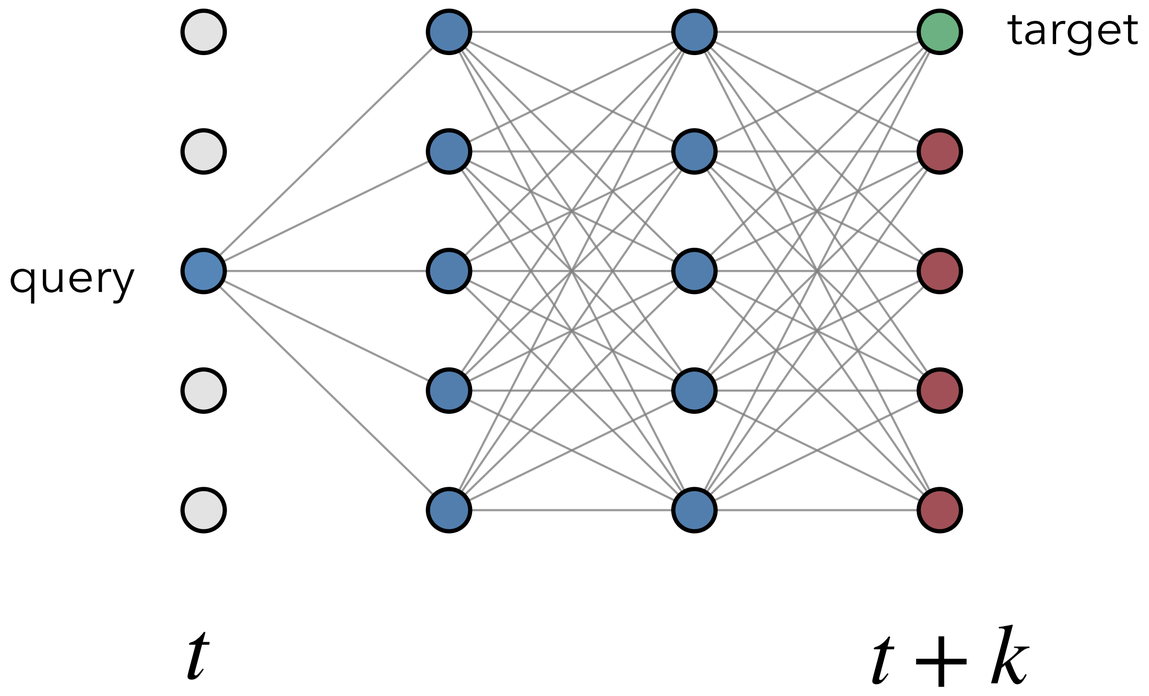


Harder case:

Ambiguity

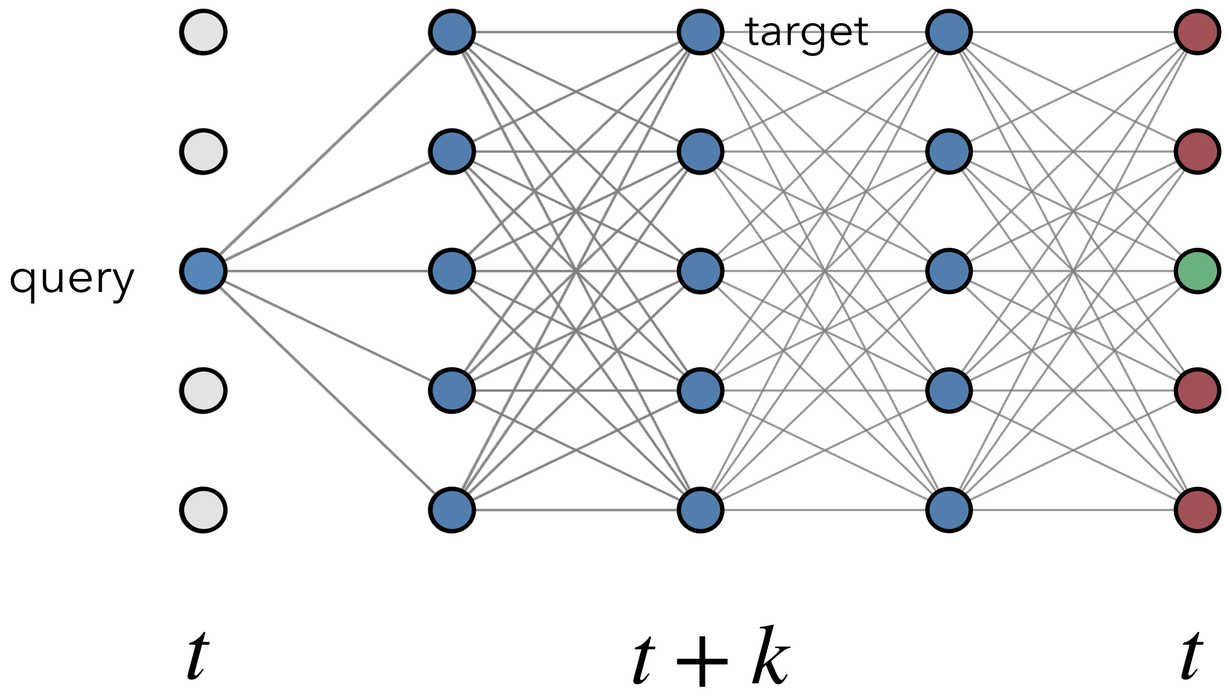
Consider multiple paths!

Chaining Correspondences



A single target supervises chains of learning problems

Supervised → Self-Supervised



Train on Palindromes

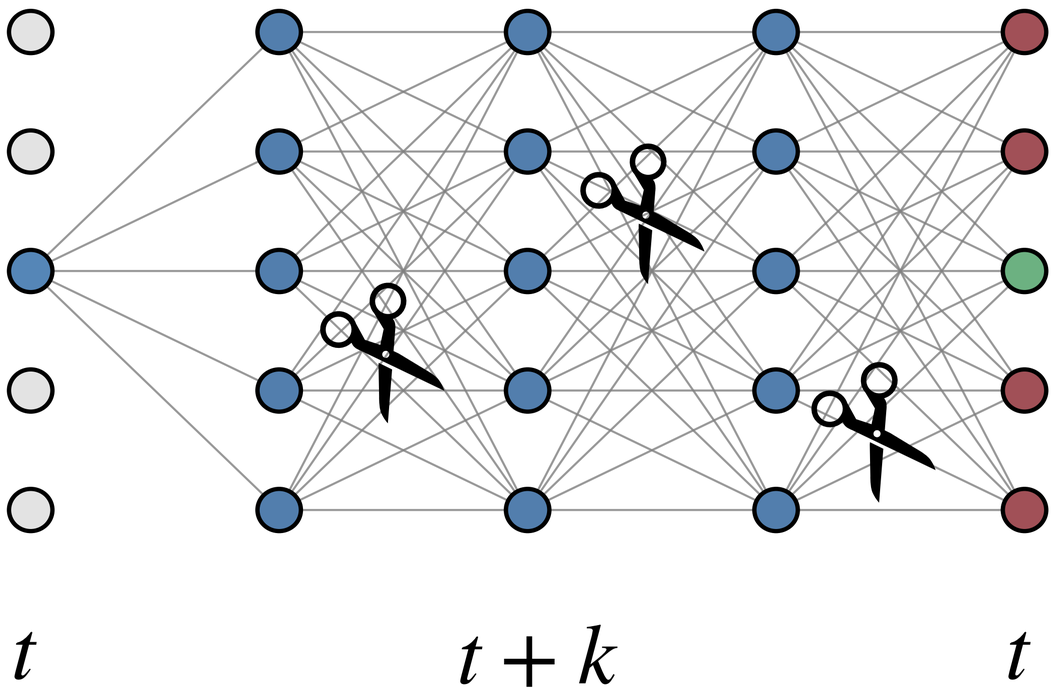
$$\mathcal{L}_{cyc}^{k} = \mathcal{L}_{CE}(\bar{A}_{t}^{t+k}\bar{A}_{t+k}^{t}, I)$$

Image
w/ query

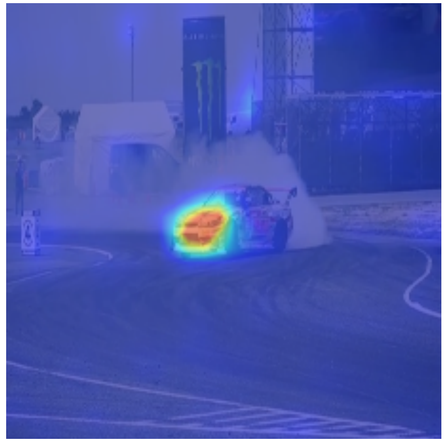
Transition
to future frame

Higher-level
Correspondence?

