Lecture 8

Video Recognition, Optical Flow & Tracking

Slides from: Du Tran, Rick Szeliski, Steve Seitz, Christoph Feichtenhofer

Overview

- Applying ConvNets to Video
  - Classification
  - Dense prediction
- Optical Flow
- Two-stream ConvNets for Video
  - Classification
  - Tracking

Overview

- Applying ConvNets to Video
  - Classification
  - Dense prediction
- Optical Flow
- Two-stream ConvNets for Video
  - Classification
  - Tracking

Traditional Computer Vision Pipeline

Feature Extraction (e.g. color, texture, motions)

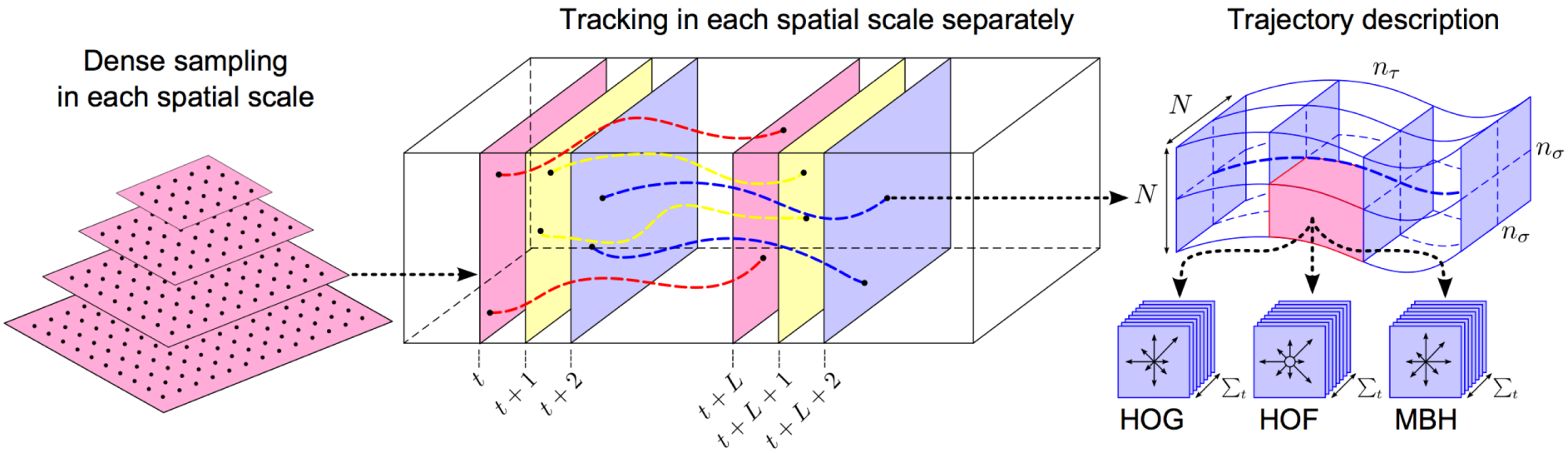
Prediction Models (e.g. SVM, HMM, Bayesian)

Predictions

Focus of this work

Best (non-DL) Video Features

- improved Dense Trajectories (iDT)



Wang et al. IJCV'13

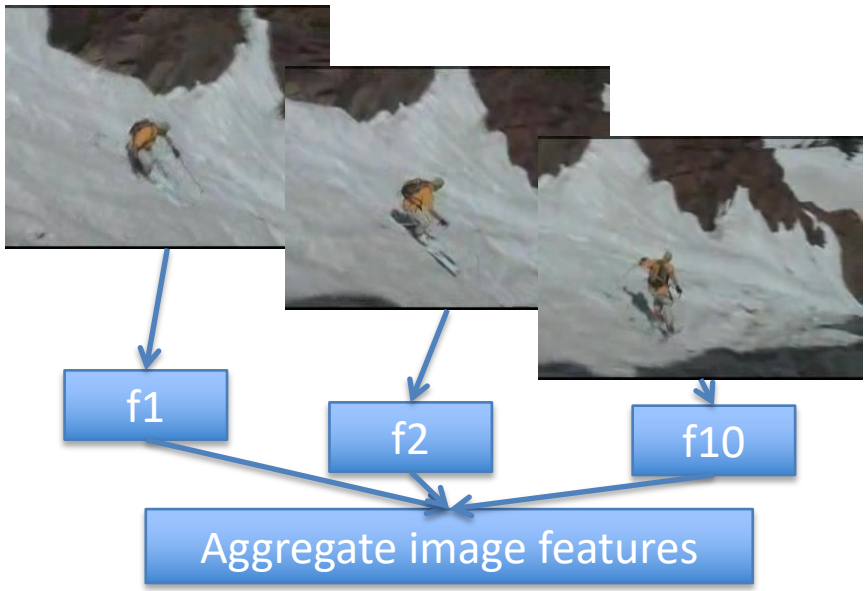
Pros:

- Don't need to learn
- Don't need large-scale training data

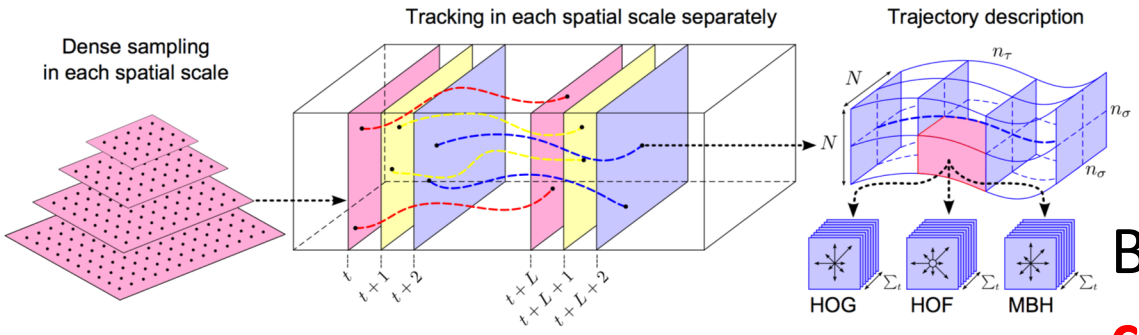
Cons:

- Highly hand-crafted
- Computational intensive
- Hard to parallelize

Spatiotemporal Feature Learning



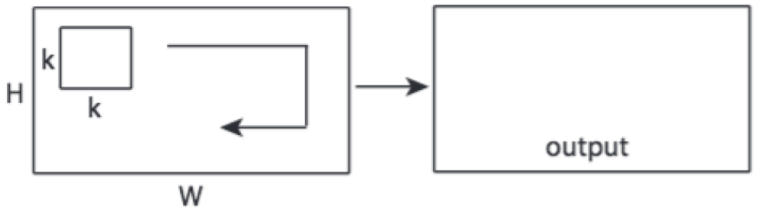
No explicit **motion modeling**



- Wang et al. IJCV'13

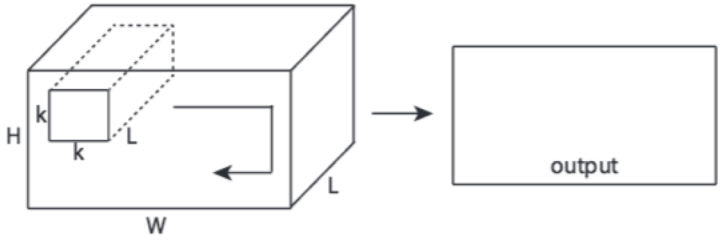
Biased to **human design** & **computationally expensive**

Why 3D ConvNets?



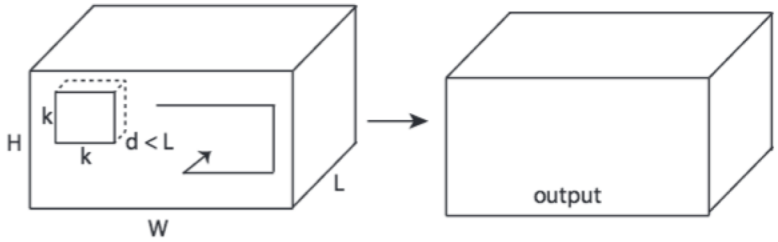
2D convolve on an image

-> no motion modeling ✖



2D convolve on multiple images as channels

-> collapse temporal signal after one convolution layer ✖



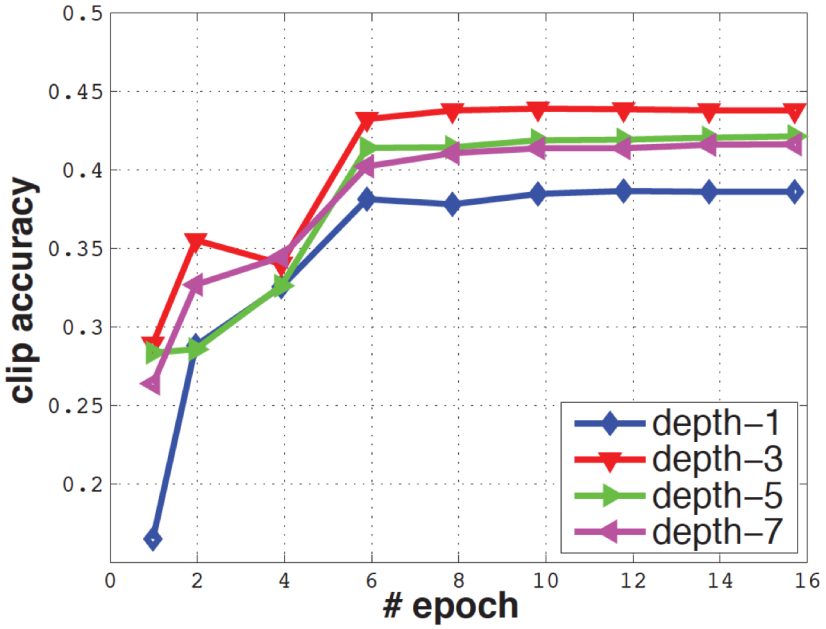
Spatial-temporally convolve on multiple frames

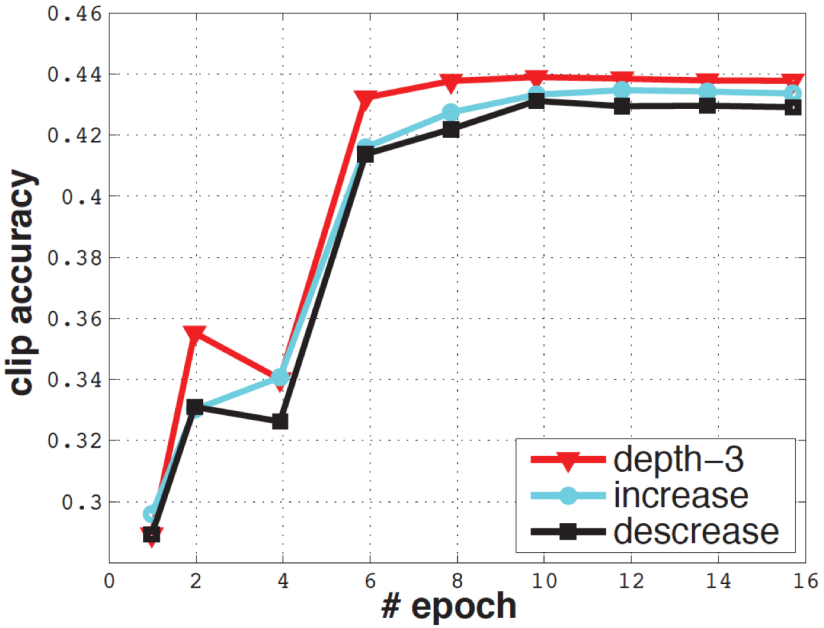
-> hierarchically group temporal signal ✔

What is a Good Architecture for 3D ConvNets?

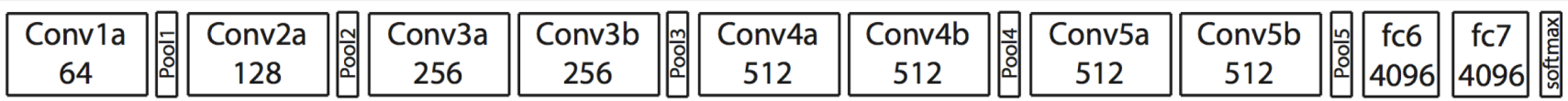
D. Tran, L. Bourdev, R. Fergus, L. Torresani, M. Paluri, ***Learning Spatiotemporal Features with 3D Convolutional Networks***, **ICCV15**.

- Dataset: UCF101
- Use VGG-similar architecture, varying kernel temporal length



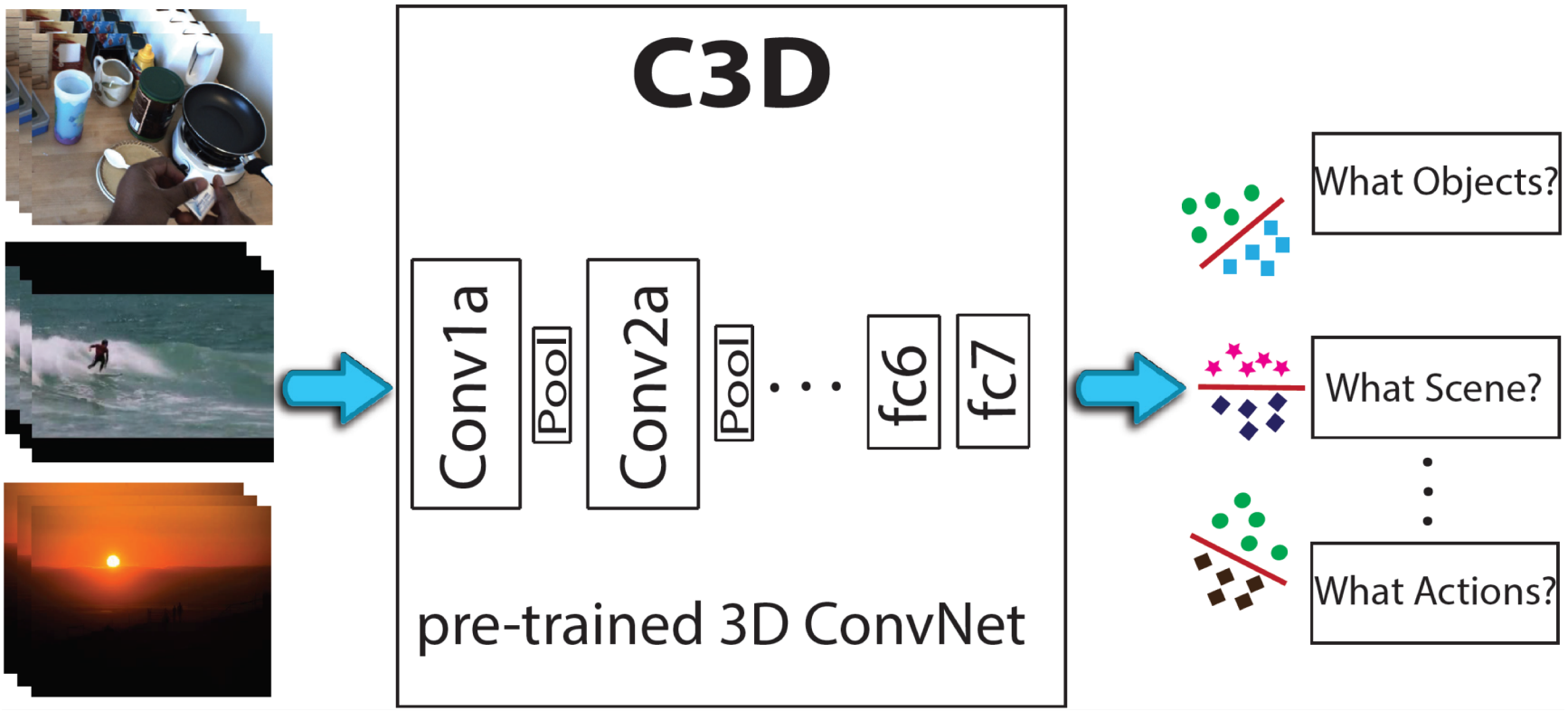


Learning Video Features with C3D



- C3D architecture
  - 8 convolution, 5 pool, 2 fully-connected layers
  - 3x3x3 convolution kernels
  - 2x2x2 pooling kernels
- Dataset: Sports-1M [Karpathy et al. CVPR14]
  - 1.1M videos of 487 different sport categories
  - Train/test splits are provided

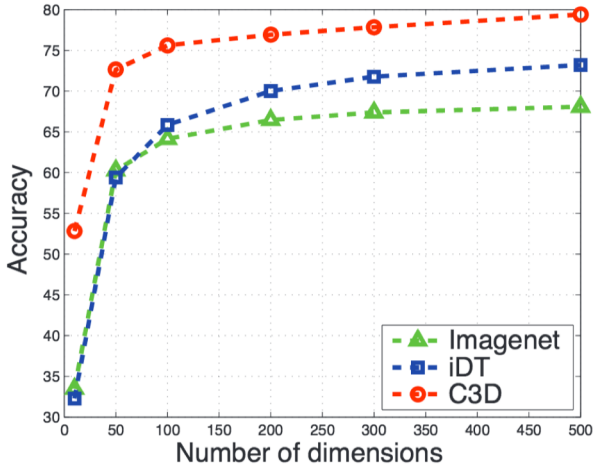
C3D as Generic Features

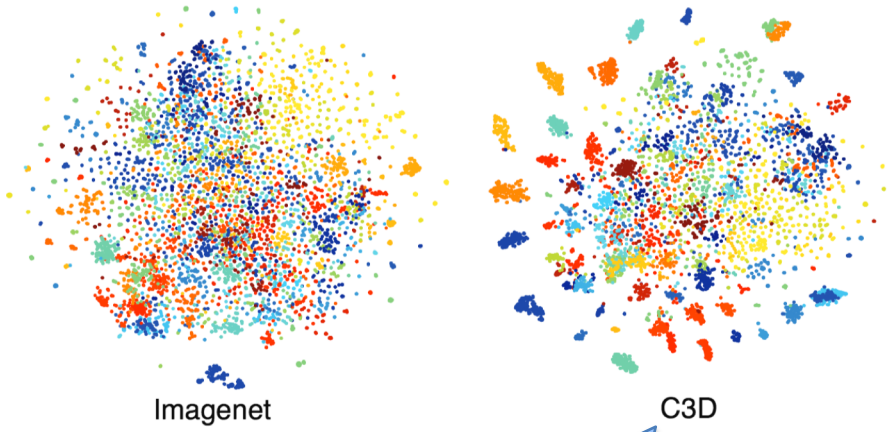


Simple recipe: C3D + linear SVM = good performance

Video Classification with C3D

Dataset Task	Sport1M action recognition	UCF101 action recognition	ASLAN action similarity labeling	YUPENN scene classification	UMD scene classification	Object object recognition
Method	[19]	[39]([26])	[31]	[10]	[10]	[32]
Result	80.2	75.8 (89.1)	68.7	96.2	77.7	12.0
C3D	**85.2**	**85.2 (90.4)**	**78.3**	**98.1**	**87.7**	**22.3**
Δ	5.0	9.4 (1.3)	9.6	1.9	10.0	10.3





C3D is discriminative and compact!