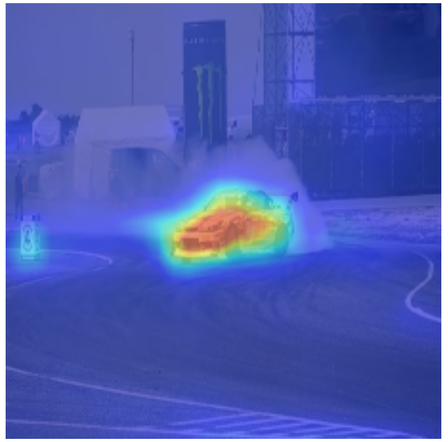
Edge Dropout



Force alternate, **context** paths

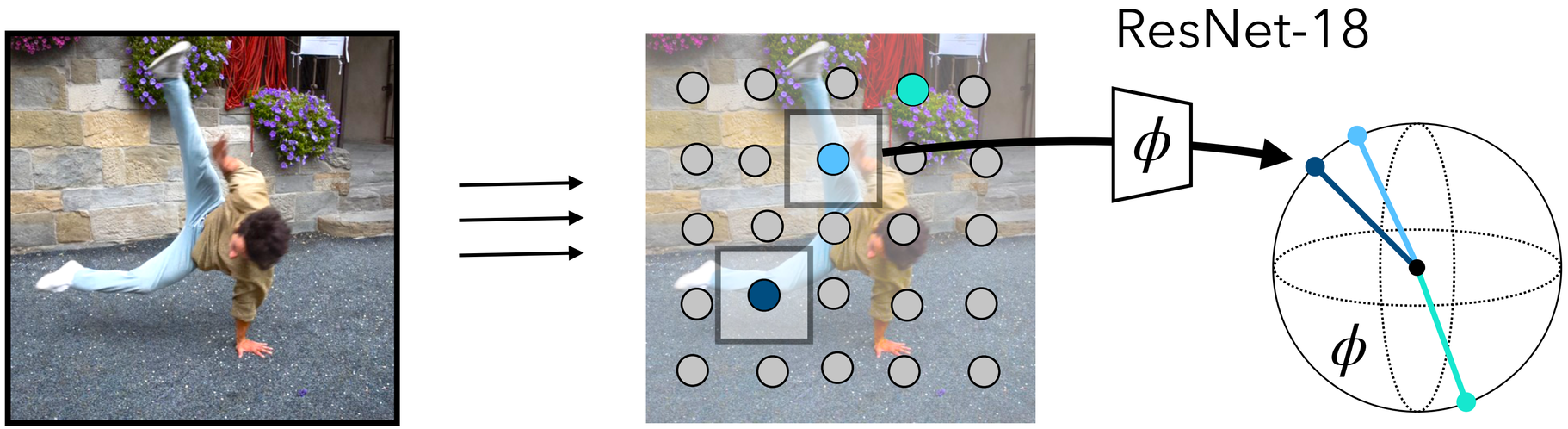


No Edge Dropout



Edge Dropout
p = 0.1

Implementation



frame
256 x 256 px

nodes
64 x 64 px
patches

embeddings
$\in \mathbb{R}^{128}$

Train on *Kinetics* dataset without labels

The Kinetics Human Action Video Dataset. Kay et al, 2017.

Evaluation: Using ϕ for Label Propagation

Object Propagation 1-4 Objects **DAVIS Benchmark**

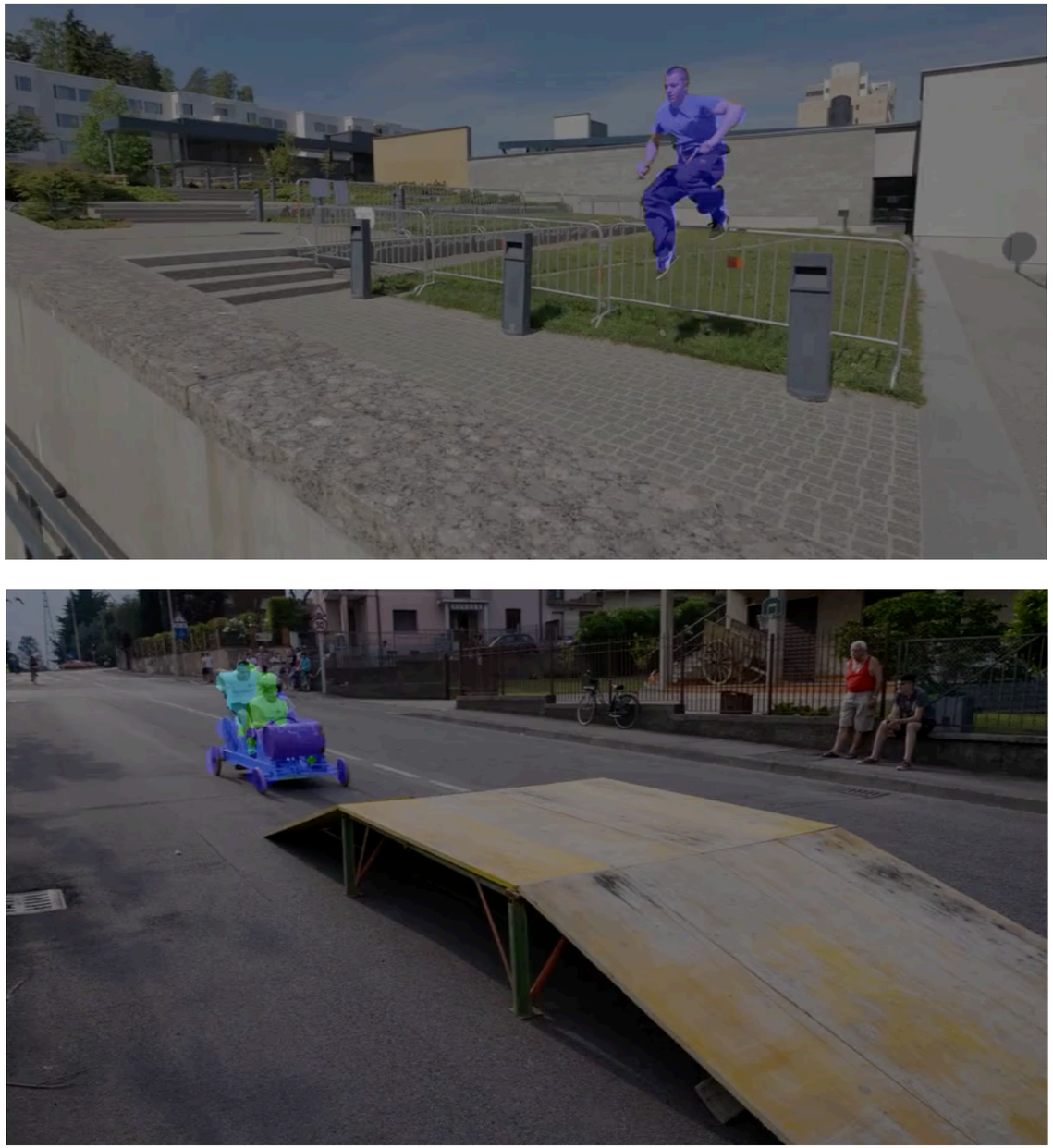
Semantic Part Propagation 20 Parts **VIP Benchmark**

Pose Propagation 15 Keypoints **JHMDB Benchmark**

Qualitative Results: Video Object Propagation (DAVIS)

UVC
Li et al. (2019)

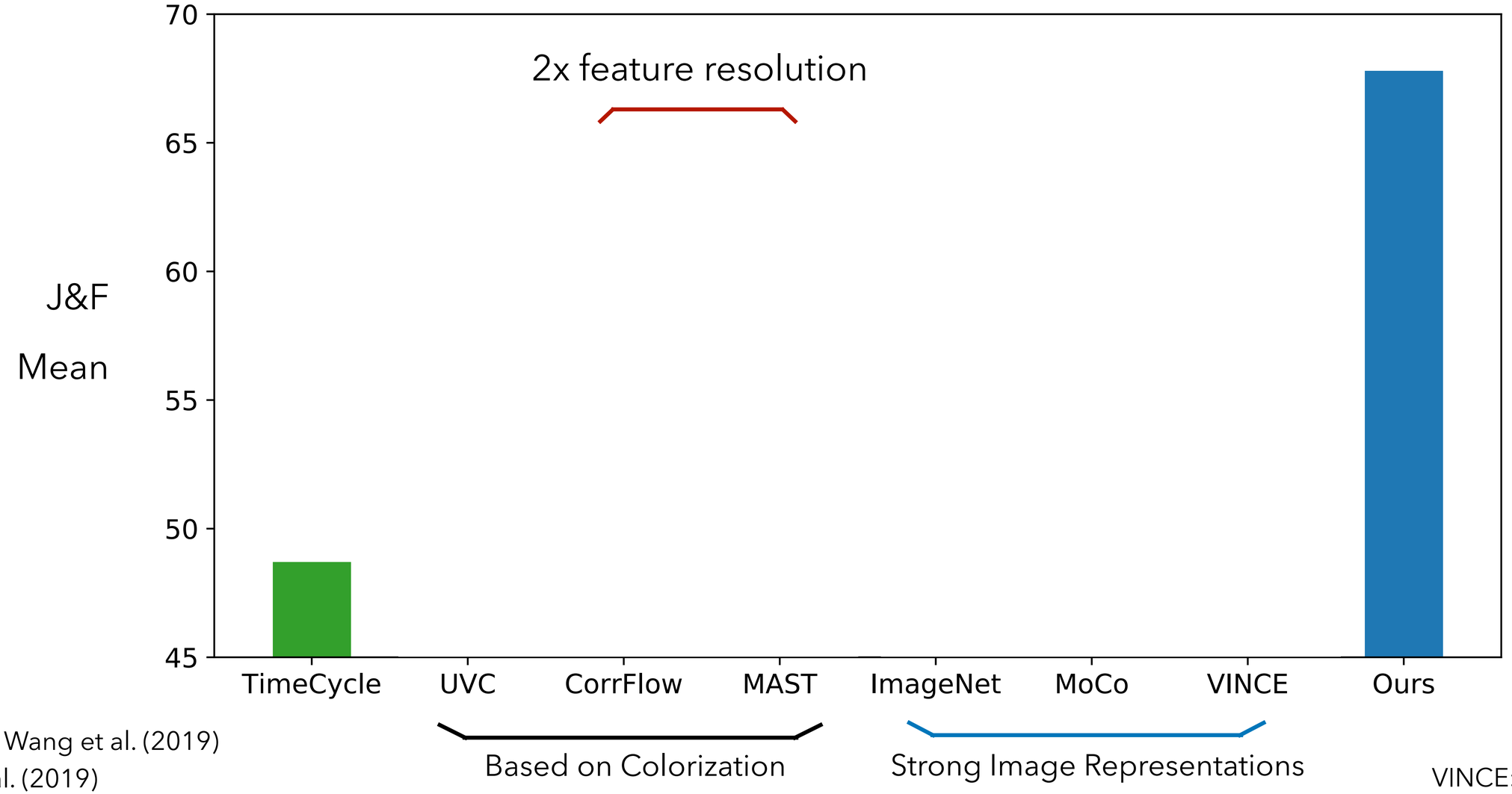
Ours



UVC
Li et al. (2019)

Ours

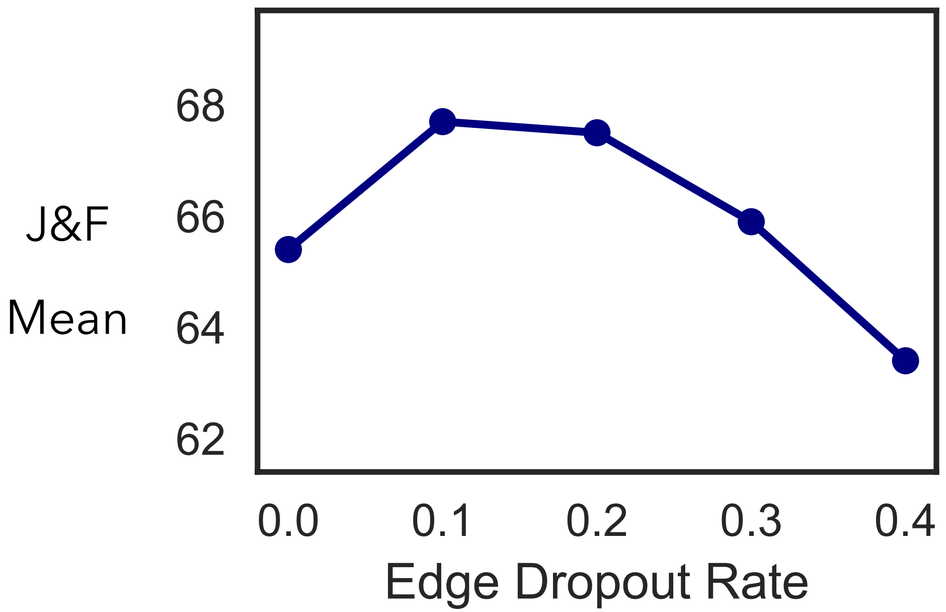
DAVIS Video Object Segmentation



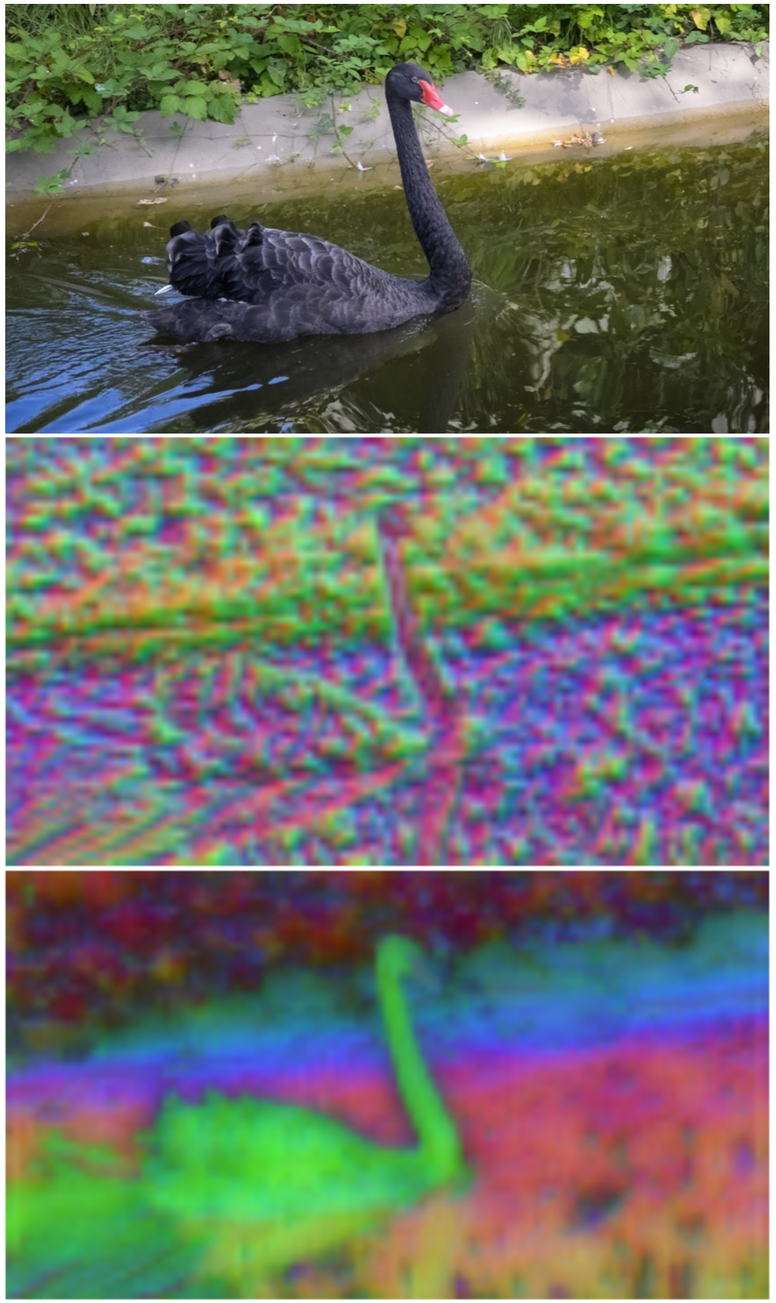
TimeCycle: Wang et al. (2019)
UVC: Li et al. (2019)
CorrFlow: Lai et al. (2019)
MAST: Lai et al. (2019)

VINCE: Gordon et al. (2020)
MoCo: He et al. (2019)