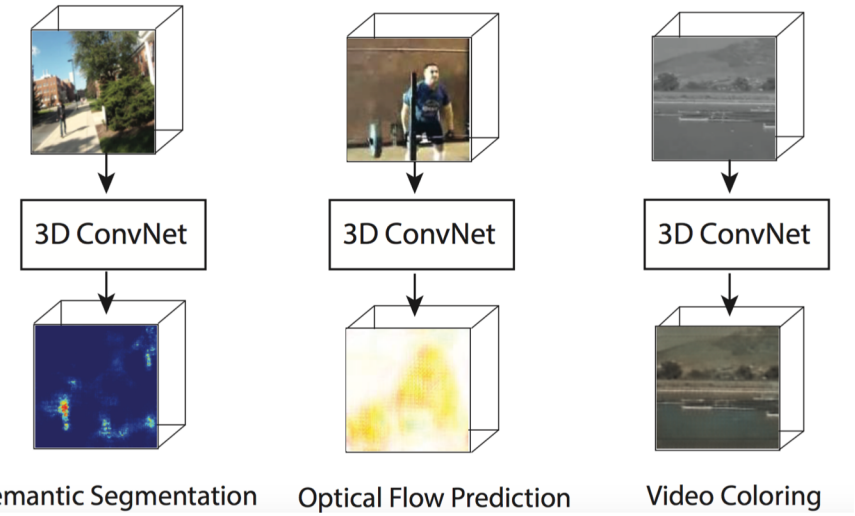
C3D code/model is publicly available

Action Recognition Task
UCF101

Video Voxel Prediction

D. Tran, L. Bourdev, R. Fergus, L. Torresani, M. Paluri, ***Deep End2End Voxel2Voxel Prediction***, **CVPRW16**

- Current methods make simple predictions
  - E.g. classification or detection
- Some others make highly-abstracted predictions
  - E.g. video captioning/description



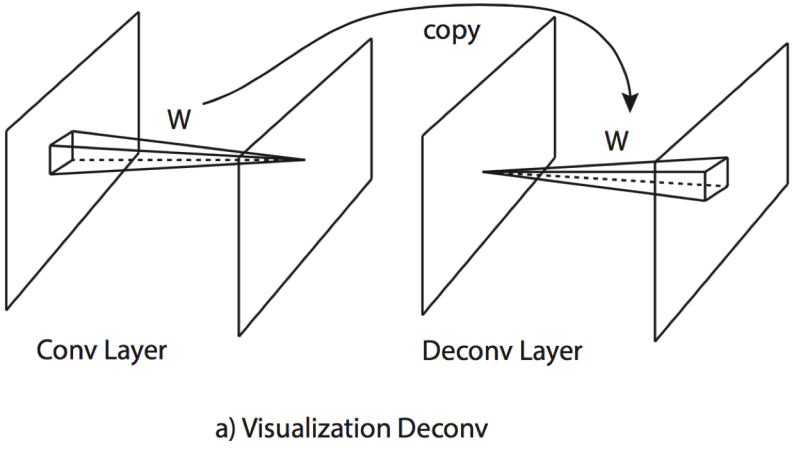
Voxel Prediction: making a prediction for every voxel

Many potential applications

Limitations of 3D ConvNets

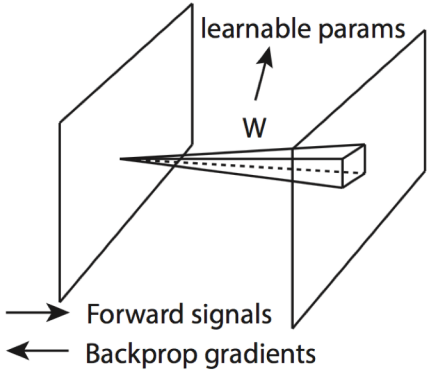
- Smaller output resolution at deeper layers
  - E.g. 2x7x7 at conv5b for C3D (input 16x112x112)
- Traditional upsamplers work poorly on sparse signals
- Fully-connected upsamplers involve many parameters

3D ConvNets with Deconv



a) Visualization Deconv

[Zeiler & Fergus ECCV'14]



b) Trainable Deconv

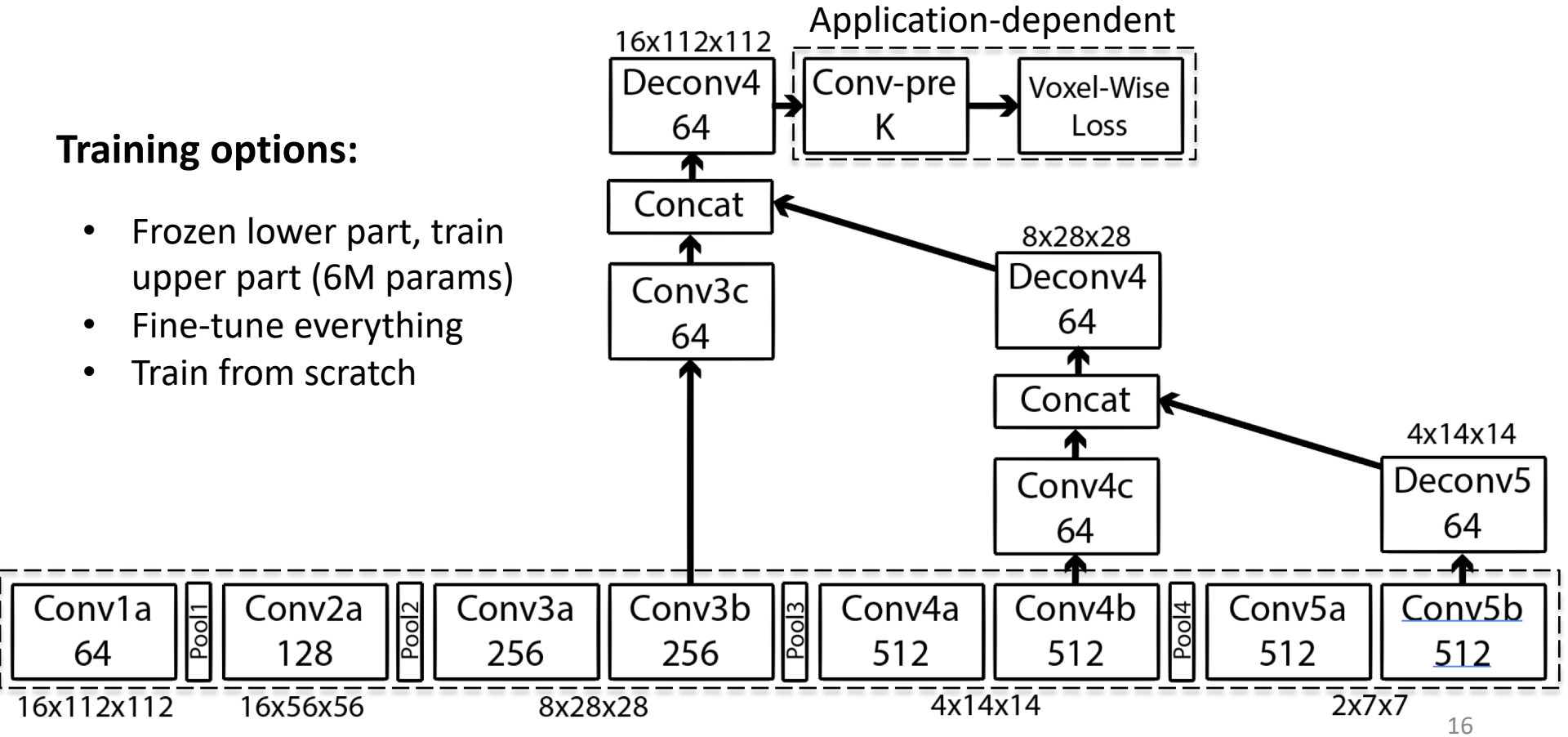
[Long et al. CVPR'15]

- Acts as convolutional upsamplers
- Has much smaller #params compared to fully-connected upsamplers
- Provides locally smooth predictions

V2V: End-to-end Voxel Prediction

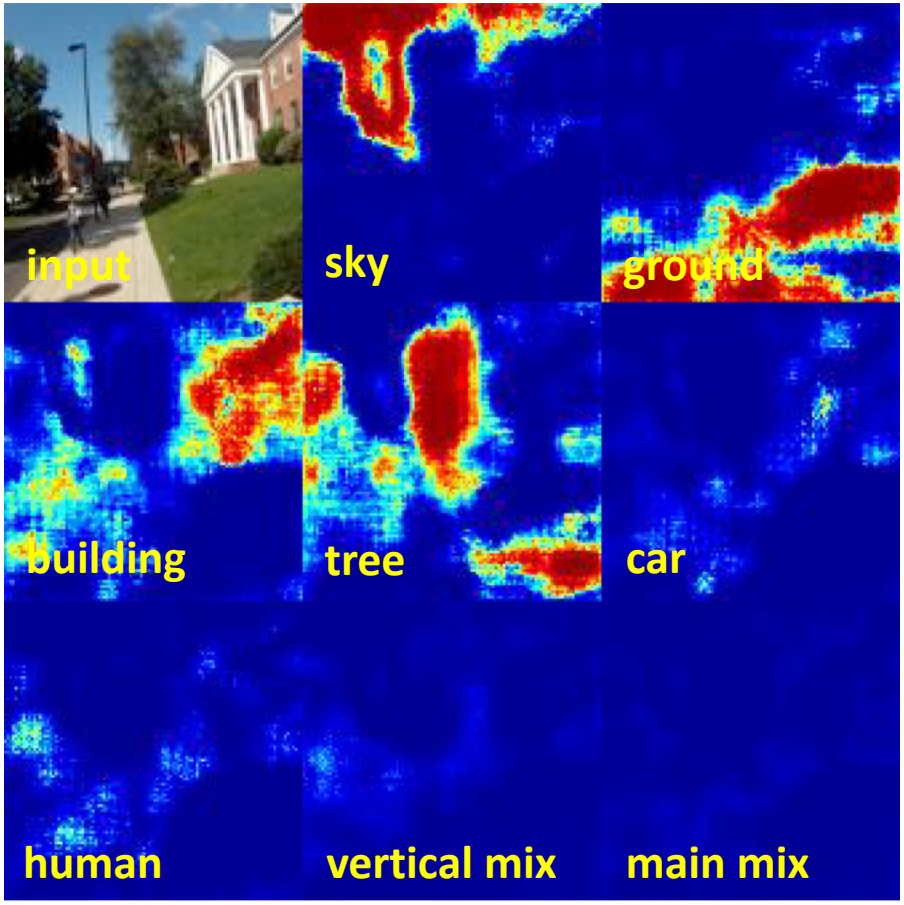
Training options:

- Frozen lower part, train upper part (6M params)
- Fine-tune everything
- Train from scratch



Semantic Segmentation Results

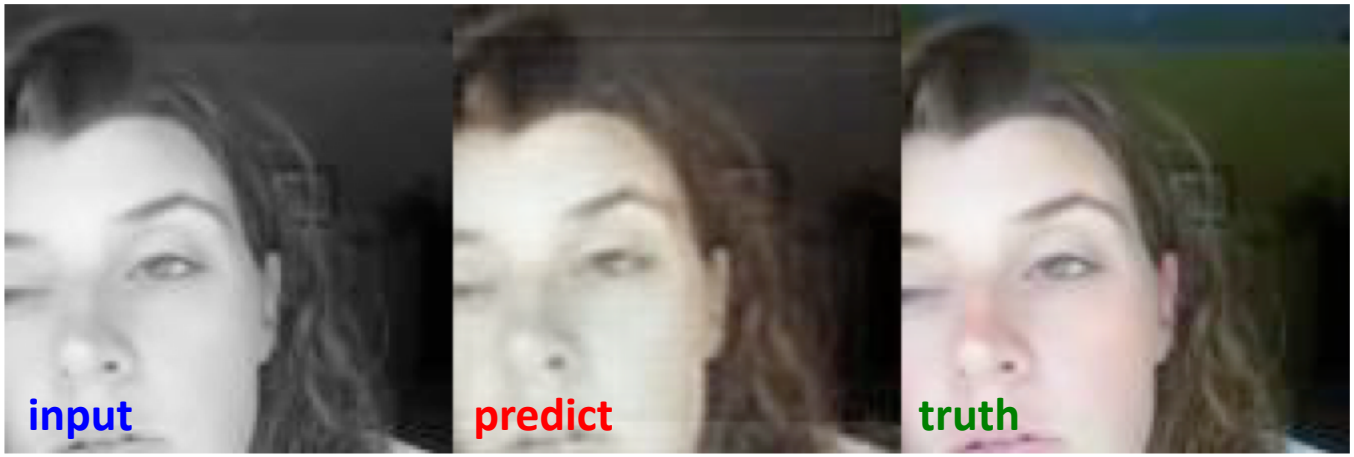
Method	Train	Acc (%)
2D-V2V	from scratch	55.7
Conv3b+Up	fine-tune	69.7
Conv4b+Up	fine-tune	72.7
Conv5b+Up	fine-tune	72.1
V2V-0	from scratch	66.7
V2V	fine-tune	**76.0**
V2V-remove-skip-0	from scratch	60.8
V2V-remove-skip	fine-tune	74.7
V2V-strided-convolution	from scratch	72.6



Results on GATECH

Video Coloring on UCF101

- Train & test on UCF101 train/test split 1
- An ill-posed and challenging problem
  - Cloth can take any colors



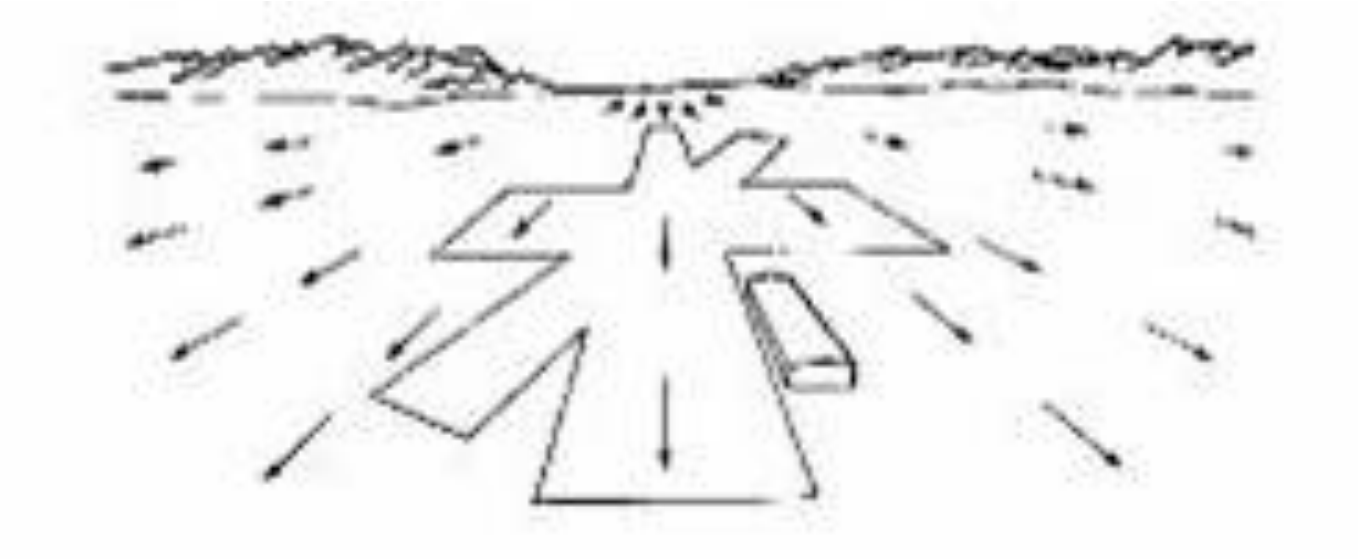
- Good predictions on "*common sense*" colors.

Overview

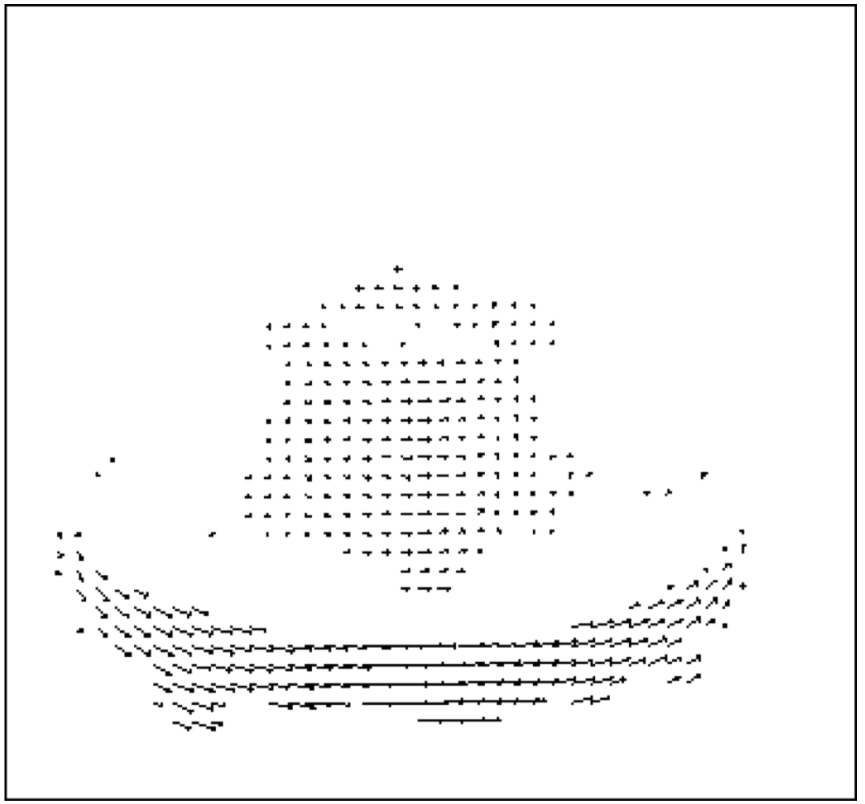
- Applying ConvNets to Video
  - Classification
  - Dense prediction
- Optical Flow
- Two-stream ConvNets for Video
  - Classification
  - Tracking

Optical flow

Combination of slides from Rick Szeliski, Steve Seitz, Alyosha Efros and Bill Freeman and Fredo Durand



Motion estimation: Optical flow



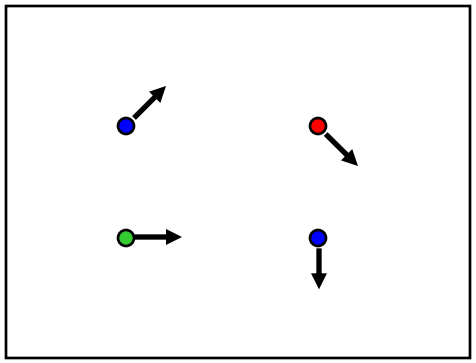
Will start by estimating motion of each pixel separately
Then will consider motion of entire image

Why estimate motion?

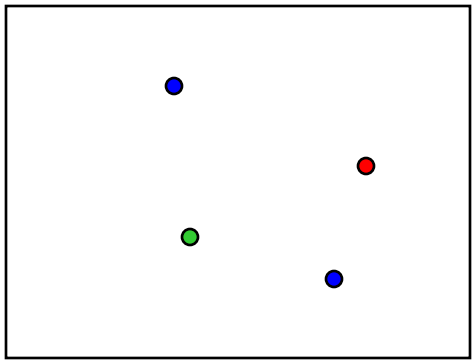
Lots of uses

- Feature representation for DeepNets [coming up]
- Track object behavior
- Correct for camera jitter (stabilization)
- Align images (mosaics)
- 3D shape reconstruction
- Special effects

Problem definition: optical flow



$H(x, y)$



$I(x, y)$

How to estimate pixel motion from image H to image I?

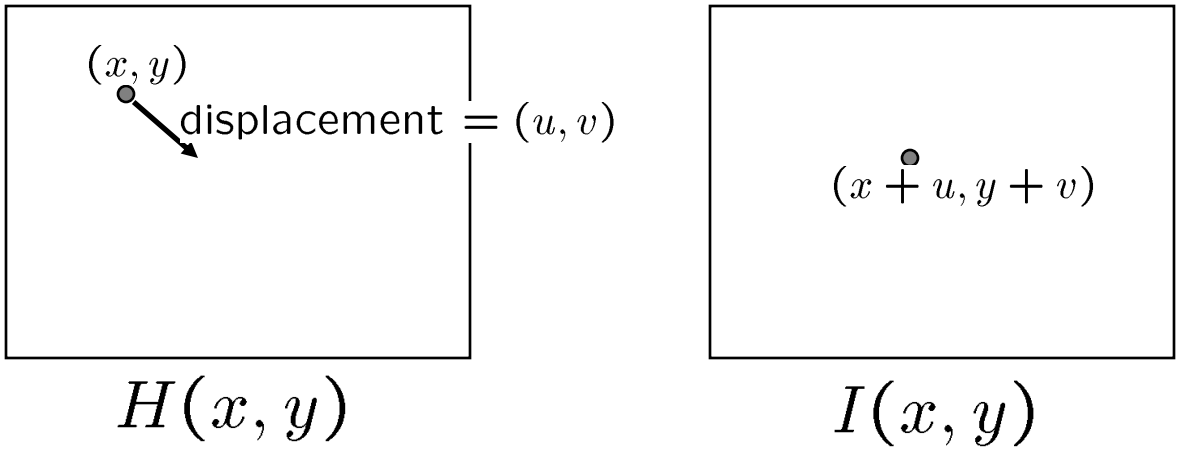
- Solve pixel correspondence problem
  - given a pixel in H, look for nearby pixels of the same color in I

Key assumptions

- color constancy: a point in H looks the same in I
  - For grayscale images, this is brightness constancy
- small motion: points do not move very far

This is called the optical flow problem

Optical flow constraints (grayscale images)



Let's look at these constraints more closely

- brightness constancy: Q: what's the equation?

  H(x,y)=I(x+u, y+v)

- small motion: (u and v are less than 1 pixel)
  - suppose we take the Taylor series expansion of I:

$$I(x+u, y+v) = I(x, y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \text{higher order terms}$$

$$\approx I(x, y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v$$

Optical flow equation

Combining these two equations

shorthand: $I_x = \frac{\partial I}{\partial x}$

$$
\begin{aligned}
0 &= I(x+u, y+v) - H(x,y) \\
&\approx I(x,y) + I_x u + I_y v - H(x,y) \\
&\approx (I(x,y) - H(x,y)) + I_x u + I_y v \\
&\approx I_t + I_x u + I_y v \\
&\approx I_t + \nabla I \cdot [u \;\; v]
\end{aligned}
$$

In the limit as u and v go to zero, this becomes exact

$$0 = I_t + \nabla I \cdot [\frac{\partial x}{\partial t} \;\; \frac{\partial y}{\partial t}]$$

Optical flow equation

$$0 = I_t + \nabla I \cdot [u \ v]$$

Q: how many unknowns and equations per pixel?

2 unknowns, one equation

Intuitively, what does this constraint mean?

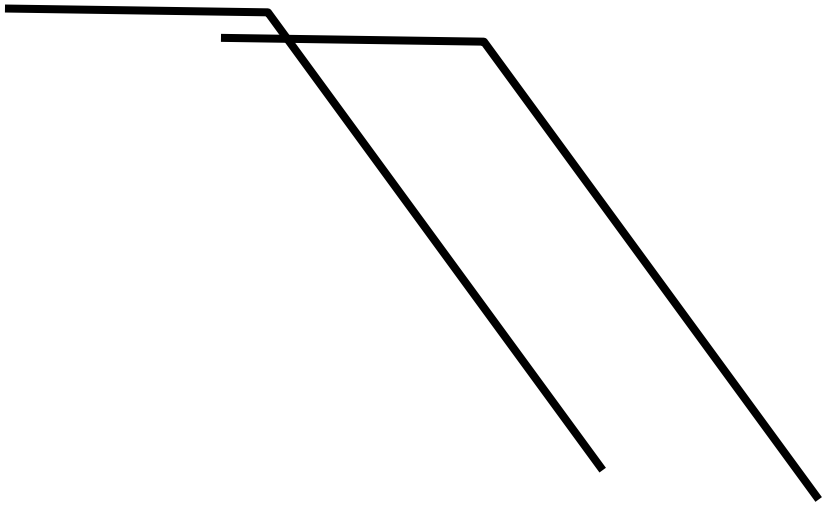
- The component of the flow in the gradient direction is determined
- The component of the flow parallel to an edge is unknown

This explains the Barber Pole illusion

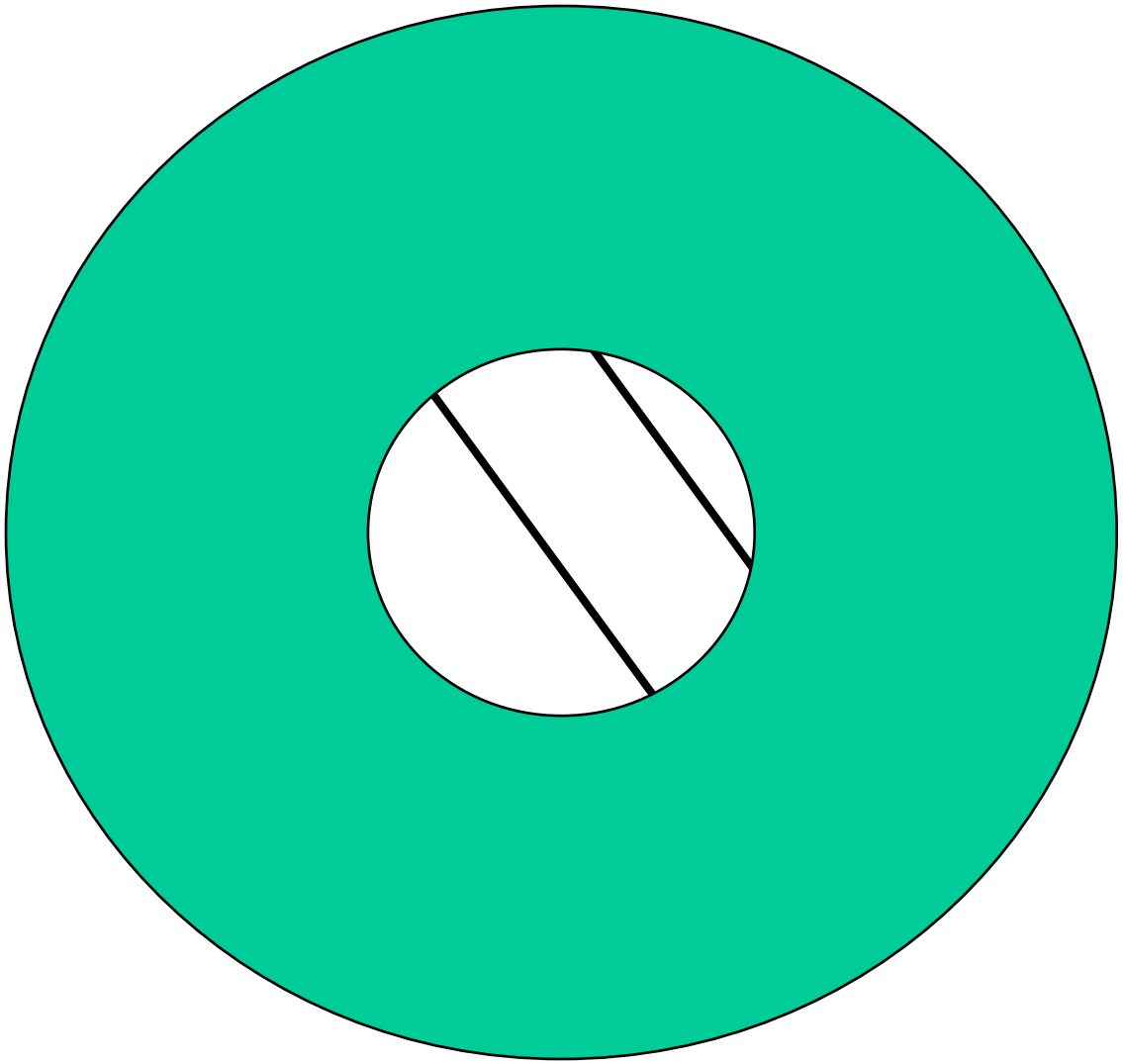
http://www.sandlotscience.com/Ambiguous/Barberpole_Illusion.htm
http://www.liv.ac.uk/~marcob/Trieste/barberpole.html

http://en.wikipedia.org/wiki/Barber's_pol

Aperture problem



Aperture problem



Solving the aperture problem

How to get more equations for a pixel?

- Basic idea: impose additional constraints
  - most common is to assume that the flow field is smooth locally
  - one method: pretend the pixel's neighbors have the same (u,v)
    - » If we use a 5x5 window, that gives us 25 equations per pixel!

$$0 = I_t(\mathbf{p_i}) + \nabla I(\mathbf{p_i}) \cdot [u\ v]$$

$$\begin{bmatrix} I_x(\mathbf{p_1}) & I_y(\mathbf{p_1}) \\ I_x(\mathbf{p_2}) & I_y(\mathbf{p_2}) \\ \vdots & \vdots \\ I_x(\mathbf{p_{25}}) & I_y(\mathbf{p_{25}}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{p_1}) \\ I_t(\mathbf{p_2}) \\ \vdots \\ I_t(\mathbf{p_{25}}) \end{bmatrix}$$

$$\underset{25\times2}{A} \qquad \underset{2\times1}{d} \qquad \underset{25\times1}{b}$$

RGB version

How to get more equations for a pixel?

- Basic idea: impose additional constraints
  - most common is to assume that the flow field is smooth locally
  - one method: pretend the pixel's neighbors have the same (u,v)
    - » If we use a 5x5 window, that gives us 25*3 equations per pixel!

$$0 = I_t(\mathbf{p_i})[0,1,2] + \nabla I(\mathbf{p_i})[0,1,2] \cdot [u\ v]$$

$$\begin{bmatrix} I_x(\mathbf{p_1})[0] & I_y(\mathbf{p_1})[0] \\ I_x(\mathbf{p_1})[1] & I_y(\mathbf{p_1})[1] \\ I_x(\mathbf{p_1})[2] & I_y(\mathbf{p_1})[2] \\ \vdots & \vdots \\ I_x(\mathbf{p_{25}})[0] & I_y(\mathbf{p_{25}})[0] \\ I_x(\mathbf{p_{25}})[1] & I_y(\mathbf{p_{25}})[1] \\ I_x(\mathbf{p_{25}})[2] & I_y(\mathbf{p_{25}})[2] \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{p_1})[0] \\ I_t(\mathbf{p_1})[1] \\ I_t(\mathbf{p_1})[2] \\ \vdots \\ I_t(\mathbf{p_{25}})[0] \\ I_t(\mathbf{p_{25}})[1] \\ I_t(\mathbf{p_{25}})[2] \end{bmatrix}$$

A (75x2) $\quad d$ (2x1) $\quad b$ (75x1)

Note that RGB is not enough to disambiguate because R, G & B are correlated
Just provides better gradient

Lukas-Kanade flow

Prob: we have more equations than unknowns

$$\underset{25\text{x}2}{A}\ \underset{2\text{x}1}{d} = \underset{25\text{x}1}{b} \longrightarrow \text{minimize } \|Ad - b\|^2$$

Solution: solve least squares problem

- minimum least squares solution given by solution (in d) of:

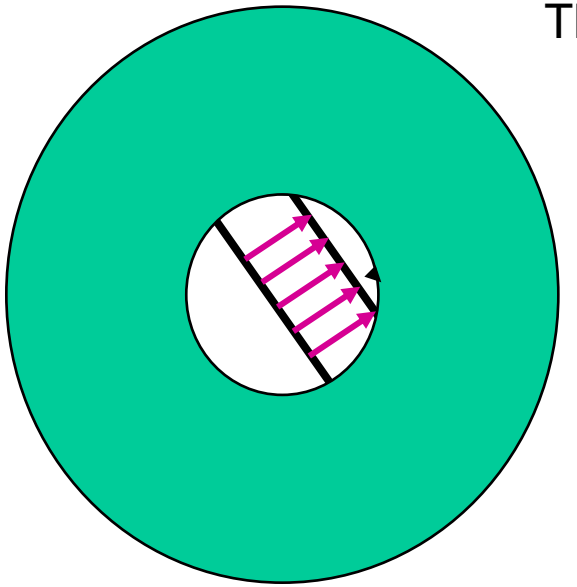
$$\underset{2\text{x}2}{(A^T A)}\ \underset{2\text{x}1}{d} = \underset{2\text{x}1}{A^T b}$$

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

$A^T A$ $\qquad$ $A^T b$

- The summations are over all pixels in the K x K window
- This technique was first proposed by Lukas & Kanade (1981)

Aperture Problem and Normal Flow

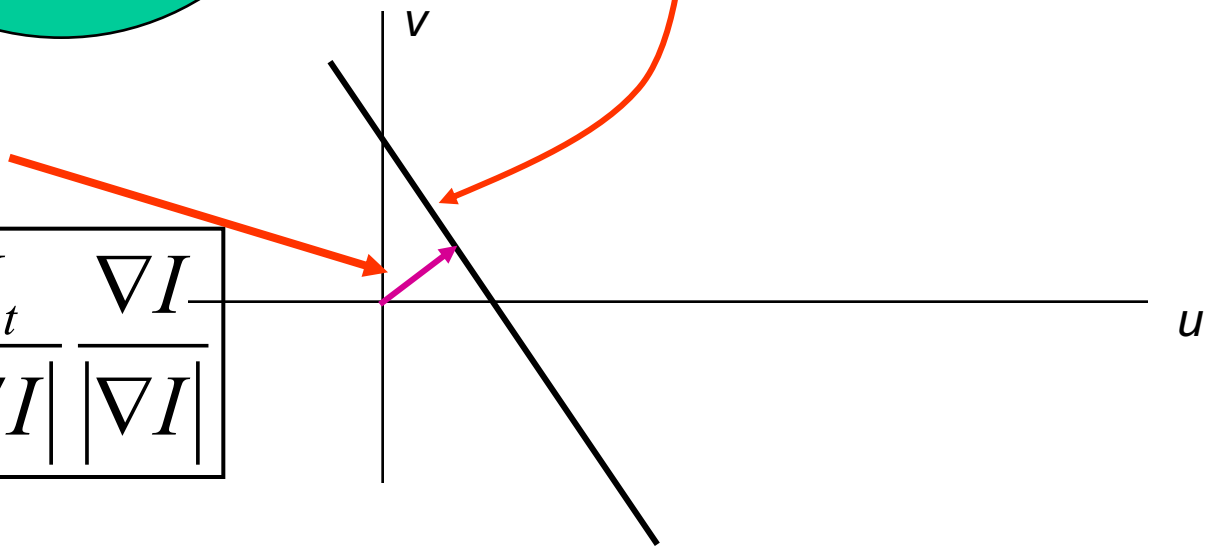


The gradient constraint:

$$I_x u + I_y v + I_t = 0$$

$$\nabla I \bullet \vec{U} = 0$$

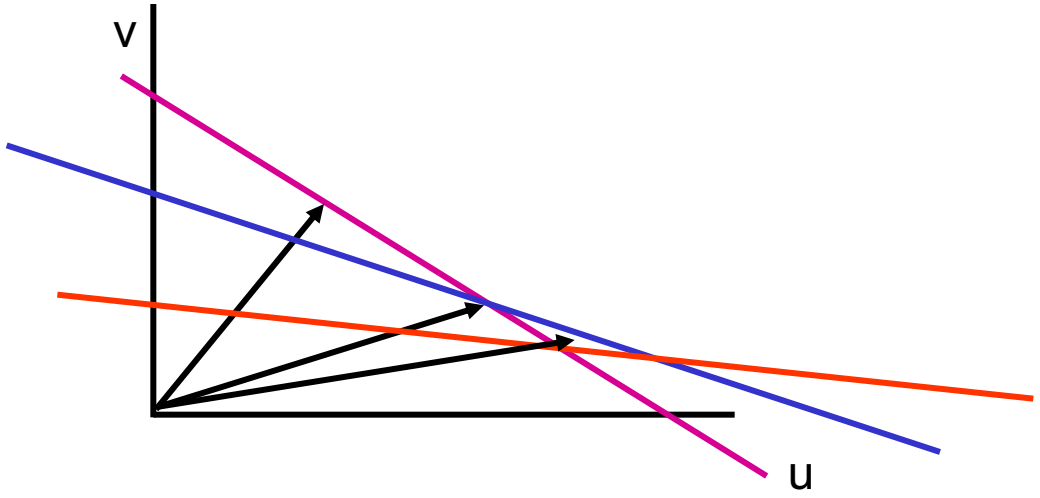
Defines a line in the *(u,v)* space



Normal Flow:

$$u_\perp = -\frac{I_t}{|\nabla I|}\frac{\nabla I}{|\nabla I|}$$

Combining Local Constraints



$$\nabla I^1 \bullet U = -I_t^1$$

$$\nabla I^2 \bullet U = -I_t^2$$

$$\nabla I^3 \bullet U = -I_t^3$$

etc.

Conditions for solvability

- Optimal (u, v) satisfies Lucas-Kanade equation

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

$$A^T A \qquad\qquad A^T b$$

When is This Solvable?