Effect of Edge Dropout



PCA Feature Visualization

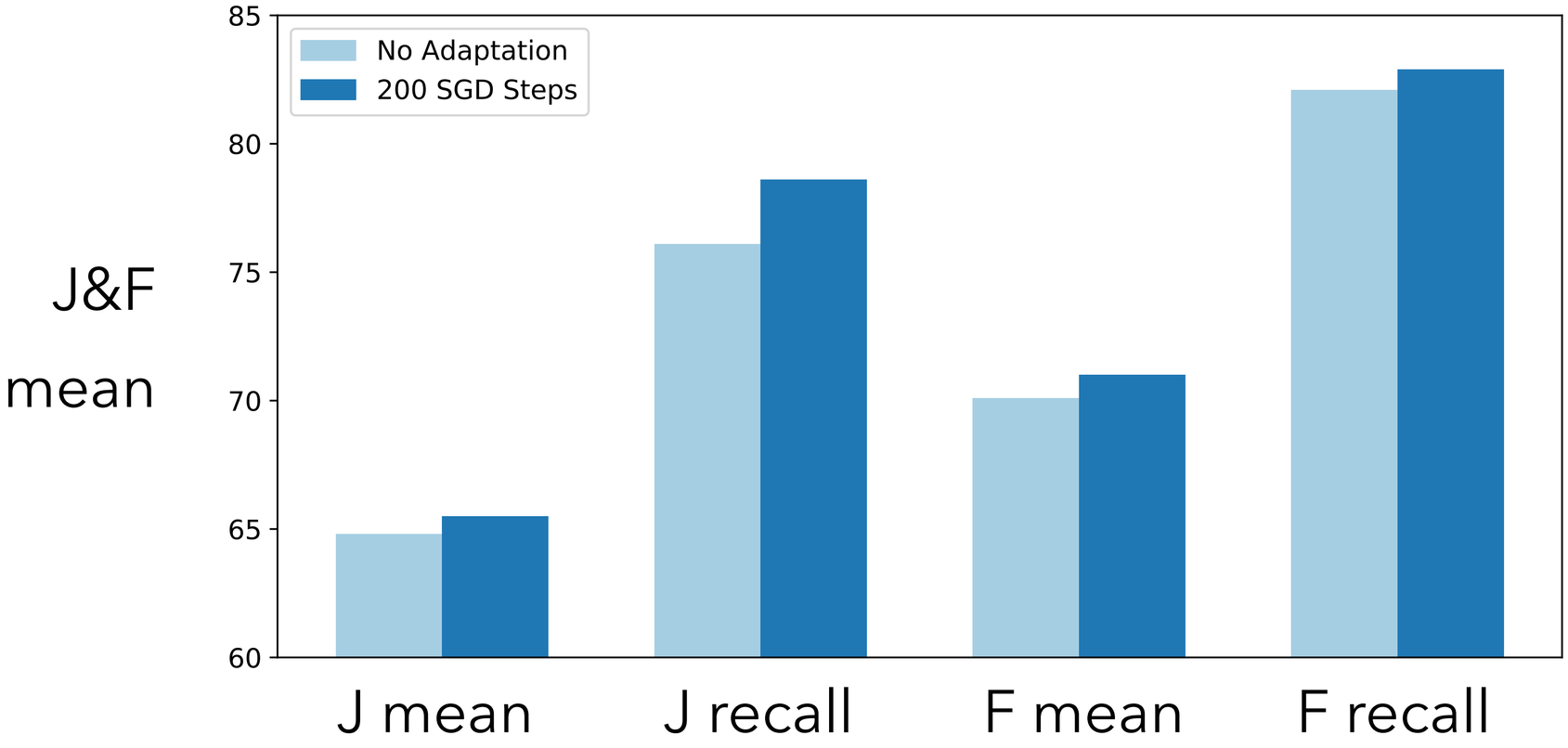


Image

No Edge Dropout

Edge Dropout 0.1

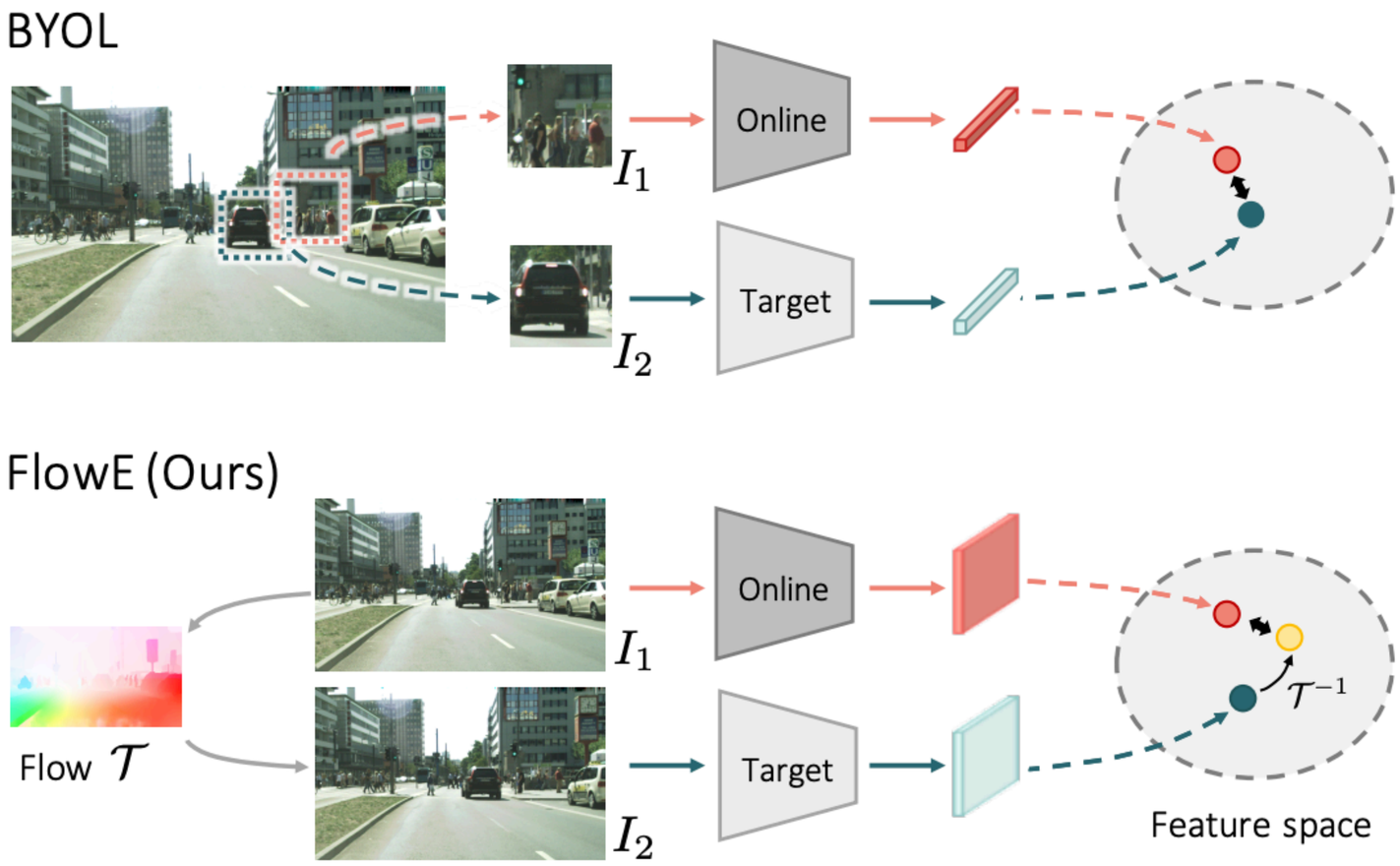
Test-time Adaptation (DAVIS)



Keypoints (JHMDB)

Semantic Parts (VIP)

Another way of getting views for contrastive learning: optical flow



Yuwen Xiong, Mengye Ren, Wenyuan Zeng, Raquel Urtasun. Self-Supervised Representation Learning from Flow Equivariance. arXiv 2021.

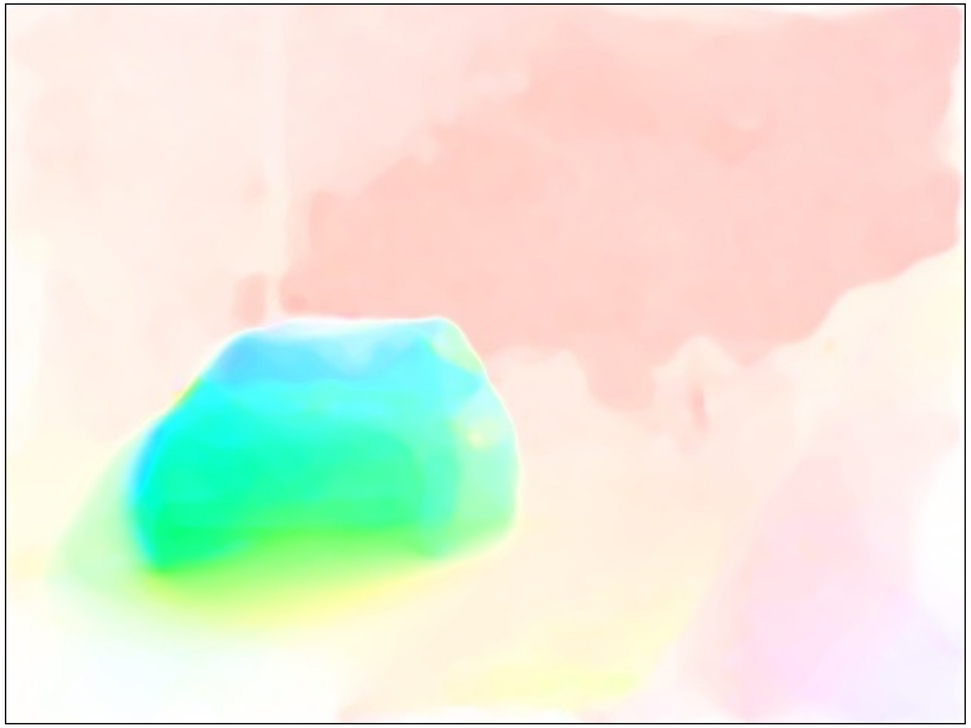
Discussion

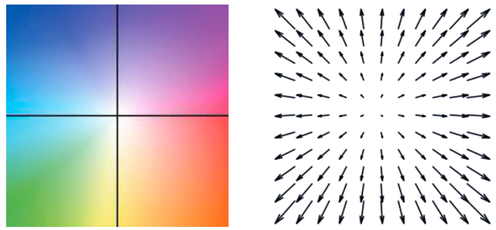
1. This paper suggests that there is a difference between category-level vs. instance-level unsupervised learning. Is this an important distinction?

2. Is this motion representation actually useful? What are its limitations?

3. Trivial cycles are possible if positional information is available. Why is this? Is this a big limitation?

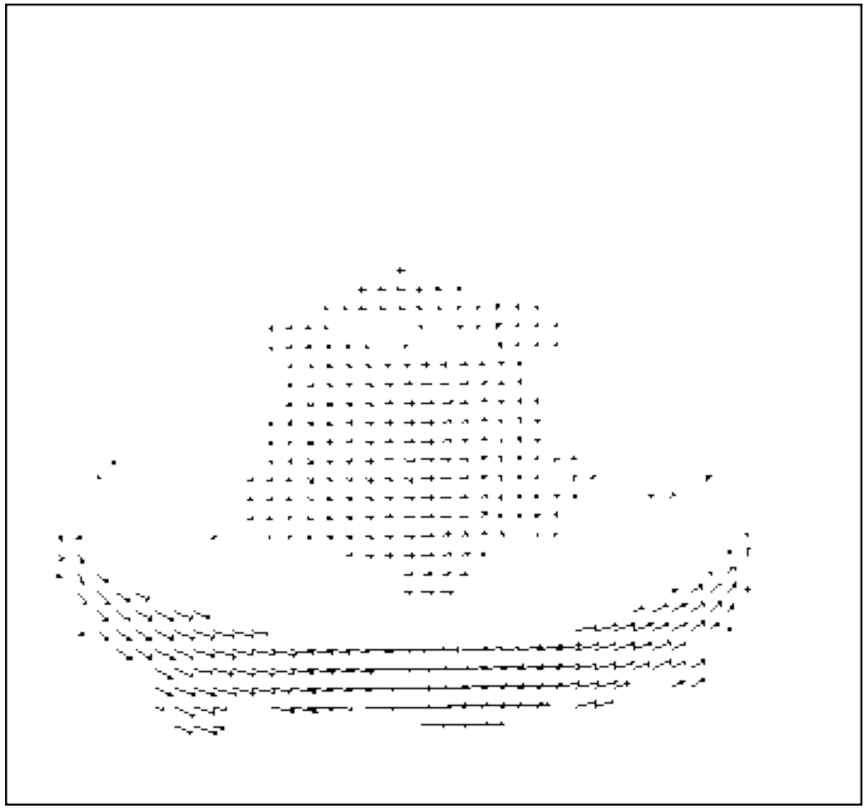
What we want: dense motion fields





Flow visualization

What we want: dense motion fields



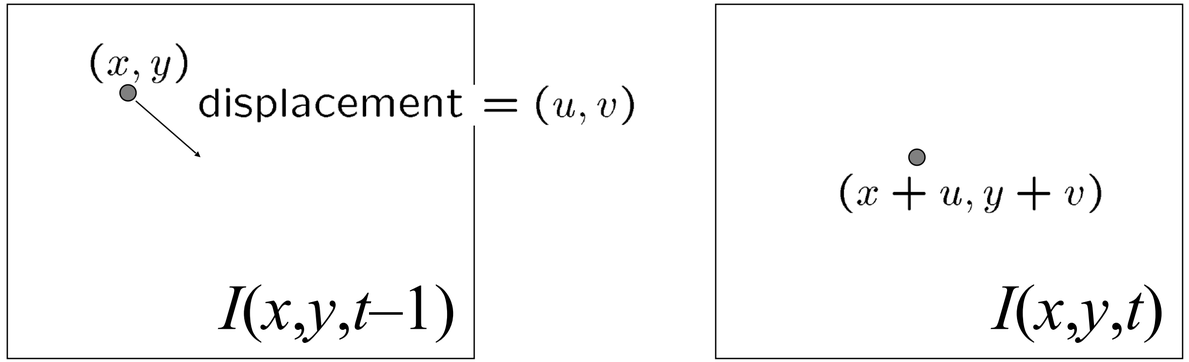
Source: S. Lazebnik

What Matters in Unsupervised Optical Flow

Rico Jonschkowski[1,2], Austin Stone[1,2], Jonathan T. Barron[2],
Ariel Gordon[1,2], Kurt Konolige[1,2], and Anelia Angelova[1,2]

[1]Robotics at Google and [2]Google AI

The brightness constancy constraint



Simple loss function. Find flow that minimizes:

$$\mathcal{L}(u, v) = \sum_{x,y} [I(x, y, t-1) - I(x + u(x, y), y + v(x, y), t)]^2$$

Why does this fail?