- A^TA should be invertible
- A^TA should not be too small due to noise
  - eigenvalues λ_1 and λ_2 of A^TA should not be too small
- A^TA should be well-conditioned
  - λ_1 / λ_2 should not be too large (λ_1 = larger eigenvalue)

A^TA is solvable when there is no aperture problem

$$A^T A = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} = \sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} [I_x \; I_y] = \sum \nabla I (\nabla I)^T$$

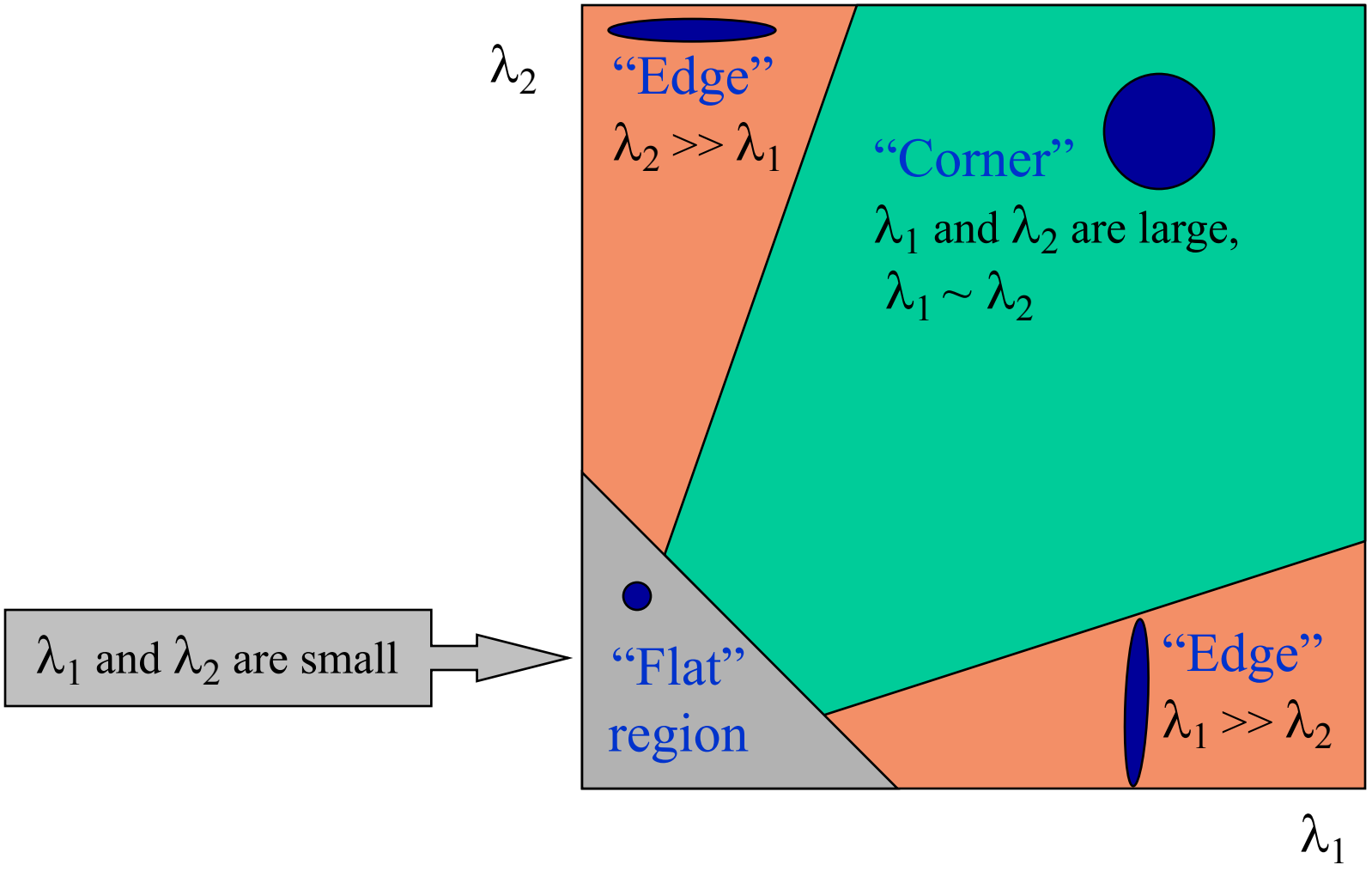
Eigenvectors of A^TA

$$A^T A = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} = \sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} [I_x \; I_y] = \sum \nabla I (\nabla I)^T$$

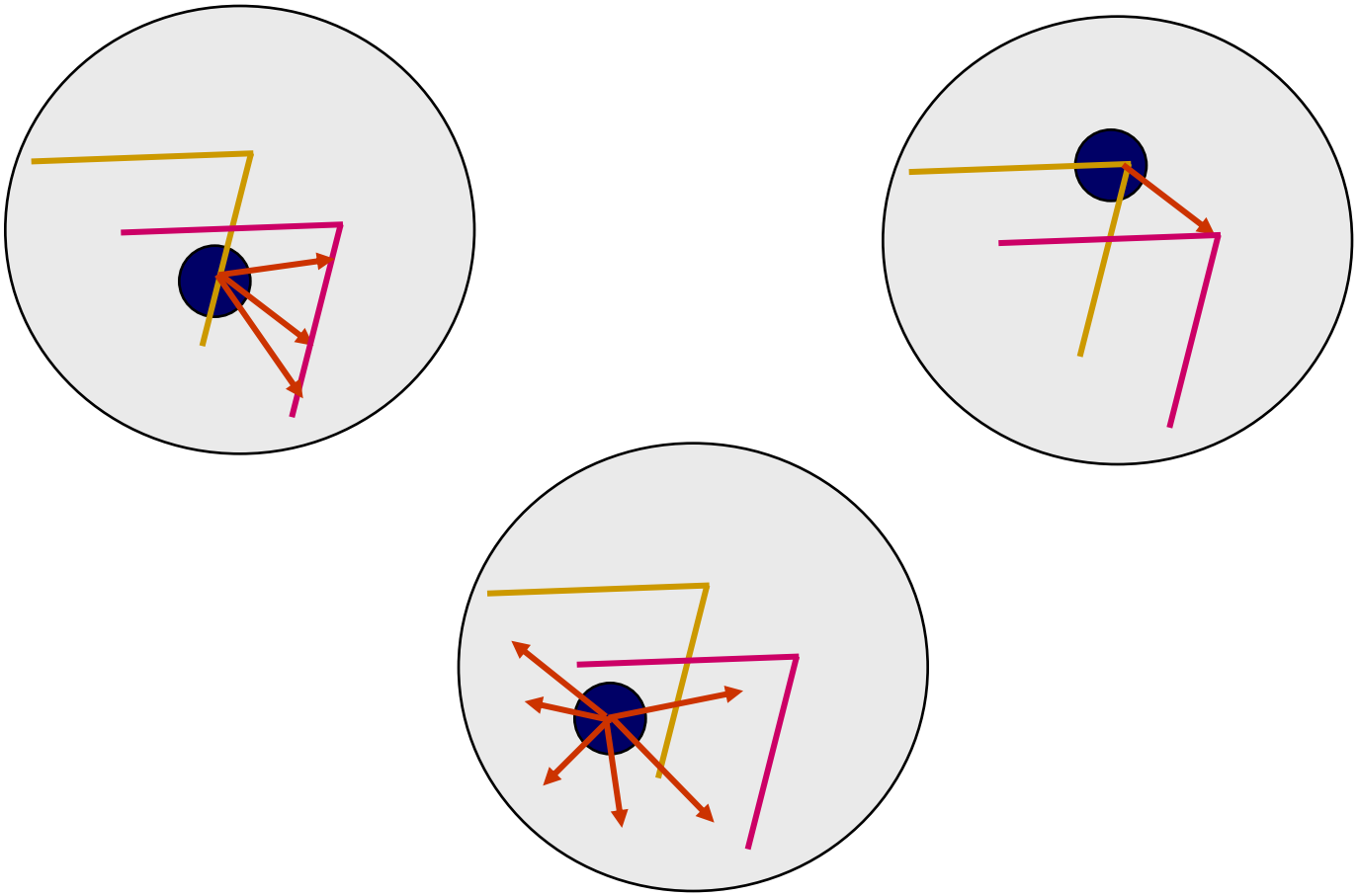
- Recall the Harris corner detector: $M = A^TA$ is the *second moment matrix*
- The eigenvectors and eigenvalues of M relate to edge direction and magnitude
  - The eigenvector associated with the larger eigenvalue points in the direction of fastest intensity change
  - The other eigenvector is orthogonal to it

Interpreting the eigenvalues

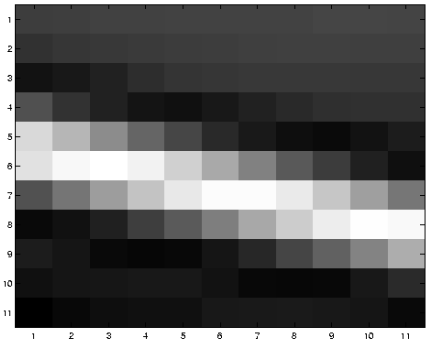
Classification of image points using eigenvalues of the second moment matrix:

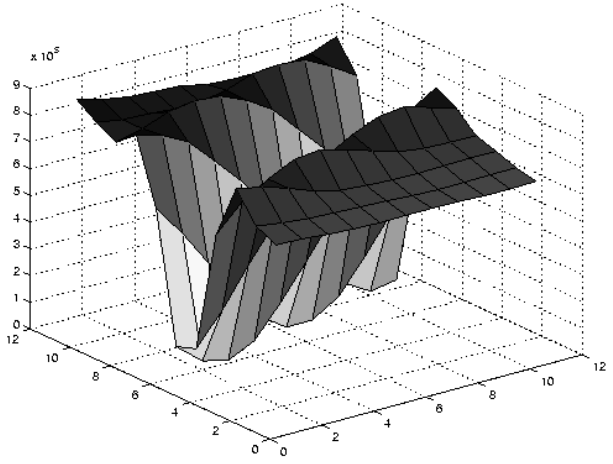


Local Patch Analysis



Edge

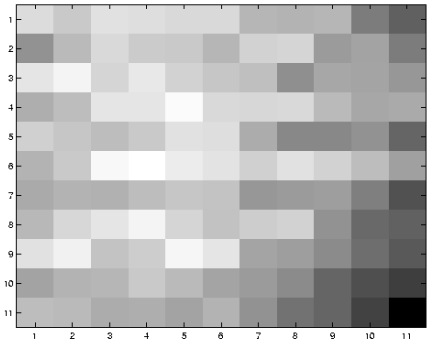


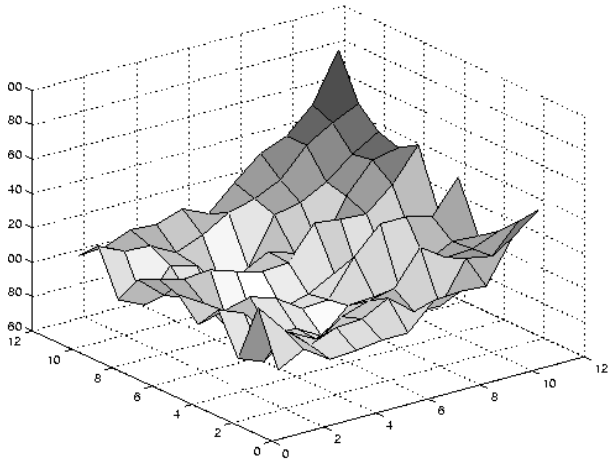


$$\sum \nabla I (\nabla I)^T$$

- large gradients, all the same
- large λ_1, small λ_2

Low texture region



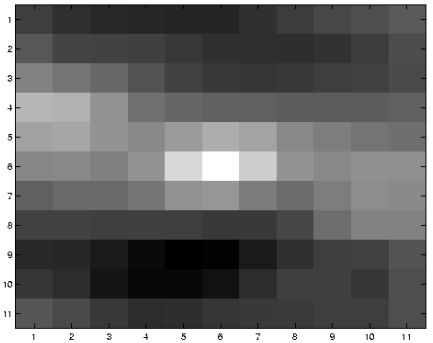


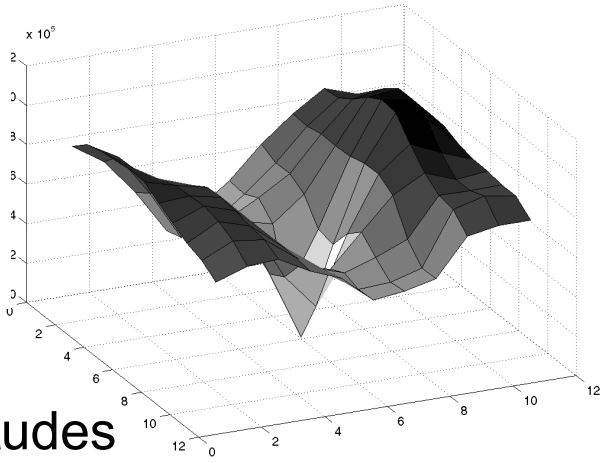
$$\sum \nabla I (\nabla I)^T$$

– gradients have small magnitude

– small λ_1, small λ_2

High textured region





$$\sum \nabla I (\nabla I)^T$$

– gradients are different, large magnitudes

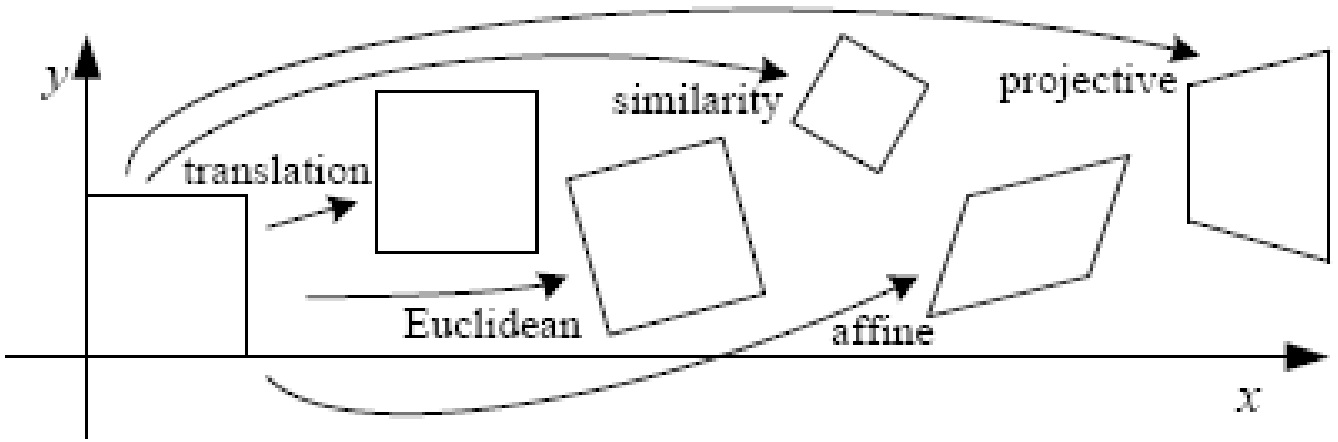
– large λ_1, large λ_2

Observation

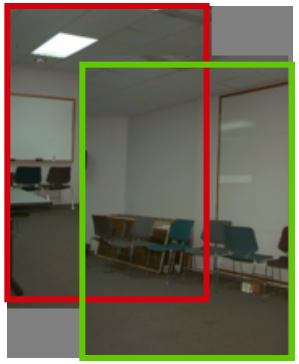
This is a two image problem BUT

- Can measure sensitivity by just looking at one of the images!
- This tells us which pixels are easy to track, which are hard
  - very useful later on when we do feature tracking...

Motion models

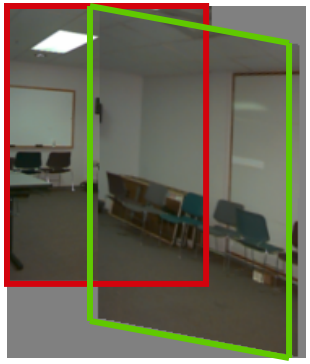


Translation



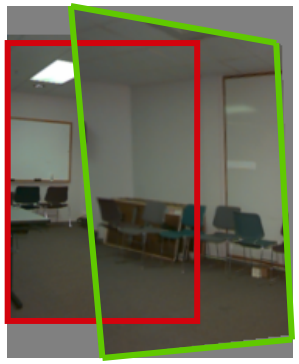
2 unknowns

Affine



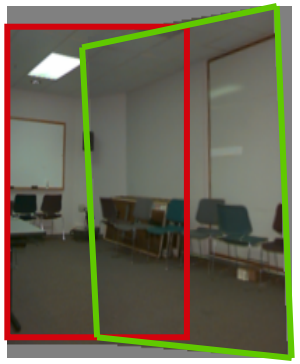
6 unknowns

Perspective



8 unknowns

3D rotation

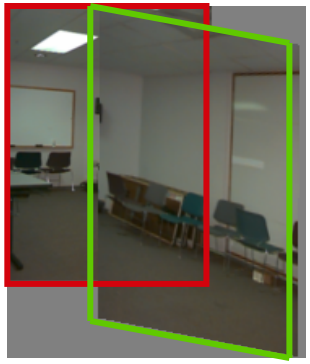


3 unknowns

Affine motion

$$u(x,y) = a_1 + a_2 x + a_3 y$$

$$v(x,y) = a_4 + a_5 x + a_6 y$$



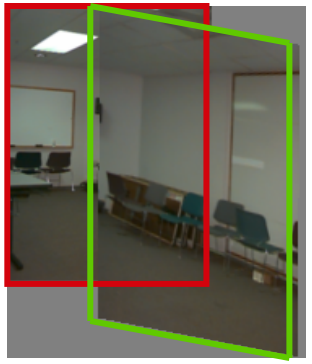
- Substituting into the brightness constancy equation:

$$I_x \cdot u + I_y \cdot v + I_t \approx 0$$

Affine motion

$$u(x,y) = a_1 + a_2 x + a_3 y$$

$$v(x,y) = a_4 + a_5 x + a_6 y$$



- Substituting into the brightness constancy equation:

$$I_x(a_1 + a_2 x + a_3 y) + I_y(a_4 + a_5 x + a_6 y) + I_t \approx 0$$

- Each pixel provides 1 linear constraint in 6 unknowns

- Least squares minimization:

$$Err(\vec{a}) = \sum \left[I_x(a_1 + a_2 x + a_3 y) + I_y(a_4 + a_5 x + a_6 y) + I_t \right]^2$$

Errors in Lukas-Kanade

What are the potential causes of errors in this procedure?

- Suppose A^TA is easily invertible
- Suppose there is not much noise in the image

When our assumptions are violated

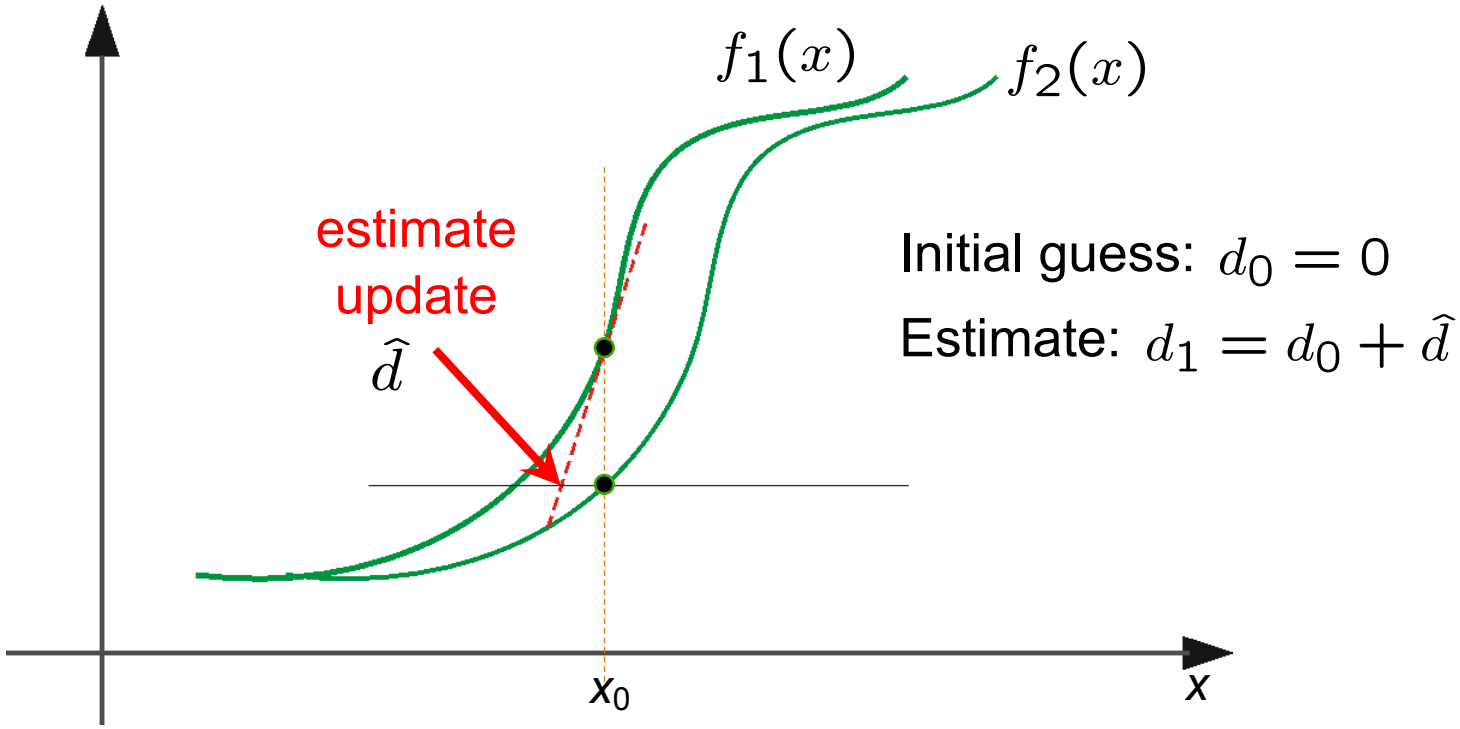
- Brightness constancy is not satisfied
- The motion is not small
- A point does not move like its neighbors
  - window size is too large
  - what is the ideal window size?

Iterative Refinement

Iterative Lukas-Kanade Algorithm

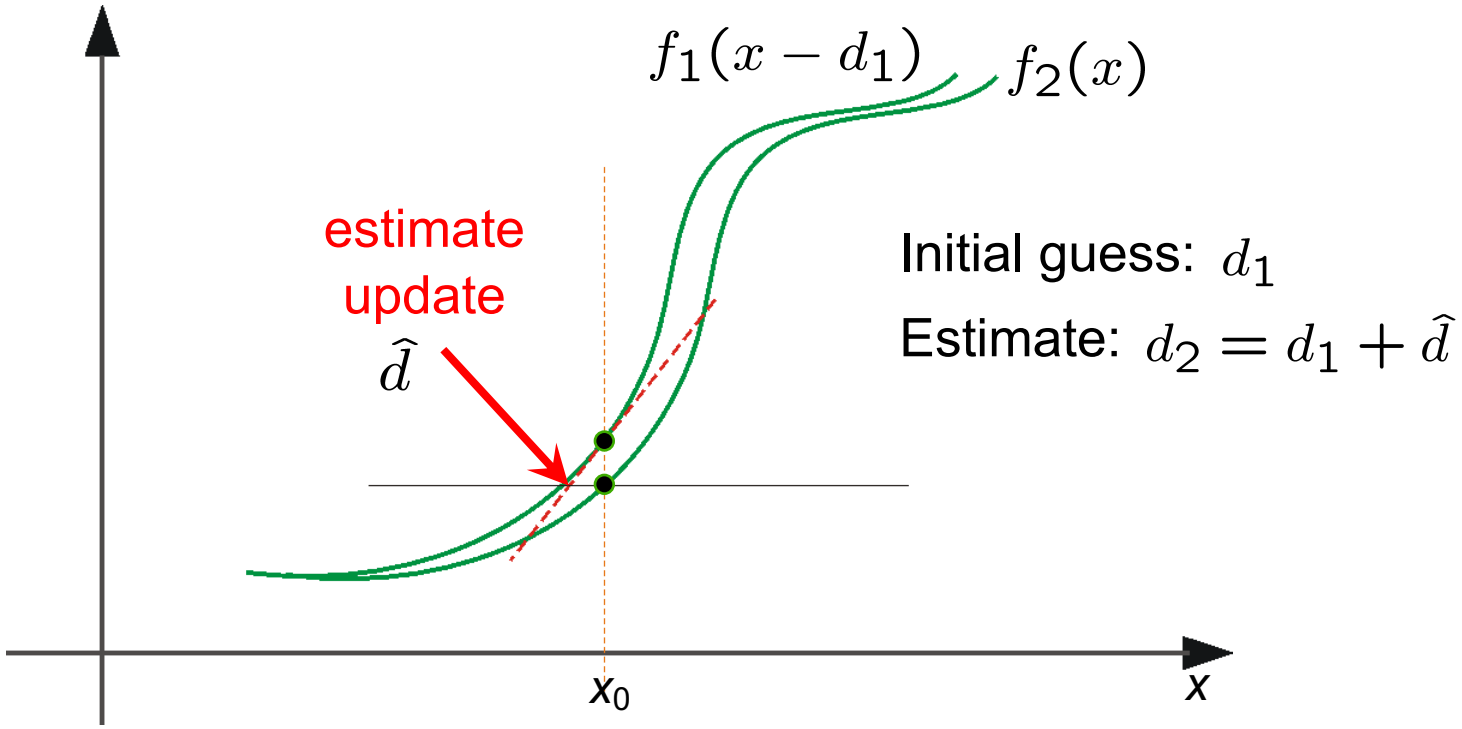
1. Estimate velocity at each pixel by solving Lucas-Kanade equations
2. Warp H towards I using the estimated flow field
   - *use image warping techniques*
3. Repeat until convergence

Optical Flow: Iterative Estimation

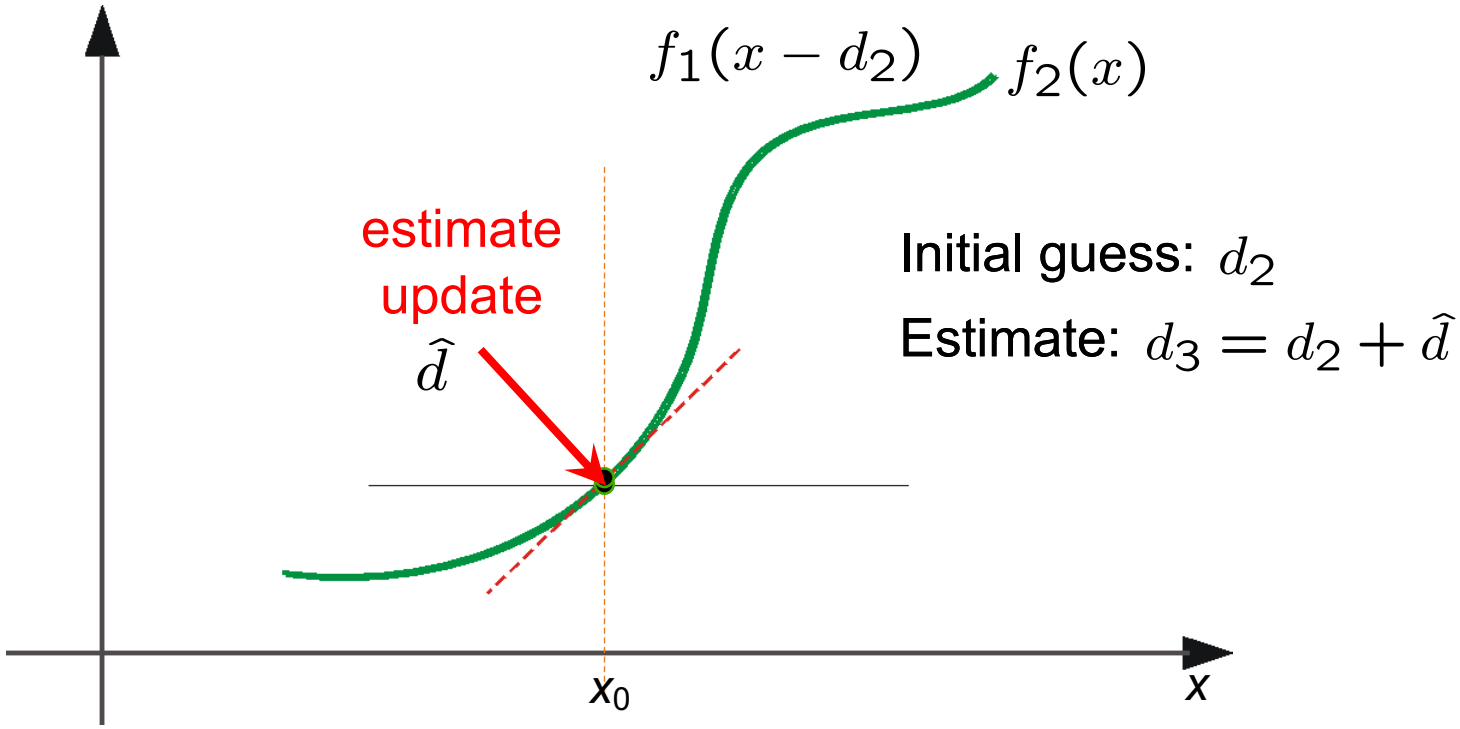


(using *d* for *displacement* here instead of *u*)

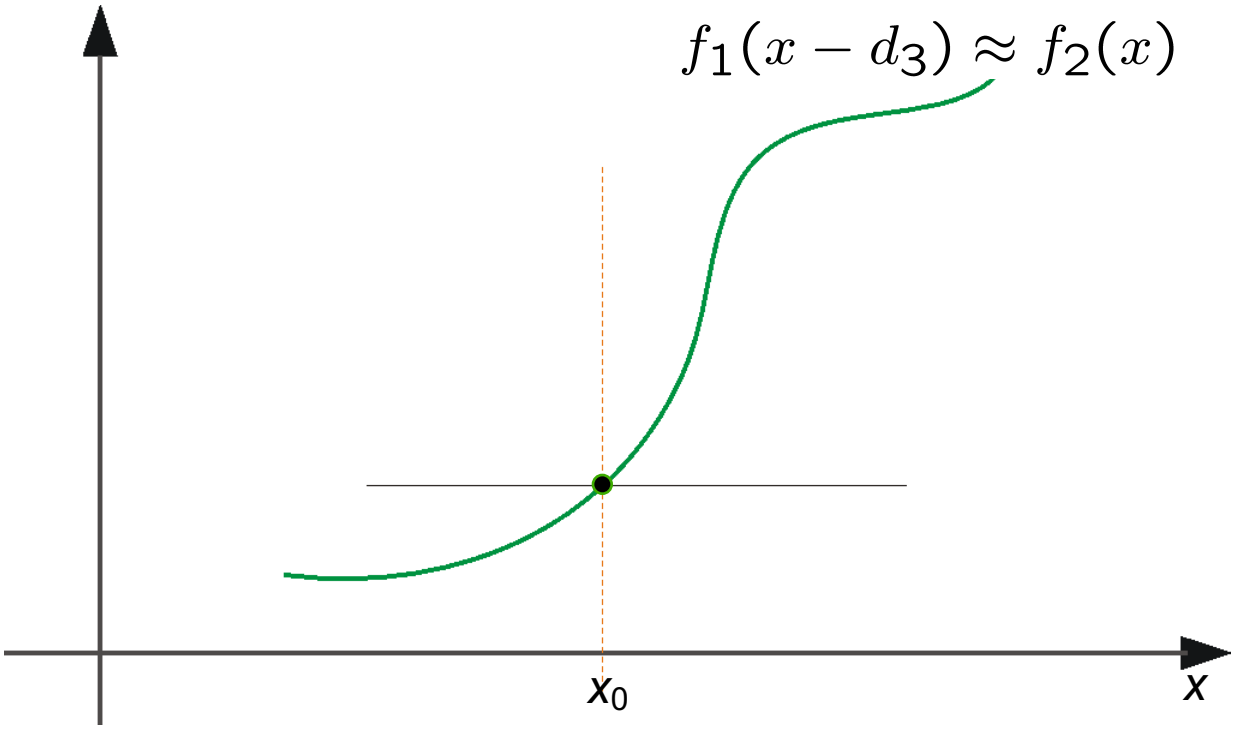
Optical Flow: Iterative Estimation



Optical Flow: Iterative Estimation



Optical Flow: Iterative Estimation



Optical Flow: Iterative Estimation

Some Implementation Issues:

- Warping is not easy (ensure that errors in warping are smaller than the estimate refinement)
- Warp one image, take derivatives of the other so you don't need to re-compute the gradient after each iteration.
- Often useful to low-pass filter the images before motion estimation (for better derivative estimation, and linear approximations to image intensity)

Revisiting the small motion assumption

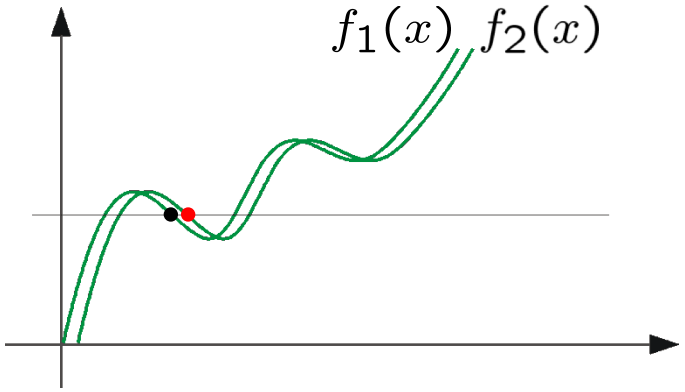
Is this motion small enough?

- Probably not—it's much larger than one pixel (2nd order terms dominate)
- How might we solve this problem?

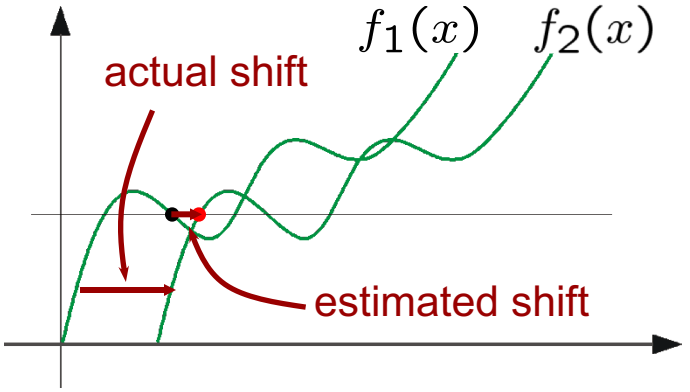
Optical Flow: Aliasing

Temporal aliasing causes ambiguities in optical flow because images can have many pixels with the same intensity.

I.e., how do we know which ‘correspondence’ is correct?



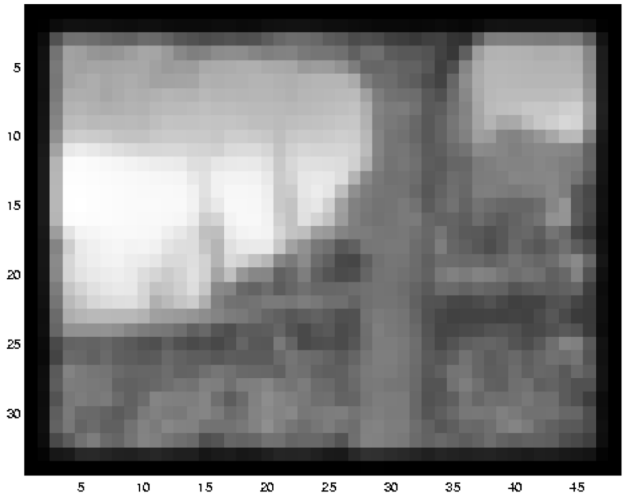
nearest match is correct (no aliasing)

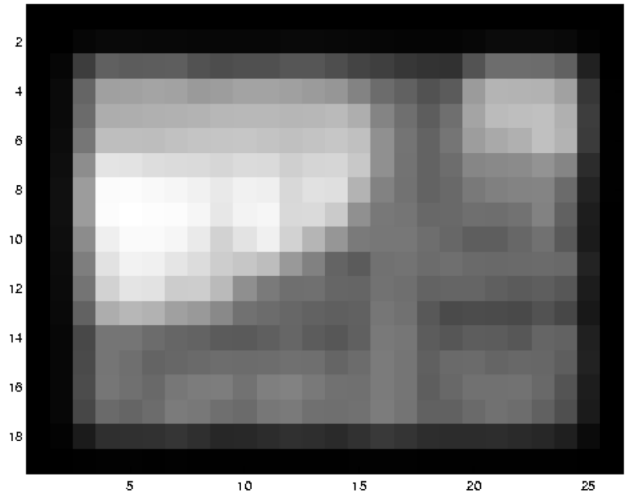


nearest match is incorrect (aliasing)

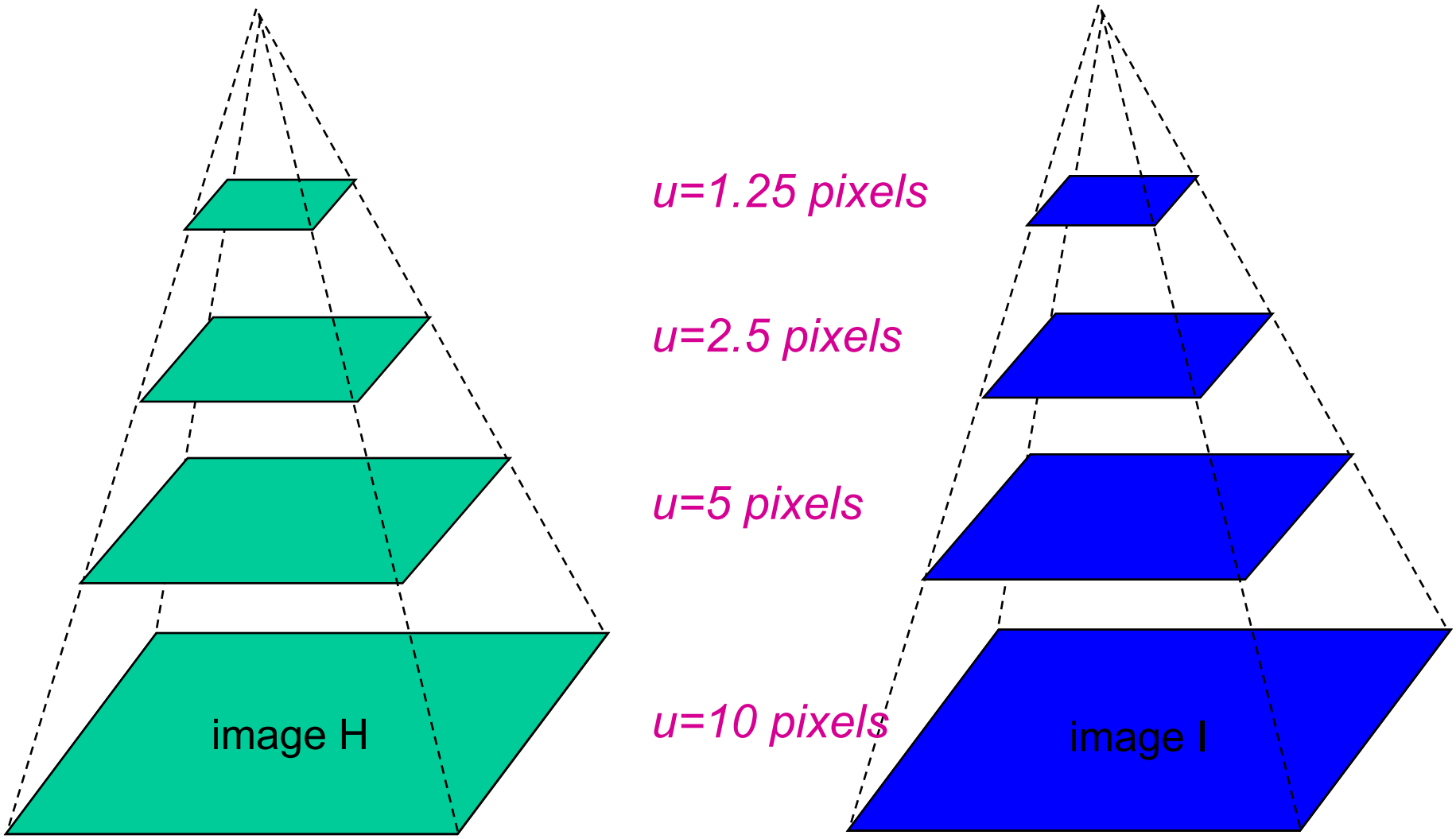
To overcome aliasing: coarse-to-fine estimation.