Source: S. Lazebnik

Photometric cost + smoothness prior

Goal: minimize matching error + smoothness [Horn and Schunck 1981]

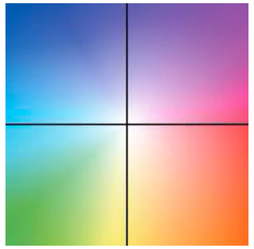
$$\underbrace{\sum_{x,y}[I(x,y,t-1) - I(u(x),v(y),t)]^2}_{E_d(u,v)\ \text{match cost}} + \underbrace{\sum_{p}\sum_{p'\in\mathcal{N}}(u(p)-u(p'))^2 + (v(p)-v(p'))^2}_{E_s(u,v)\ \text{smoothness}}$$

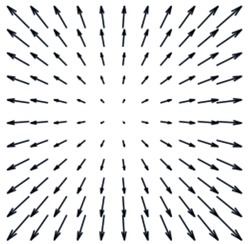
where p and p’ are neighboring pixels

Smoothness assumption

Input two frames

Horn-Schunck



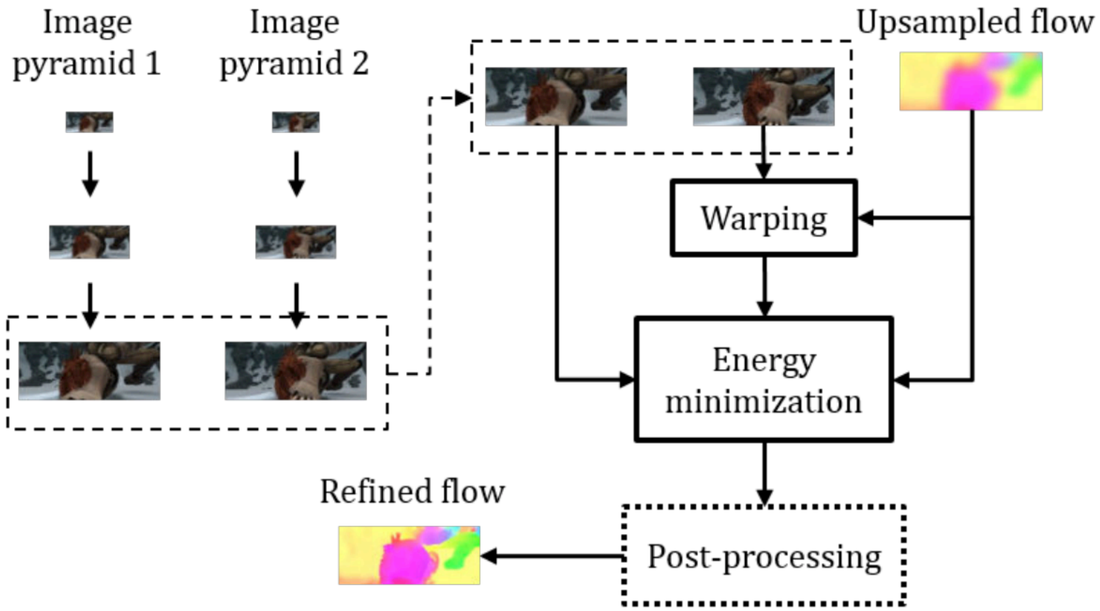


Flow visualization

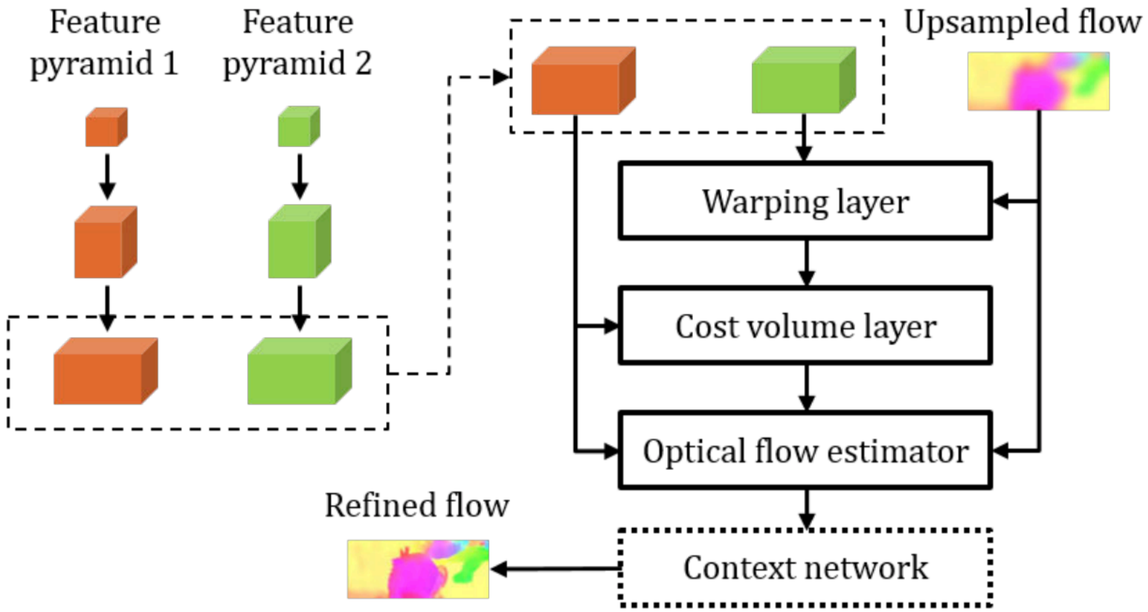
Coarse-to-fine LK

Source: Ce Liu

Flow CNNs: match features instead of pixels



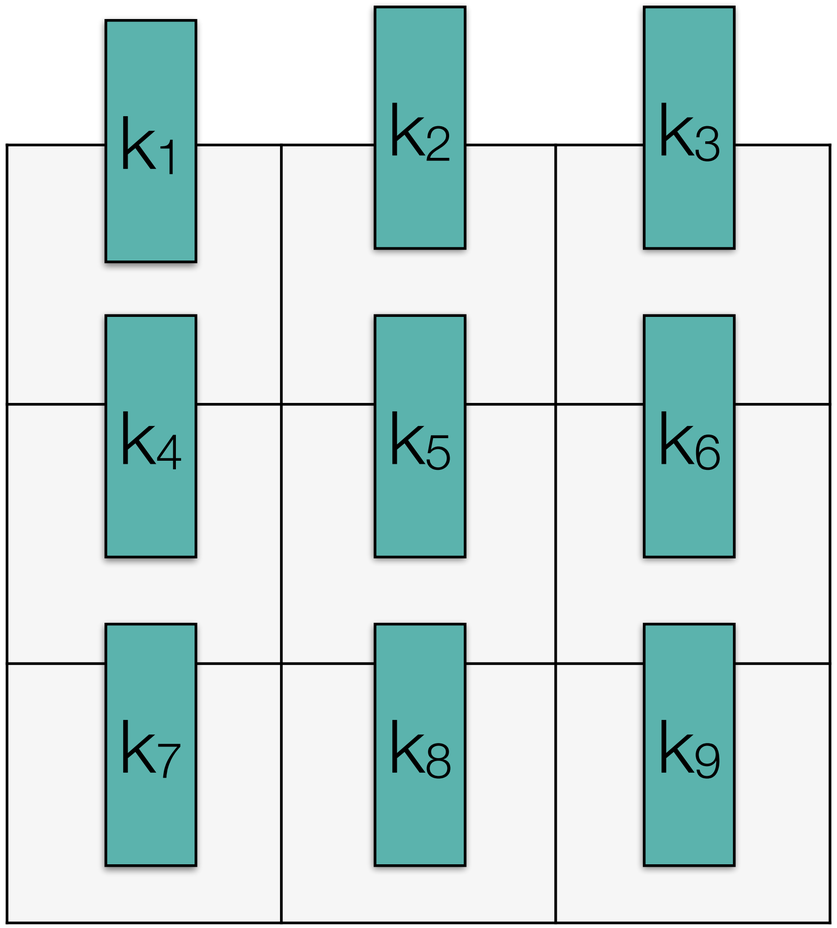
Traditional coarse-to-fine flow



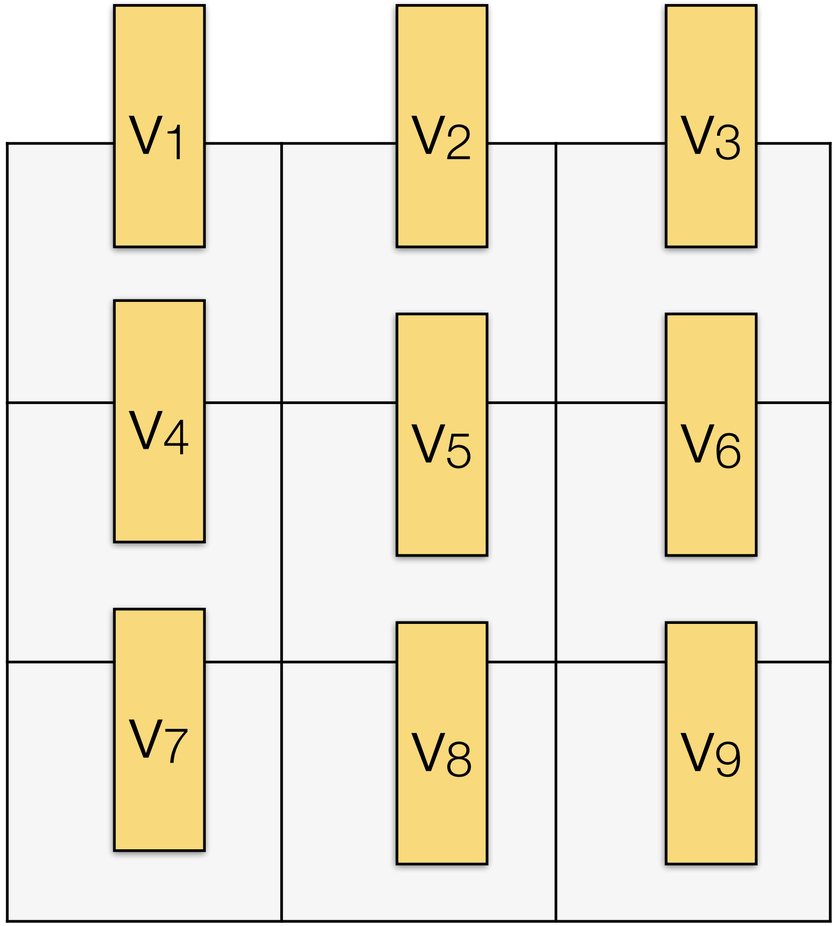
PWC-net

[Sun et al., “PWC-Net”, 2018]