Reduce the resolution!





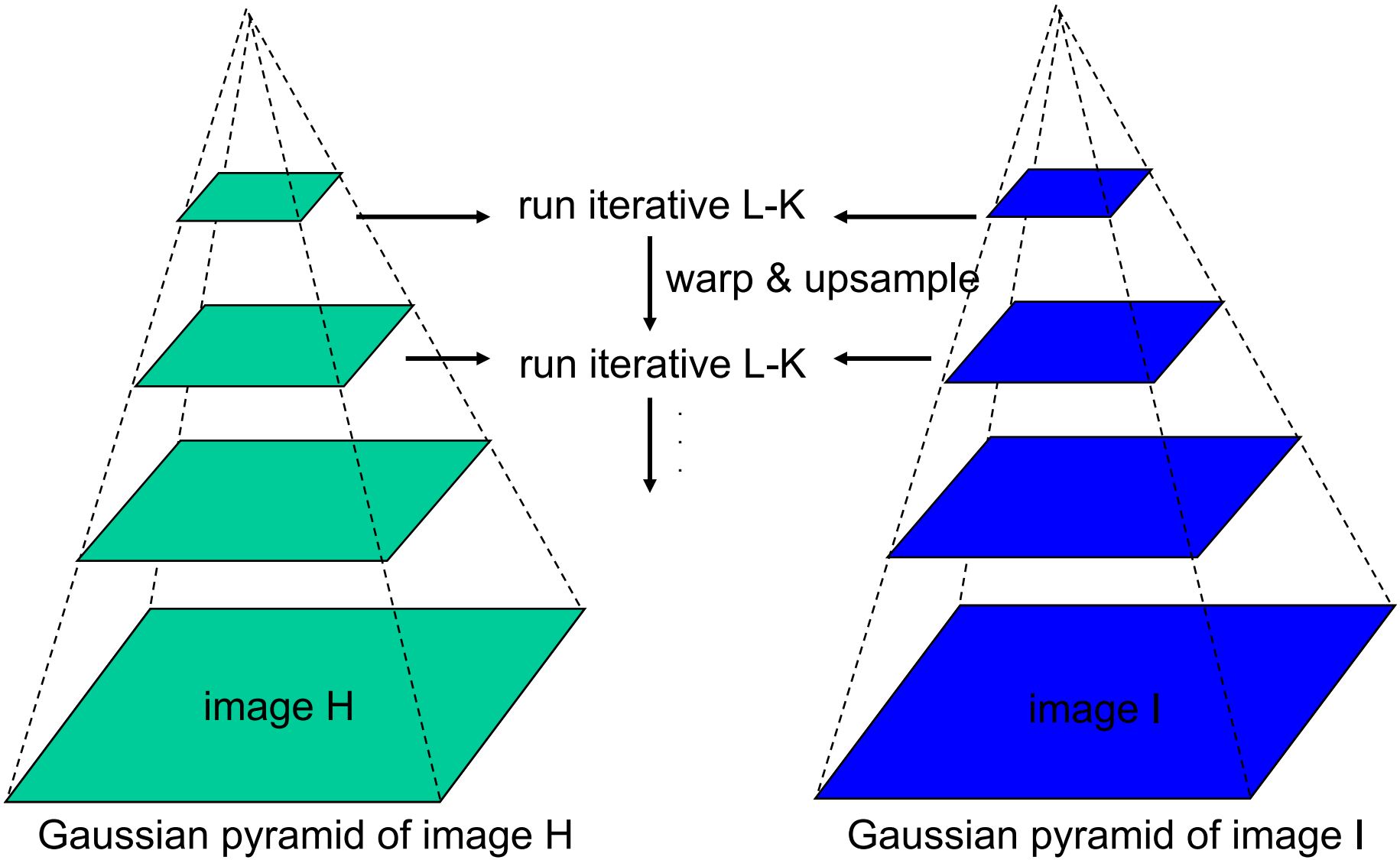
Coarse-to-fine optical flow estimation



Gaussian pyramid of image H

Gaussian pyramid of image I

Coarse-to-fine optical flow estimation



Recap: Classes of Techniques

Direct-methods (e.g. optical flow)

- Directly recover image motion from spatio-temporal image brightness variations
- Global motion parameters directly recovered without an intermediate feature motion calculation
- Dense motion fields, but more sensitive to appearance variations
- Suitable for video and when image motion is small (< 10 pixels)

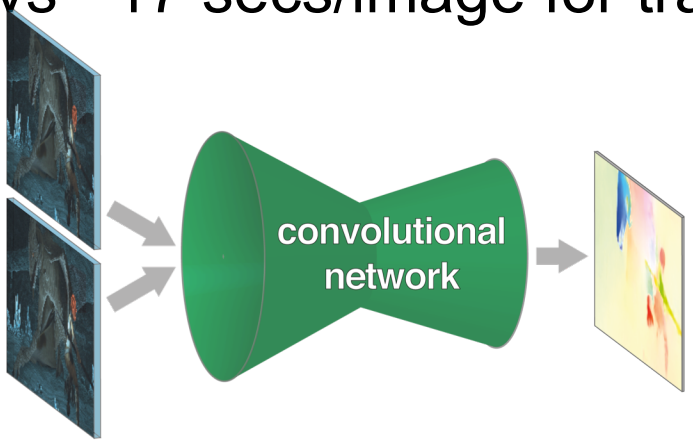
Feature-based methods (e.g. SIFT+Ransac+regression) [To be covered]

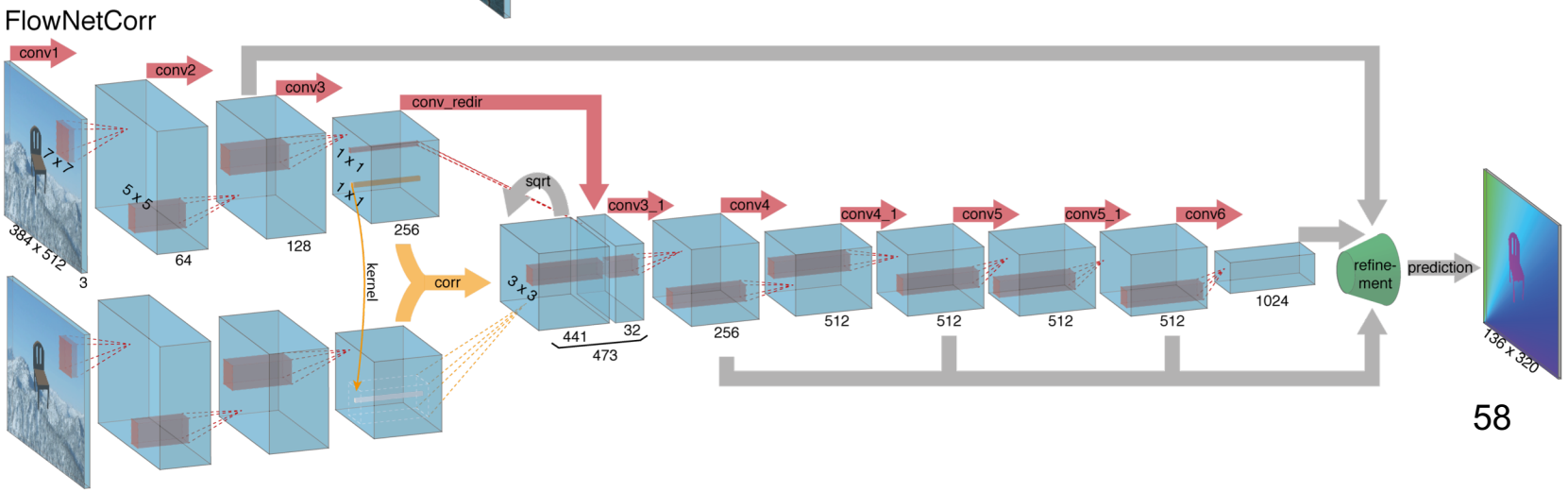
- Extract visual features (corners, textured areas) and track them over multiple frames
- Sparse motion fields, but possibly robust tracking
- Suitable especially when image motion is large (10-s of pixels)

FlowNet

FlowNet: Learning Optical Flow with Convolutional Networks [Fischer et al. 2015]

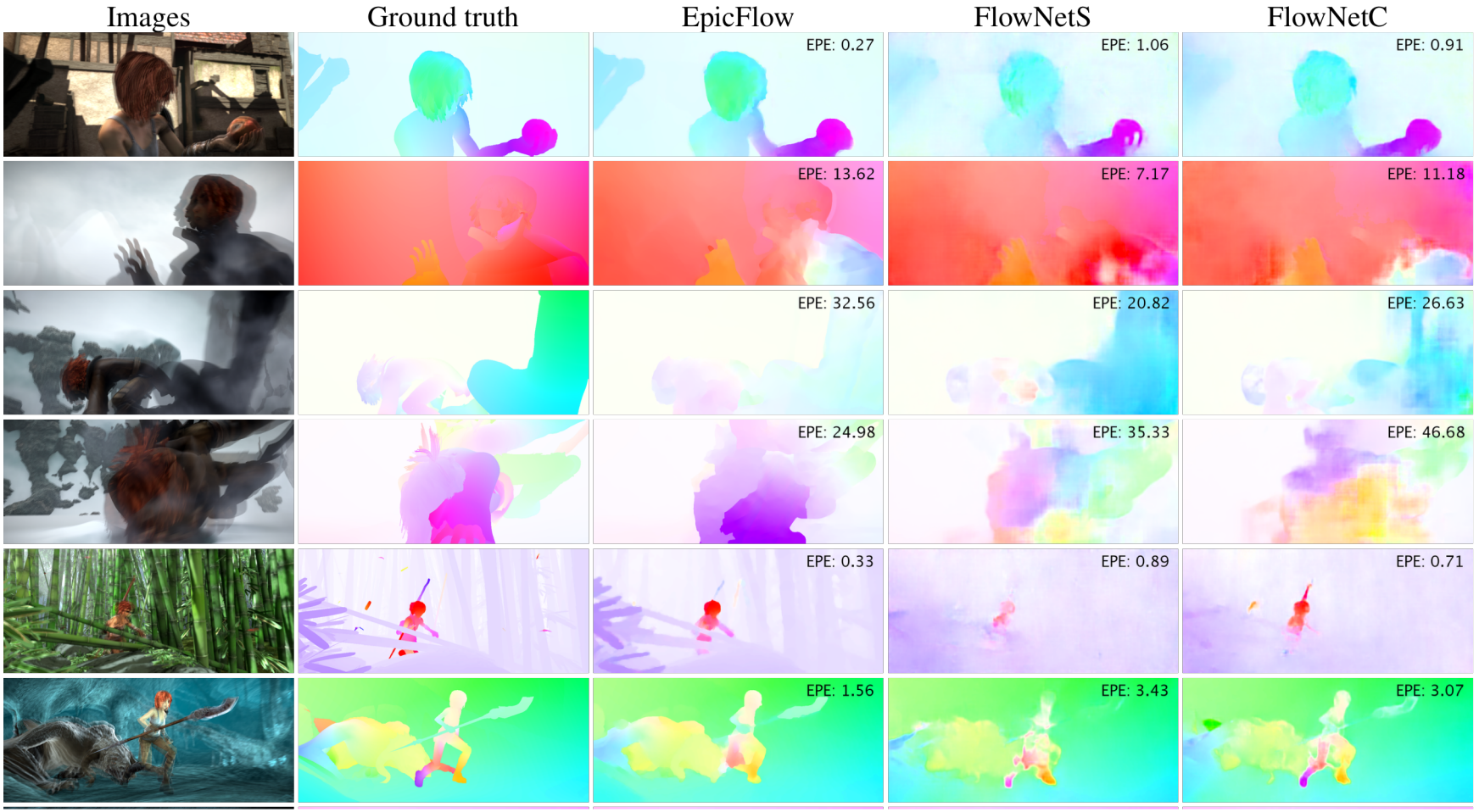
~ 1sec/image vs ~17 secs/image for traditional optical flow





FlowNet

- FlowNet: Learning Optical Flow with Convolutional Networks [Fischer et al. 2015]



Overview

Applying ConvNets to Video

- Classification
- Dense prediction

Optical Flow

Two-stream ConvNets for Video

- Classification
- Tracking

Deep Learning for Video Recognition

Christoph Feichtenhofer

work with

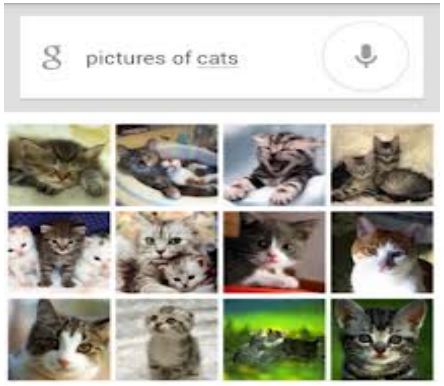
Axel Pinz
Graz University of Technology

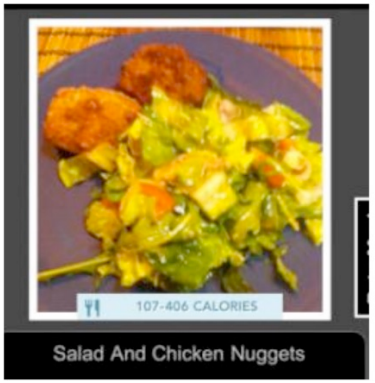
Richard P. Wildes
York University, Toronto

Andrew Zisserman
University of Oxford

Outline

- Progress in image-based recognition has been dramatic





- Part 1: Advancing Video Recognition with Deep Learning Architectures for Action, Scene and Object Recognition in Video
  - Published in CVPR16, NIPS16, 2xCVPR17 and ICCV17

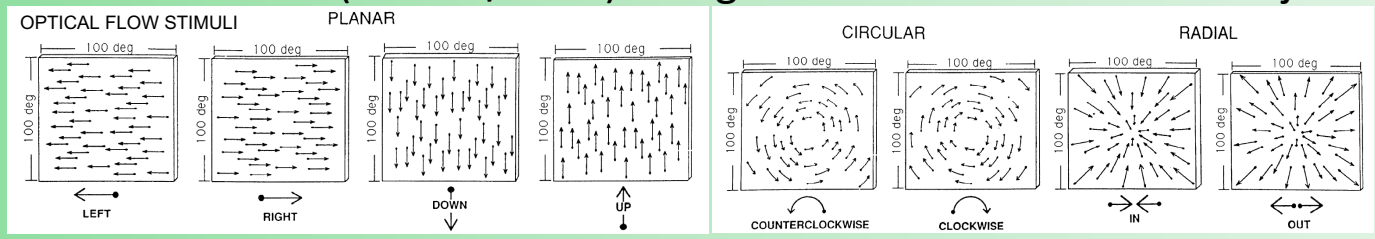
- Part 2: Understanding Deep Video Representations
  - What do deep architectures build internally?
  - Why are they performing so well and when do they fail?
  - In submission to CVPR'18 (2x)

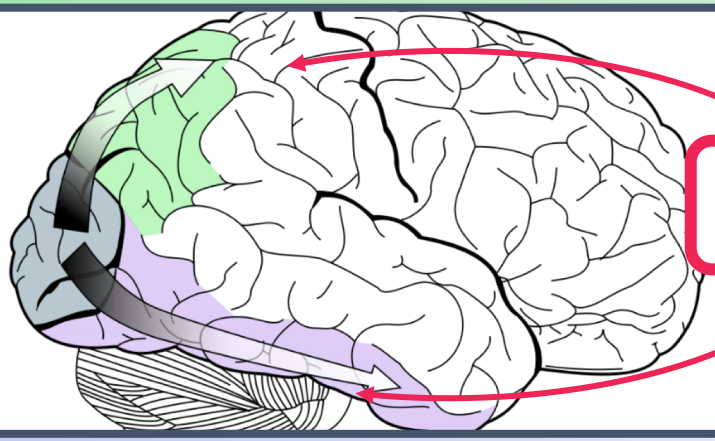
➔ **Amazing what the brain can do without appearance information**

Sources: Johansson, G. “Visual perception of biological motion and a model for its analysis.” Perception & Psychophysics. 14(2):201-211. 1973.

Motivation: Separate visual pathways in nature

➔ Dorsal stream (‘where/how’) recognizes motion and locates objects





➔ **“Interconnection” e.g. in STS area**

➔ Ventral (‘what’) stream performs object recognition

Sources: “*Sensitivity of MST neurons to optic flow stimuli. I. A continuum of response selectivity to large-field stimuli." Journal of neurophysiology 65.6 (1991).*

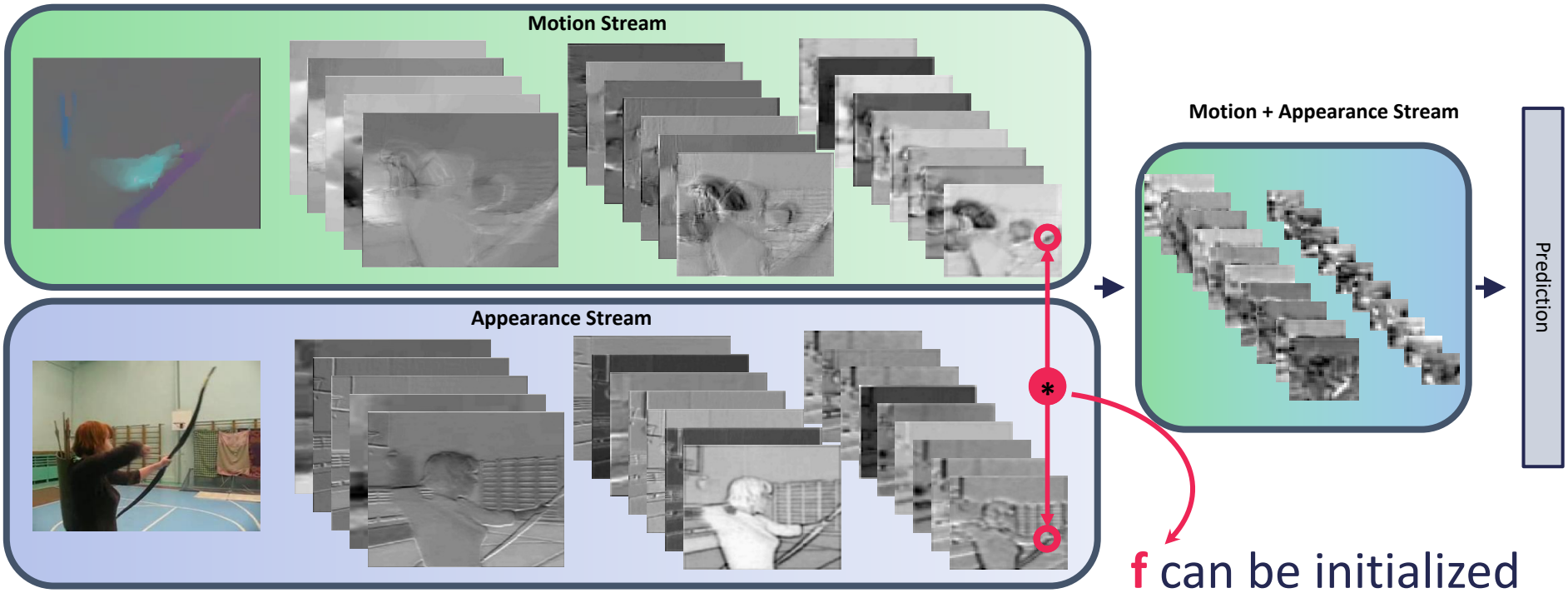
“*A cortical representation of the local visual environment*”, Nature. 392 (6676): 598–601, 2009

https://en.wikipedia.org/wiki/Two-streams_hypothesis

Convolutional Two-Stream Network Fusion

- We study a number of ways of fusing two-stream ConvNets

[Simonyan & Zisserman, NIPS'14]

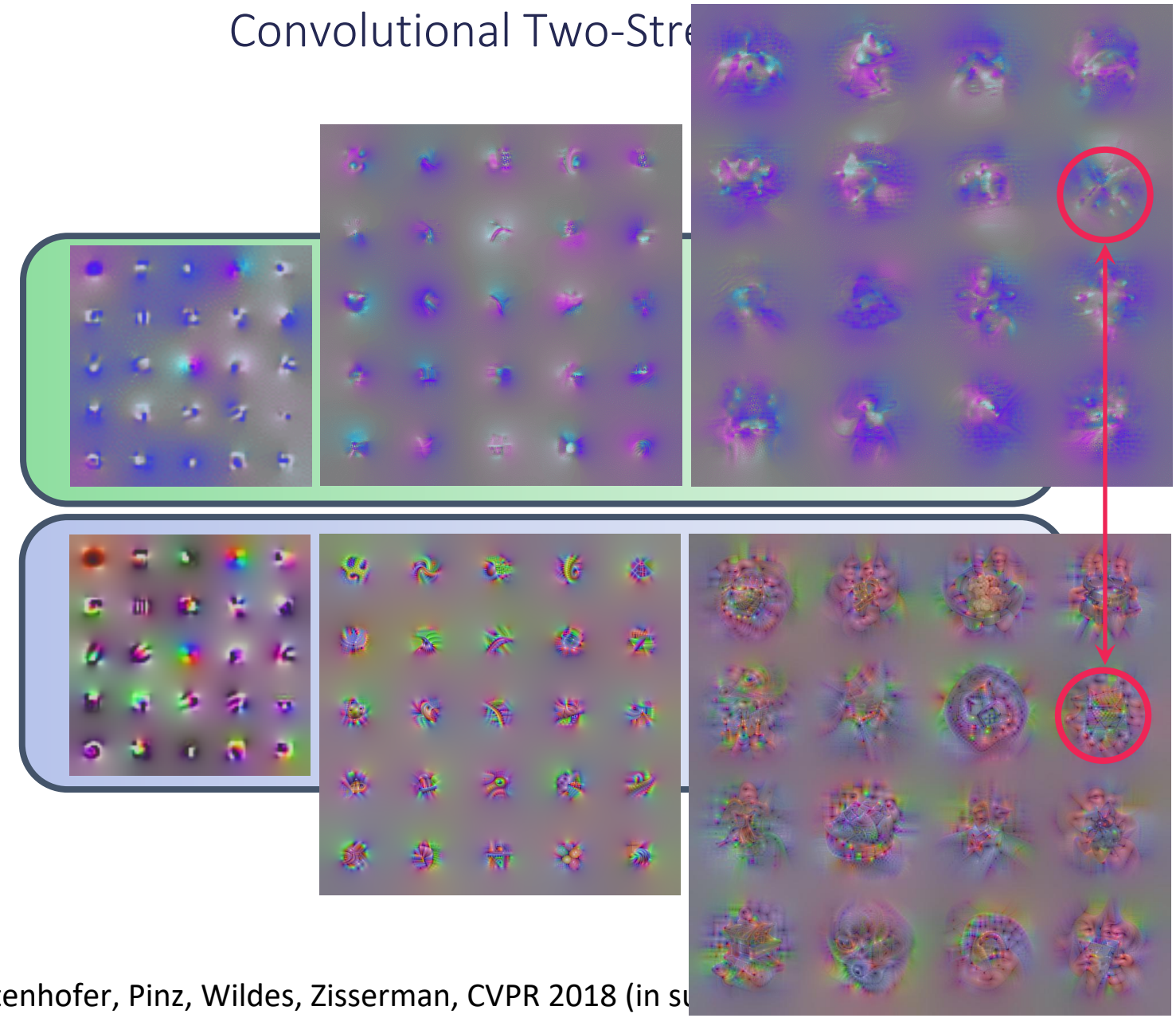


- Sum fusion works surprisingly well

f can be initialized as a sum kernel + feature identity mapping

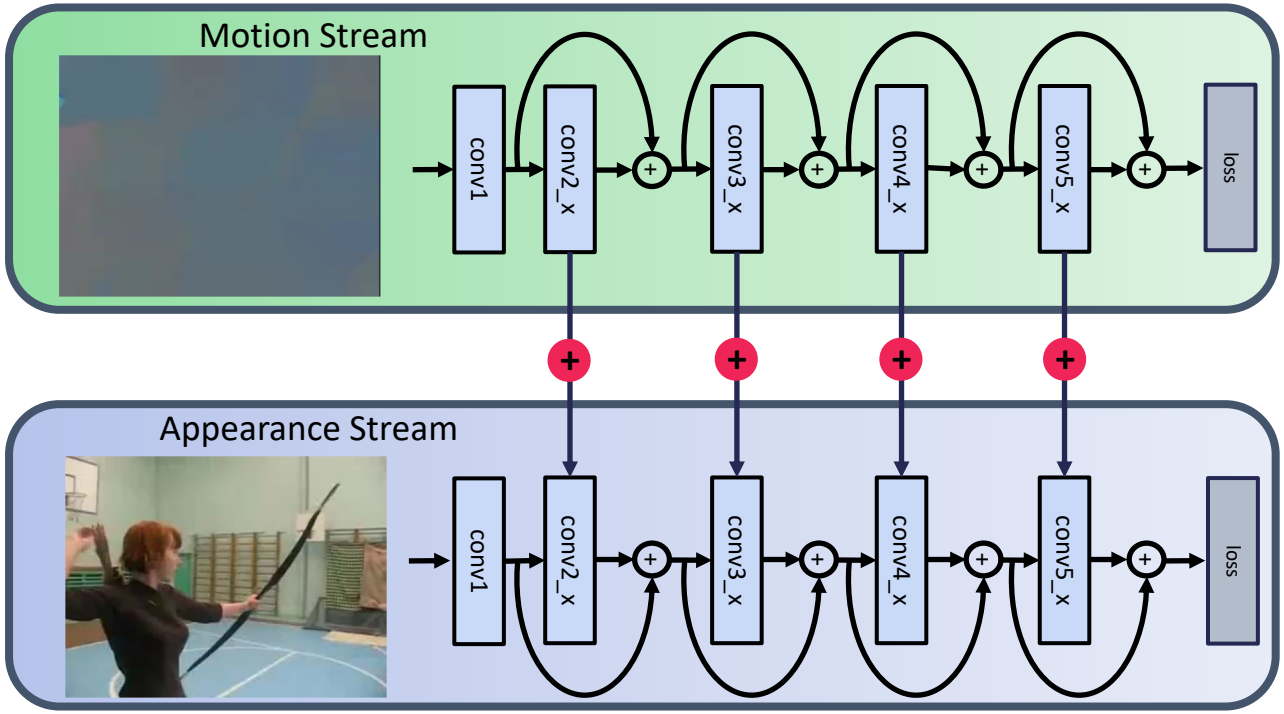
Feichtenhofer, Pinz, Zisserman, CVPR 2016

Convolutional Two-Stre



Feichtenhofer, Pinz, Wildes, Zisserman, CVPR 2018 (in su

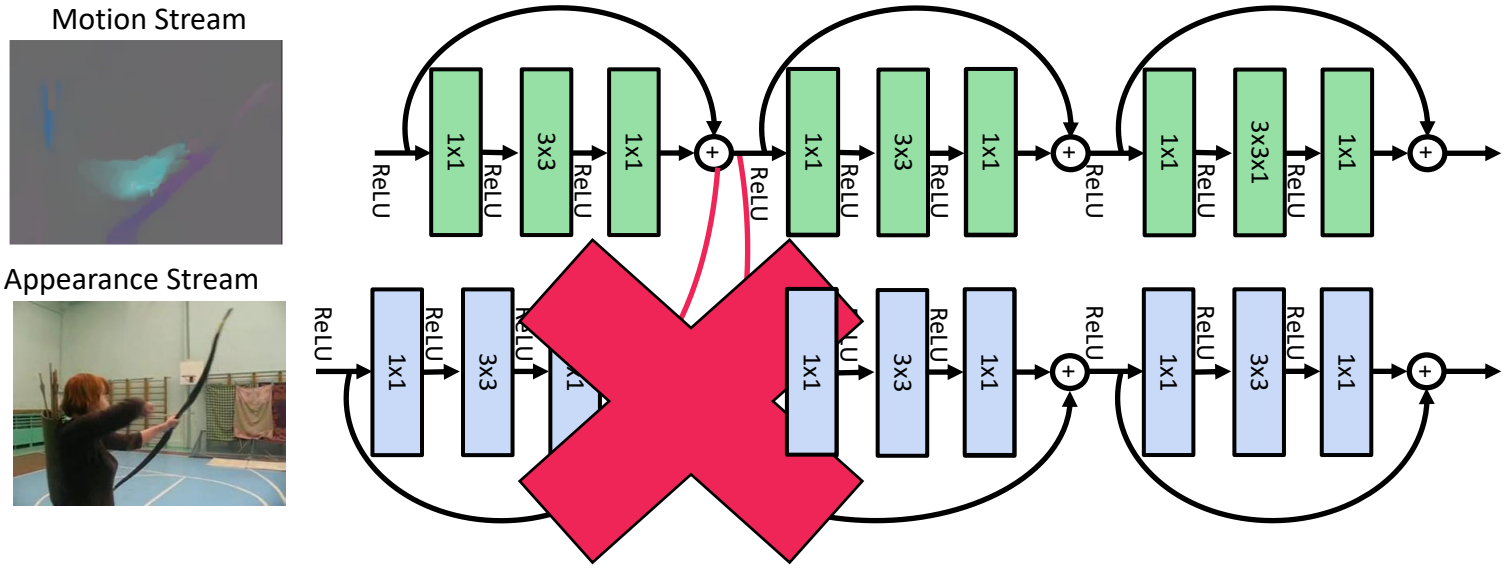
Spatiotemporal Residual Networks



- ST-ResNet allows the hierarchical learning of spacetime features by connecting the appearance and motion channels of a two-stream architecture.
- Though, naive fusion does not work.

Feichtenhofer, Pinz, Wildes, NIPS'16

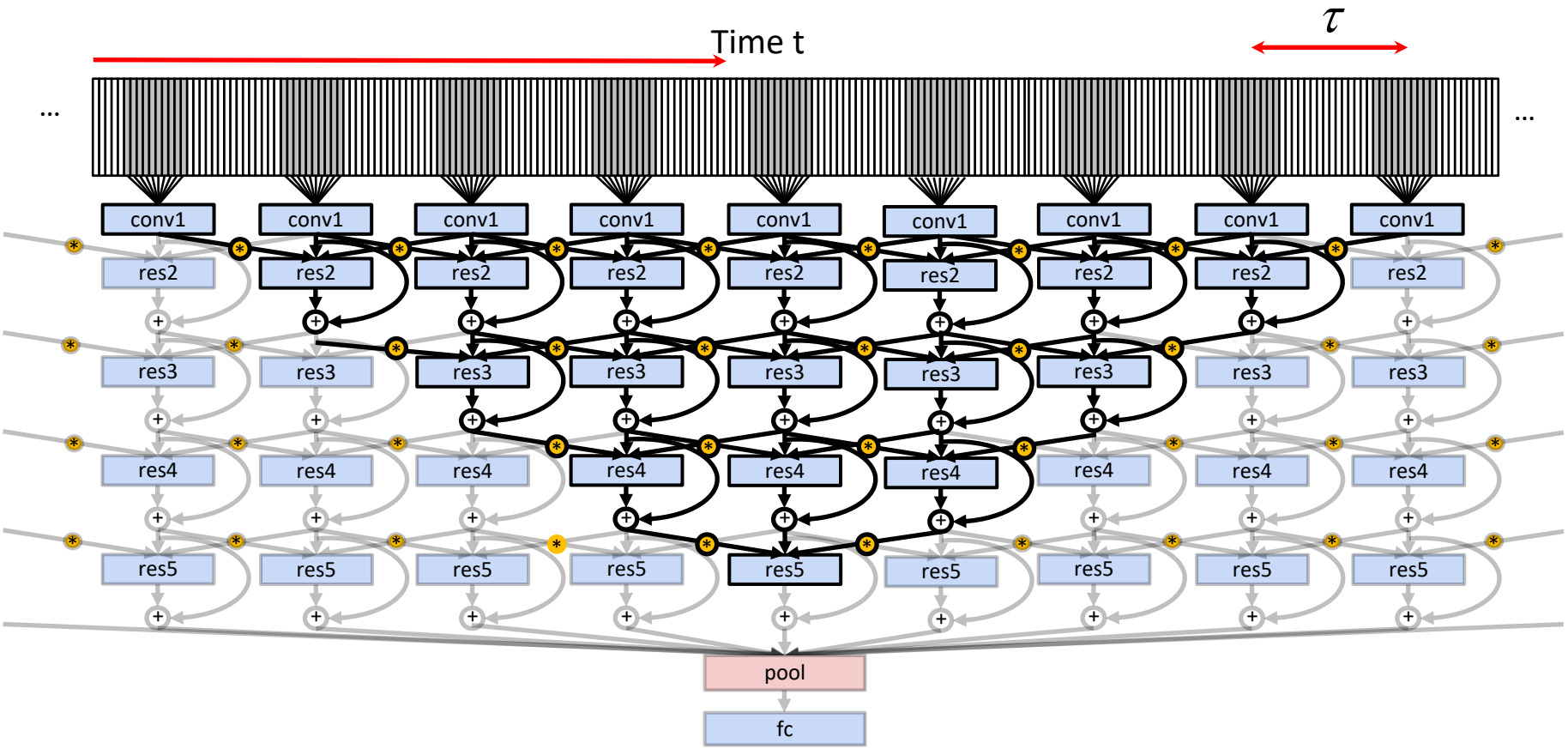
Fusing Two-Stream ResNets & Injecting Temporal Filters



- ResNets for the spatiotemporal domain by introducing residual connections in two ways
  1. Residuals between the motion and appearance pathways to allow spatiotemporal interaction between the streams
  2. Transformation * of pretrained image ConvNets by filters initialized as residuals in time

Feichtenhofer, Pinz, Wildes, NIPS'16

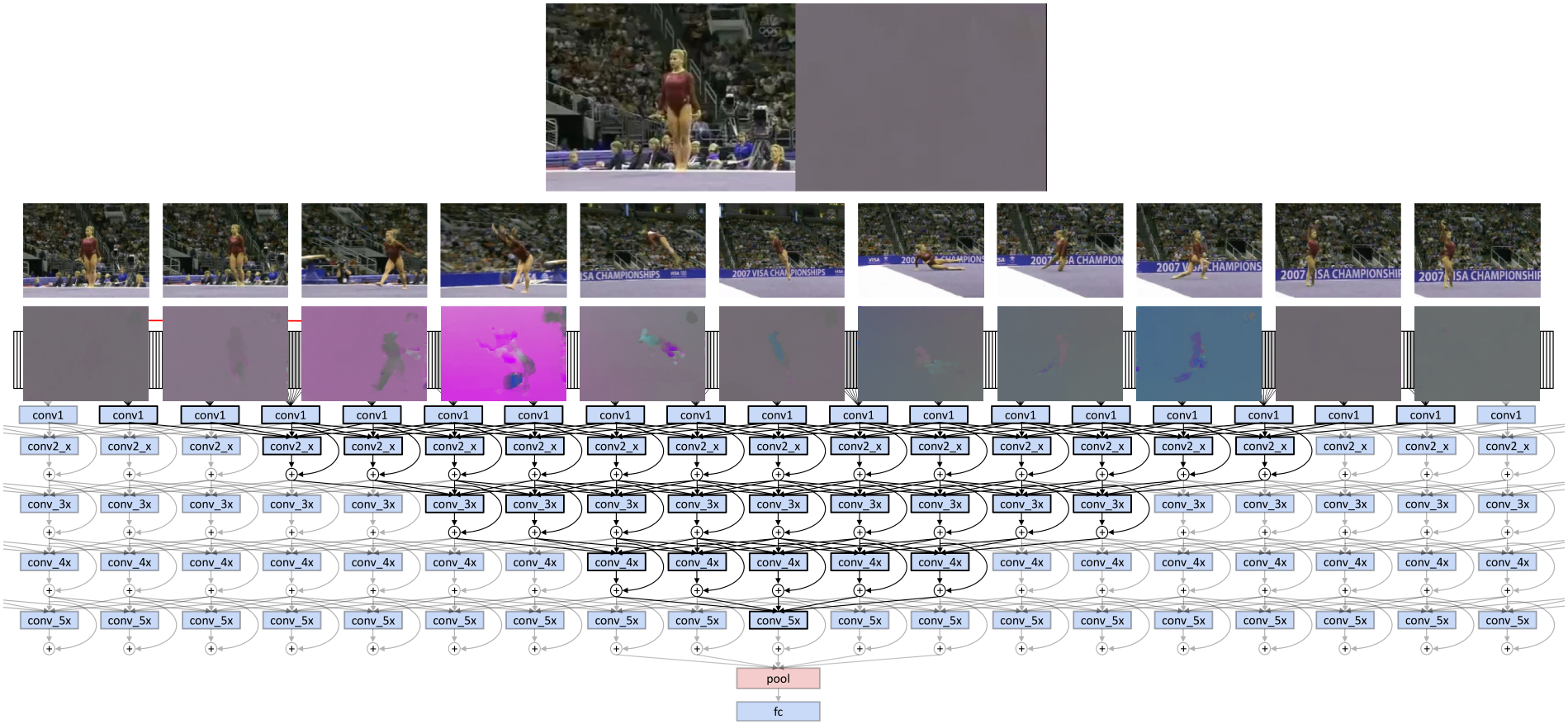
Increasing the temporal receptive field of ResNets



- The temporal receptive field is modulated by the temporal filters * and input stride **τ**

Feichtenhofer, Pinz, Wildes, NIPS'16

Transforming spatial filters to spatiotemporal ones



- Chaining temporal filters supports *hierarchical* learning of long-term correspondences between features of the appearance and motion stream.
- For example, if the stride is set to $\tau = 15$ frames and we transform 8 filters, a unit at conv5_3 sees a window of $17 \times 15 = 255$ frames.