Correlation between CNN features

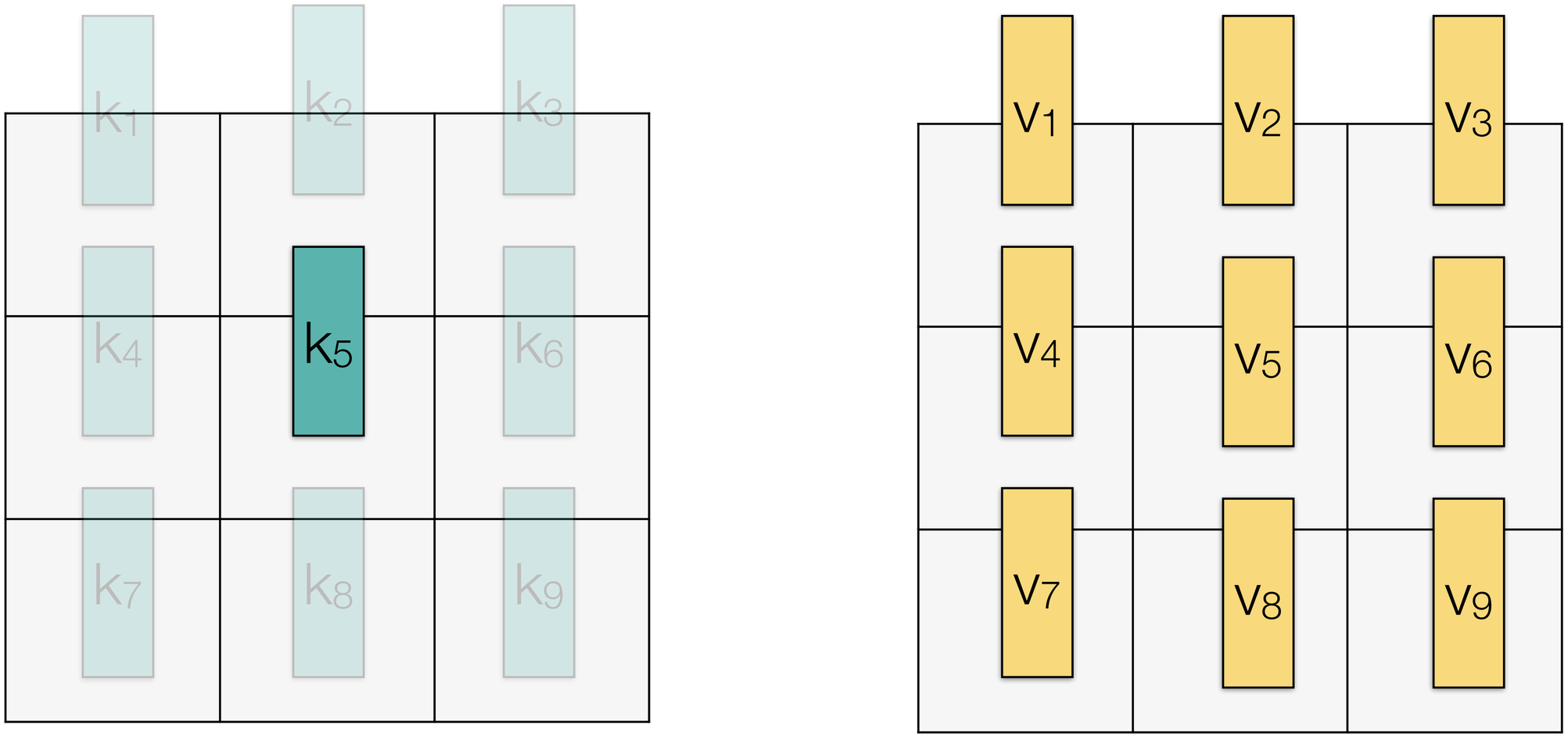


CNN feature map for I_1



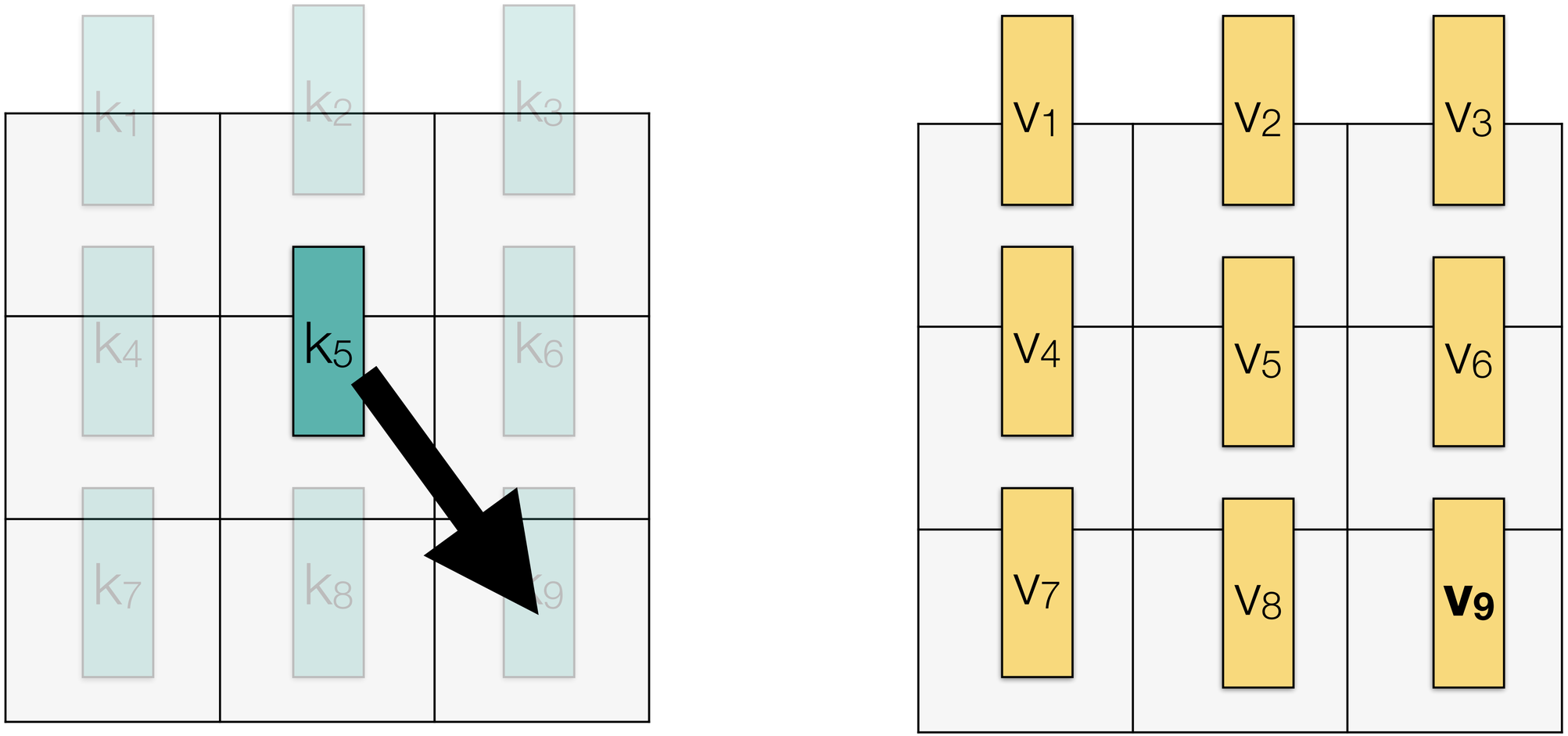
CNN feature map for I_2

Correlation between CNN features



Take dot product between features to get a **cost volume.**

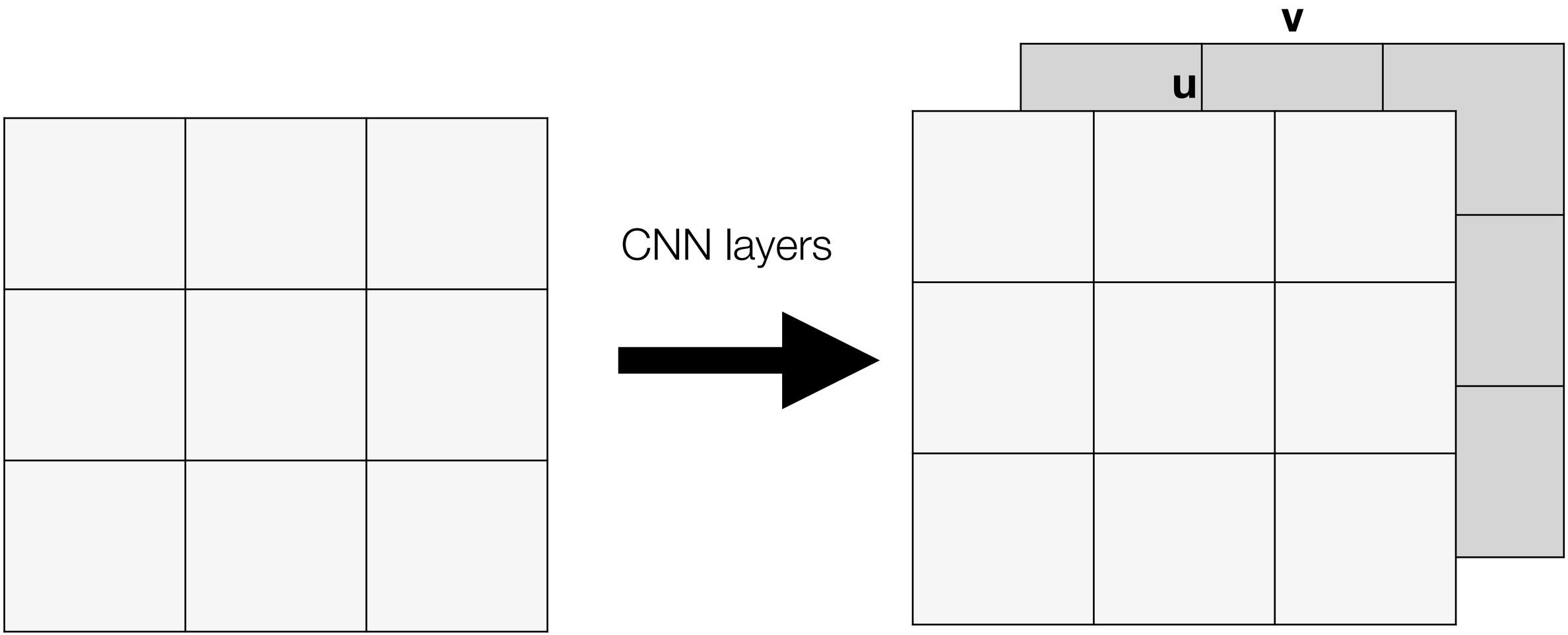
Correlation between CNN features



Can choose the match with **largest dot product**.

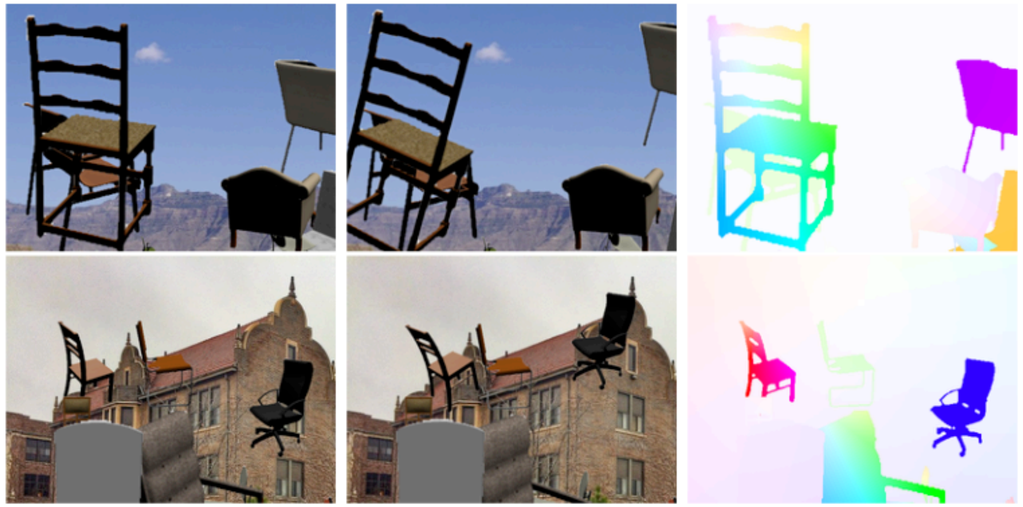
$u = 1,\ v = 1$

Often a regression step on top

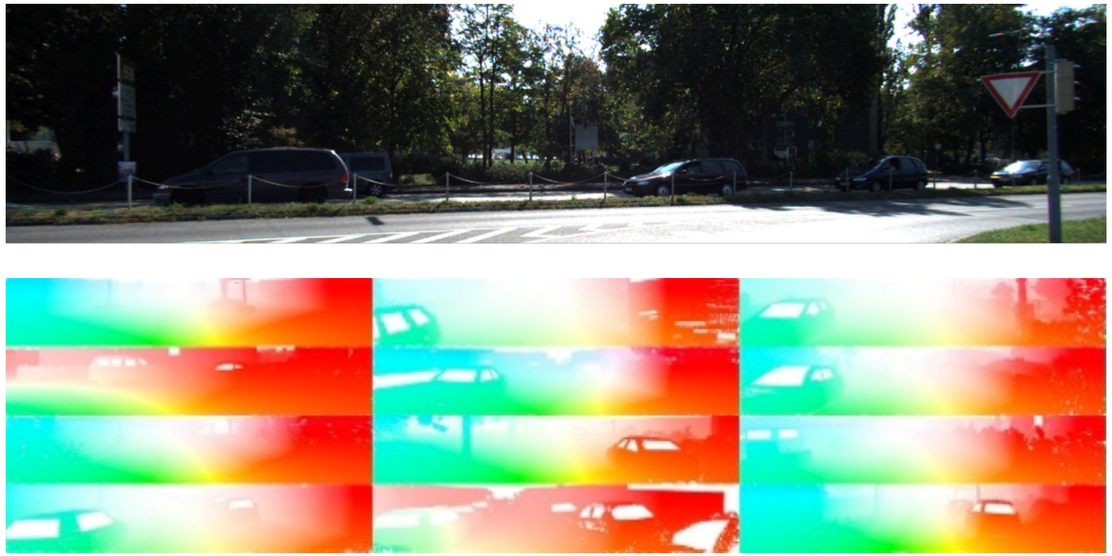


Alternatively, **regress** flow from cost volume.

Problem: hard to get supervision!



Flying Chairs dataset [Fischer et al., 2015]



KITTI dataset [Geiger et al.]

Less than 1K images!

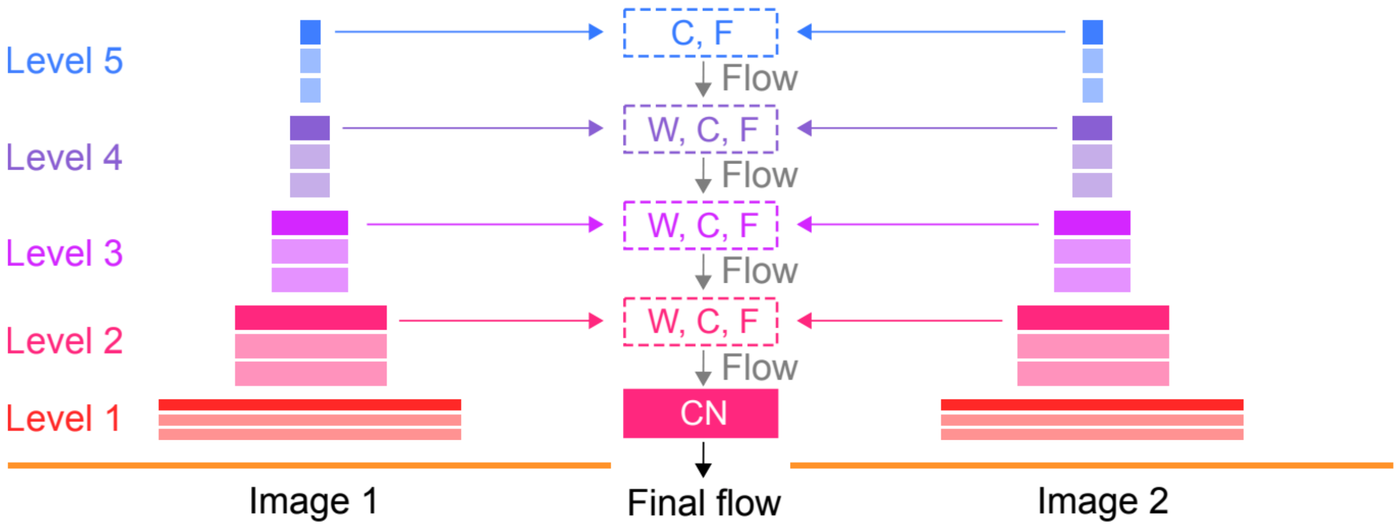
Learning with photometric cost + smoothness prior

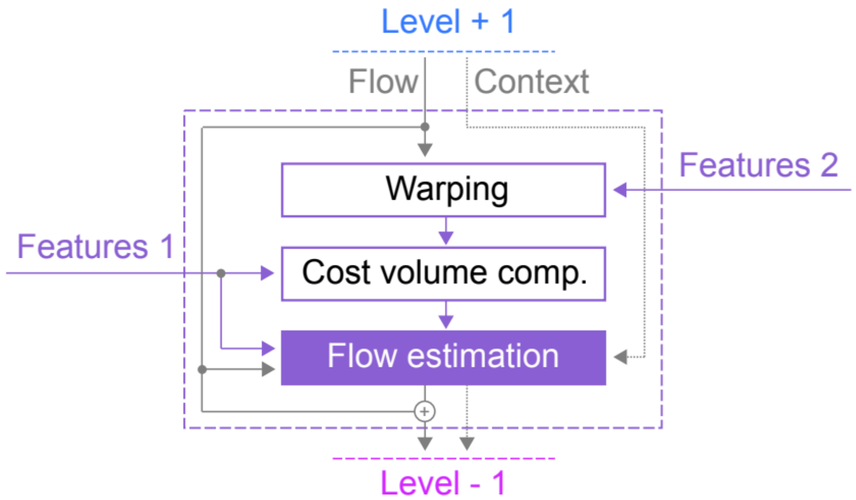
The flow you generate should have the following properties:

1. Matched pixels should have similar color.
2. Should have spatial smoothness.
3. Special handling for pixel that don’t have a match (e.g. occlusions)

This paper: experimental comparison of different ways of achieving these goals. Plus some training tricks.

Network design





Idea #1: Network design

Problem: Coarse-to-fine matching provides poor learning signal in coarse layers. Features obtain very low norm.

Solution: Normalize the features, “cost volume normalization” (similar to batch/instance normalization).

$$C^{(\ell)}_{x,y,u,v} = \sum_d \left(\frac{F^{(1,\ell)}_{x,y,d} - \mu^{(1,\ell)}}{\sigma^{(1,\ell)}} \right) \left(\frac{F^{(2,\ell)}_{x+u,y+v,d} - \mu^{(2,\ell)}}{\sigma^{(2,\ell)}} \right)$$

Other changes: Fewer layers and channels, randomly “drop out” flow between levels by passing it through.

Idea #1: Network design

Table 2. Model improvements. CVN: cost volume normalization, LD: level dropout

			Chairs	Sintel *train*		KITTI-15 *train*		
	CVN	LD	*test*	Clean	Final	all	noc	ER%
Minimal	–	–	5.01	4.52	6.67	13.30	2.72	21.69
	–	✓	5.29	4.40	**6.59**	**12.75**	2.49	21.30
	✓	–	4.86	**4.19**	6.69	13.294	2.59	21.54
	✓	✓	**3.58**	4.20	6.80	13.07	**2.47**	**21.21**
Full	–	–	3.78	3.41	4.70	39.09	30.19	98.77
	–	✓	3.21	3.45	4.61	2.96	**1.96**	9.77
	✓	–	**2.54**	3.07	4.31	3.16	2.04	10.35
	✓	✓	2.55	**3.00**	**4.18**	**2.94**	1.98	**9.65**

Idea #2: Photometric consistency

Charbonnier loss:

$$\mathcal{L}_C = \frac{1}{n} \sum \left((I^{(1)} - w(I^{(2)})^2 + \epsilon^2 \right)^\alpha$$

with

$$\epsilon = 0.001 \text{ and } \alpha = 0.5$$

Census transform: look at relative differences in brightness, using Hamming distance between binarized features

124	74	32
124	64	18
157	116	84

$\longrightarrow$

1	1	0
1	x	0
1	1	1

[Example source: Wikipedia]

Other choices: L1 distance and SSIM

Idea #2: Photometric consistency

Table 3. Photometric losses. Best results of L1 and Charbonnier underlined

		Chairs	Sintel *train*		KITTI-15 *train*		
	Method	*test*	Clean	Final	all	noc	ER%
Minimal	L1	4.27	5.51	7.74	17.02	6.11	32.96
	Charbonnier	4.31	5.50	7.64	16.94	6.09	32.84
	SSIM	**3.51**	**4.01**	**5.41**	11.99	2.46	21.72
	Census	3.54	4.23	6.98	**11.66**	**2.37**	**21.15**
Full	L1	2.83	4.23	5.75	5.53	3.17	18.65
	Charbonnier	2.86	4.24	5.81	5.56	3.21	18.82
	SSIM	**2.54**	3.08	4.52	3.29	2.04	10.41
	Census	2.61	**3.00**	**4.20**	**3.08**	**2.01**	**10.01**

Idea #3: Occlusion estimation

Problem: Huge penalty if you compute photometric loss on occluded pixels, i.e. those that don't have a match.

Solution: Estimate which pixels are occluded, creating a mask. Only compute the photometric loss for non-occluded pixels.

Methods for doing this:

- Do a cycle-consistency check. Cycle-consistency failure = occlusion.
- Soft histogram. Count the number of pixels that map onto given pixel. Less than 1 full pixel = occlusion.

Important detail: Don't backpropagate through mask computation!

Idea #4: Smoothness

Use an edge-aware smoothness penalty:

$$\mathcal{L}_{smooth(k)} = \frac{1}{n} \sum \exp\left(-\frac{\lambda}{3} \sum_c \left|\frac{\partial I_c^{(1,\ell)}}{\partial x}\right|\right) \left|\frac{\partial^k V^{(1,\ell)}}{\partial x^k}\right| + \exp\left(-\frac{\lambda}{3} \sum_c \left|\frac{\partial I_c^{(1,\ell)}}{\partial y}\right|\right) \left|\frac{\partial^k V^{(1,\ell)}}{\partial y^k}\right|$$

- Note that they don't explicitly *blur*. The loss just encourages the network to learn smooth flow.
- They apply the smoothness loss to the final level (quarter resolution).
- They use different smoothness losses for each dataset (k=2 for KITTI, the "real" dataset, k=1 for synthetic)

Idea #5: Handling out-of-bounds motions

- Work on “self-supervised” flow creates synthetic training examples.
  1. Compute flow on two images
  2. Make the flow task harder via augmentation
  3. Encourage flow to be the same
- Name is a bit confusing, since self-supervised often is synonym for unsupervised…

Idea #5: Handling out-of-bounds motions

Idea #5: Handling out-of-bounds motions

1. Compute flow on full-res images
2. Remove 64-pixel border from each side
3. Compute flow on cropped images
4. Penalize flow differences
    - They use occlusion-weighted Charbonnier loss

Full loss

- Combine photometric loss, smoothness, and “self-supervision” losses

$$\mathcal{L}(D, \theta) = w_{photo} \cdot \mathcal{L}_{photo} + w_{smooth} \cdot \mathcal{L}_{smooth} + w_{self} \cdot \mathcal{L}_{self}$$

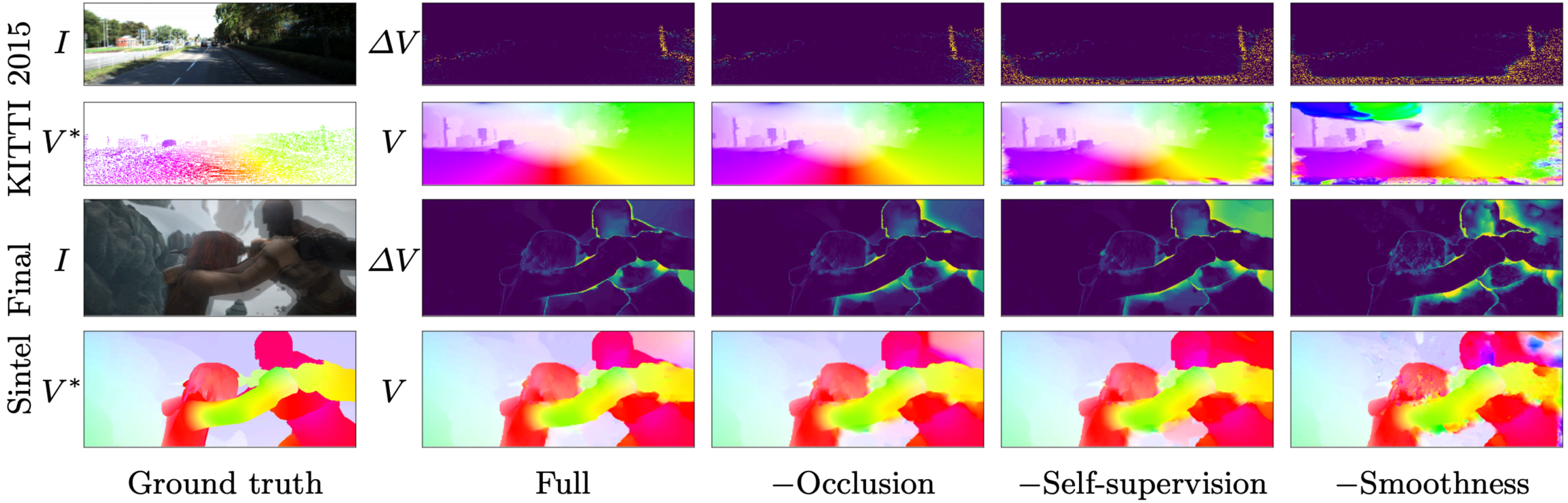
Major model components

Table 1. Core components: OM: occlusion masking, SM: smoothness, SS: self-supervision; "div.": divergence

			Chairs	Sintel *train*		KITTI-15 *train*		
OM	SM	SS	*test*	Clean	Final	all	noc	ER%
–	–	–	3.58	4.20	6.80	13.07	2.47	21.21
–	–	✓	2.99	3.34	5.18	11.36	2.30	18.61
–	✓	–	2.84	3.37	5.19	11.37	2.17	19.31
–	✓	✓	2.74	3.12	4.56	3.28	2.08	9.97
✓	–	–	3.28	3.78	5.85	div.	div.	div.
✓	–	✓	2.91	3.26	4.72	3.02	2.11	9.89
✓	✓	–	2.63	3.20	4.63	4.15	2.05	13.15
✓	✓	✓	**2.55**	**3.00**	**4.18**	**2.94**	**1.98**	**9.65**

- Consistent gains from smoothness
- Big gains for out-of-bounds pixels with self-supervision loss
- Occlusion masking also helps, especially for out-of-bounds pixels.
  - Can be a bit unstable though

Qualitative results



Comparison to other flow methods

		Method	Sintel Clean [4]		Sintel Final [4]		KITTI 2012 [6]		KITTI 2015 [19]			
			EPE		EPE		EPE		EPE	EPE (noc)	ER in %	
			train	*test*	*train*	*test*	*train*	*test*	*train*	*train*	*train*	*test*
Supervised	(A)	FlowNet2-ft [11]	(1.45)	4.16	(2.01)	5.74	(1.28)	1.8	(2.30)	–	(8.61)	11.48
	(B)	PWC-Net-ft [25]	(1.70)	3.86	(2.21)	5.13	(1.45)	1.7	(2.16)	–	(9.80)	9.60
	(C)	SelFlow-ft [15]	(1.68)	[3.74]	(1.77)	{4.26}	(0.76)	1.5	(1.18)	–	–	8.42
	(D)	VCN-ft [32]	(1.66)	2.81	(2.24)	4.40	–	–	(1.16)	–	(4.10)	6.30
	(E)	FlowNet2 [11]	**2.02**	**3.96**	**3.14**	**6.02**	**4.09**	–	9.84	–	28.20	–
	(F)	PWC-Net [25]	2.55	–	3.93	–	4.14	–	10.35	–	33.67	–
	(G)	VCN [32]	2.21	–	3.62	–	–	–	**8.36**	–	**25.10**	–
Unsupervised	(H)	Back2Basics [34]	–	–	–	–	11.30	9.9	–	–	–	–
	(I)	DSTFlow [22]	{6.16}	10.41	{7.38}	11.28	[10.43]	12.4	[16.79]	[6.96]	[36.00]	[39.00]
	(J)	OAFlow [30]	{4.03}	7.95	{5.95}	9.15	[3.55]	[4.2]	[8.88]	–	–	[31.20]
	(K)	UnFlow [18]	–	–	7.91	10.21	3.29	–	8.10	–	23.27	–
	(L)	GeoNet [33] (MDM)	–	–	–	–	–	–	10.81	8.05	–	–
	(M)	DF-Net [36] (MDM)	–	–	–	–	3.54	4.4	{8.98}	–	{26.01}	{25.70}
	(N)	CCFlow [21] (MDM)	–	–	–	–	–	–	5.66	–	20.93	25.27
	(O)	MFOccFlow [12] (MF)	{3.89}	7.23	{5.52}	8.81	–	–	[6.59]	[3.22]	–	22.94
	(P)	UnOS [29] (SDM)	–	–	–	–	1.64	**1.8**	5.58	–	–	18.00
	(Q)	EPIFlow [35]	3.94	7.00	5.08	8.51	2.61	3.4	5.56	2.56	–	16.95
	(R)	DDFlow [14]	{2.92}	6.18	{3.98}	7.40	[2.35]	3.0	[5.72]	[2.73]	–	14.29
	(S)	SelFlow [15] (MF)	[2.88]	[6.56]	{3.87}	{6.57}	[1.69]	2.2	[4.84]	[2.40]	–	14.19
	(T)	UFlow-test	**3.01**	–	**4.09**	–	**1.58**	–	**2.84**	**1.96**	**9.39**	–
	(U)	UFlow-train	{2.50}	**5.21**	{3.39}	**6.50**	1.68	1.9	{2.71}	{1.88}	{9.05}	**11.13**

Discussion

- How many of these hand-crafted loss functions generalize to new video domains?
- They still train on relatively small-scale datasets (100s of videos)

- What can we (and can’t we) learn about objects from motion?
- How do we integrate motion-based training into image-based unsupervised learning?