Feichtenhofer, Pinz, Wildes, NIPS'16

Spatiotemporal Residual Networks: Architecture

Layers	conv1	pool1	conv2_x	conv3_x	conv4_x	conv5_x	pool5
Blocks	$7\times7\times1$, 64	$3\times3\times1$ max stride 2	$\begin{bmatrix}1\times1\times1, 64\\3\times3\times1, 64\\1\times1\times1, 256\end{bmatrix}$ skip-stream $\begin{bmatrix}1\times1\times3, 64\\3\times3\times1, 64\\1\times1\times3, 256\end{bmatrix}$ $\begin{bmatrix}1\times1\times1, 64\\3\times3\times1, 64\\1\times1\times1, 256\end{bmatrix}$	$\begin{bmatrix}1\times1\times1, 128\\3\times3\times1, 128\\1\times1\times1, 512\end{bmatrix}$ skip-stream $\begin{bmatrix}1\times1\times3, 128\\3\times3\times1, 128\\1\times1\times3, 512\end{bmatrix}$ $\begin{bmatrix}1\times1\times1, 128\\3\times3\times1, 128\\1\times1\times1, 512\end{bmatrix}\times2$	$\begin{bmatrix}1\times1\times1, 256\\3\times3\times1, 256\\1\times1\times1, 1024\end{bmatrix}$ skip-stream $\begin{bmatrix}1\times1\times3, 256\\3\times3\times1, 256\\1\times1\times3, 1024\end{bmatrix}$ $\begin{bmatrix}1\times1\times1, 256\\3\times3\times1, 256\\1\times1\times1, 1024\end{bmatrix}\times4$	$\begin{bmatrix}1\times1\times1, 512\\3\times3\times1, 512\\1\times1\times1, 2048\end{bmatrix}$ skip-stream $\begin{bmatrix}1\times1\times3, 512\\3\times3\times1, 512\\1\times1\times3, 2048\end{bmatrix}$ $\begin{bmatrix}1\times1\times1, 512\\3\times3\times1, 512\\1\times1\times1, 2048\end{bmatrix}$	$7\times7\times1$ avg $1\times1\times5$ max stride 2
Output size	$112\times112\times11$	$56\times56\times11$	$56\times56\times11$	$28\times28\times11$	$14\times14\times11$	$7\times7\times11$	$1\times1\times4$
Recept. Field	$7\times7\times1$	$11\times11\times1$	$35\times35\times5\tau$	$99\times99\times9\tau$	$291\times291\times13\tau$	$483\times483\times17\tau$	$675\times675\times47\tau$

- The base network is a 50 layer ResNet in each stream.
- Building blocks show width, height, temporal extent and number of feature channels.
- Brackets outline residual units equipped with skip connections.
- Skip-stream indicates residual connections between streams.
- The temporal receptive field is modulated by the temporal stride τ.
- For example, if the stride is set to $\tau = 15$ frames, a unit at conv5_3 sees a window of $17\times15 = 255$ frames.

Feichtenhofer, Pinz, Wildes, NIPS'16 & CVPR'17a

Spatiotemporal Multiplier Networks

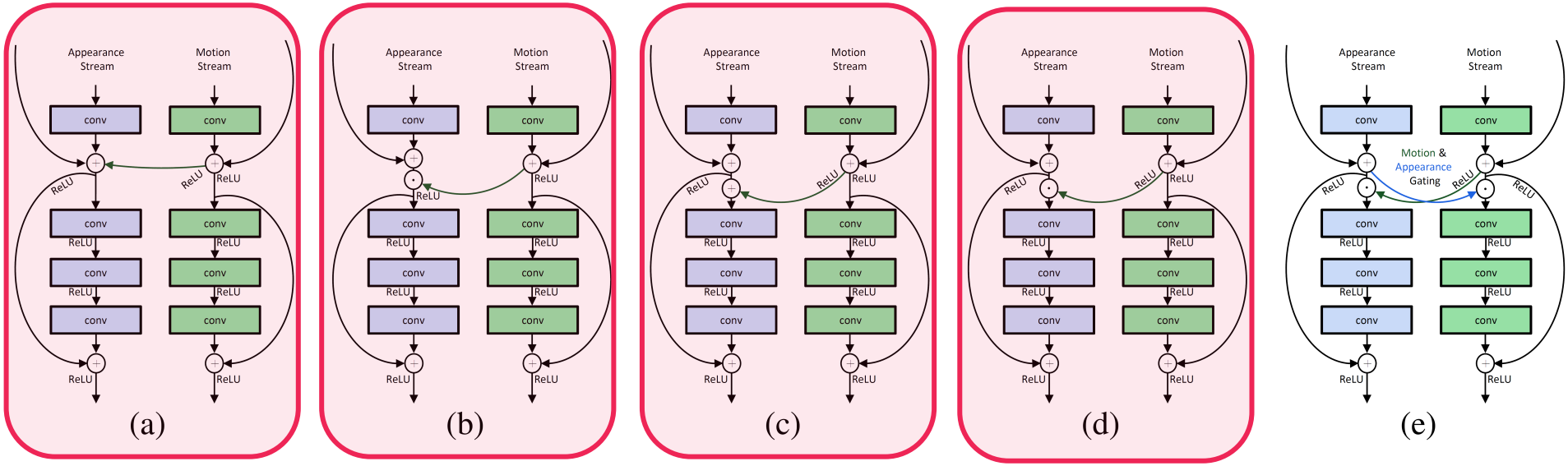
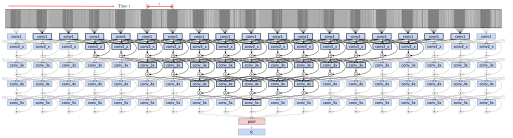


Fig. 2

Error rates (in %) on UCF101-split 1

case	into	Fig.	UCF101	HMDB51
direct ⊕	←	Fig. 2(a)	24.78	54.85
direct ⊙	←	Fig. 2(b)	81.98	77.89
residual ⊕ Sect. 3.2.1	←	Fig. 2(c)	9.38	41.89
residual ⊙ Sect. 3.2.2	←	Fig. 2(d)	8.72	37.23
residual ⊕	→	Fig. 2(c)	16.76	49.54
residual ⊙	→	Fig. 2(d)	16.68	48.43
residual ⊙	↔	Fig. 2(e)	15.15	48.56

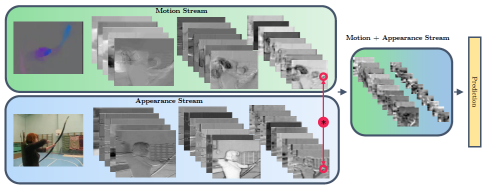
with temporal filters



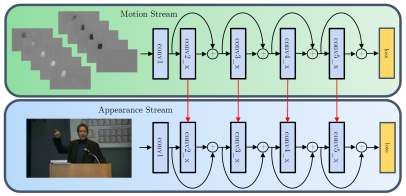
➔ 6.00 | 30.98

Feichtenhofer, Pinz, Wildes, CVPR'17a

Comparison with the state-of-the-art



CVPR‘16



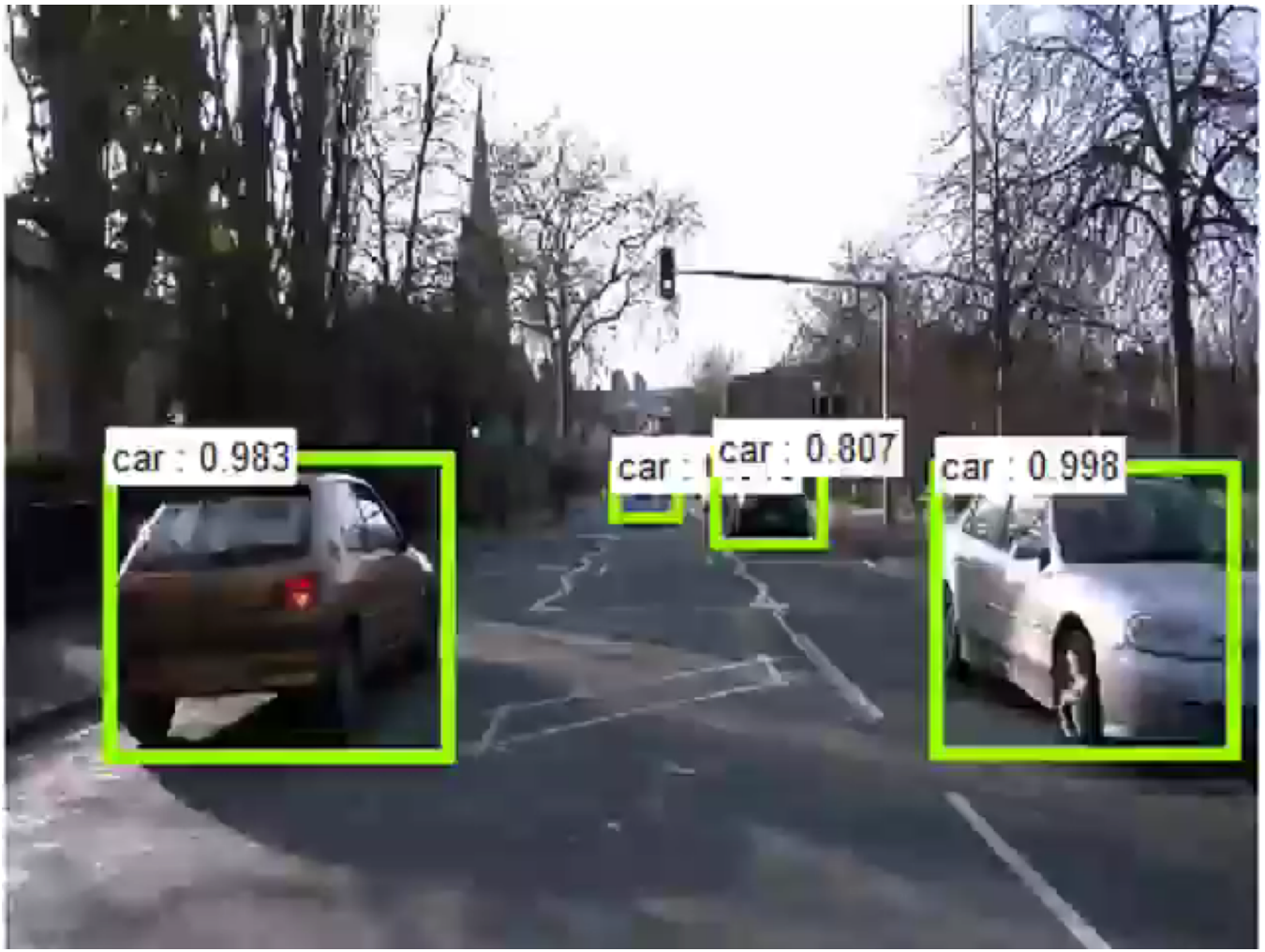
NIPS‘16

CVPR17a

Method	UCF101	HMDB51
Improved Dense Trajectories (IDT) (Wang and Schmid, 2013b)	86.4%	61.7%
Spatiotemporal ConvNet (Karpathy et al, 2014a)	65.4%	-
Two-Stream ConvNet (Simonyan and Zisserman, 2014a)	88.0%	59.4%
Long-term recurrent ConvNet (Donahue et al, 2015)	82.9%	-
Composite LSTM Model (Srivastava et al, 2015)	84.3%	44.0
Two-Stream+LSTM (Ng et al, 2015b)	88.6%	-
C3D (Tran et al, 2015)	85.2%	-
C3D + IDT (Tran et al, 2015)	90.4%	-
Dynamic Image Nets (Bilen et al, 2016)	76.9%	42.8 %
Dynamic Image Nets (Bilen et al, 2016) + IDT	89.1%	65.2%
Transformations(Wang et al, 2016a)	92.4%	62.0%
Two-Stream Fusion(Feichtenhofer et al, 2016b)	92.5%	65.4%
Two-Stream Fusion(Feichtenhofer et al, 2016b) + IDT	93.5%	69.2%
Long-term ConvNets (Varol et al, 2016)	91.7%	64.8%
Long-term ConvNets (Varol et al, 2016) + IDT	92.7%	67.2%
VideoLSTM + IDT (Li et al, 2016b)	92.2%	64.9%
Hierarchical Attention Nets (Wang et al, 2016b)	92.7%	64.3%
Key Volume Mining (Zhu et al, 2016)	93.1%	63.3%
RNN-FV (Lev et al, 2016) + C3D (Tran et al, 2015) + IDT	94.1%	67.7%
Spatiotemporal ResNets (Feichtenhofer et al, 2016a)	93.4%	66.4%
TSN (Wang et al., 2016) + IDT flow	94.2%	69.0%
Spatiotemporal ResNets (Feichtenhofer et al, 2016a) + IDT	94.6%	70.3%
Ours	**94.2%**	**68.9%**
Ours + IDT	**94.9%**	**72.2%**

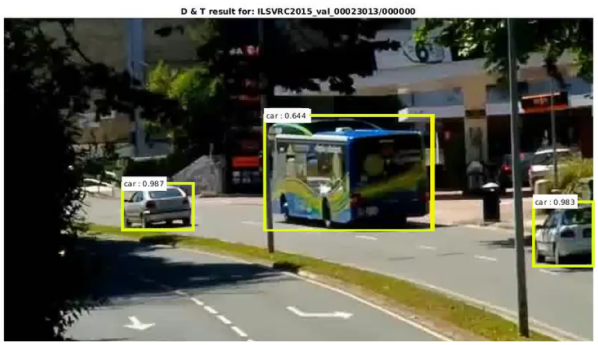
Feichtenhofer, Pinz, Wildes, CVPR’17a

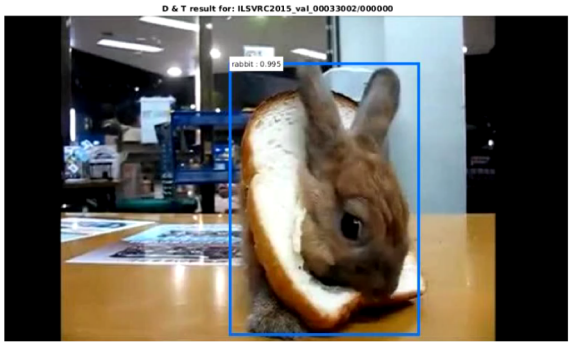
Object Detection in the wild by Faster R-CNN + ResNet-101

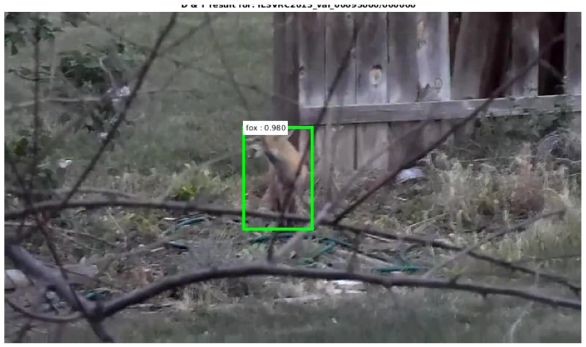


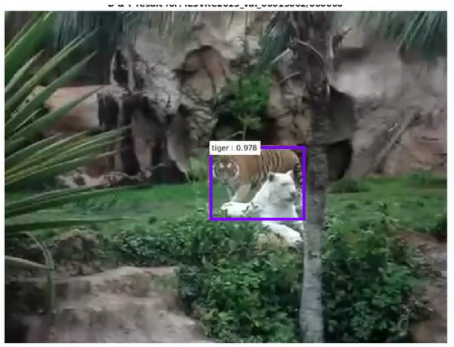
Model pre-trained on ImageNet, fine-tuned on MS COCO that has 80 categories.
Frame-by-frame detection, no temporal processing.

Object Detection and Tracking in Video





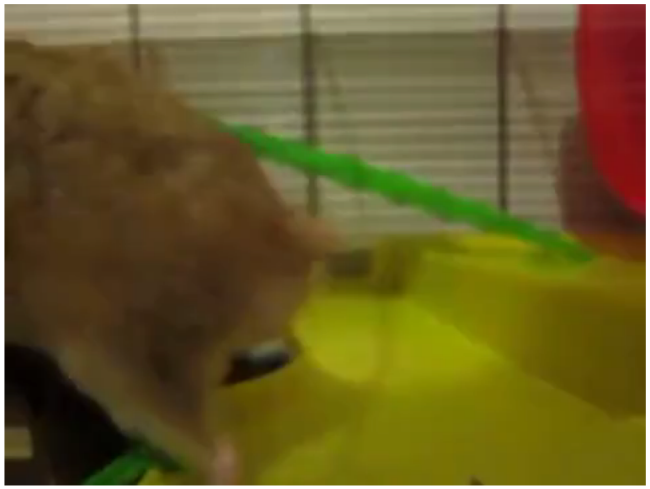




Feichtenhofer, Pinz, Zisserman, ICCV 2017

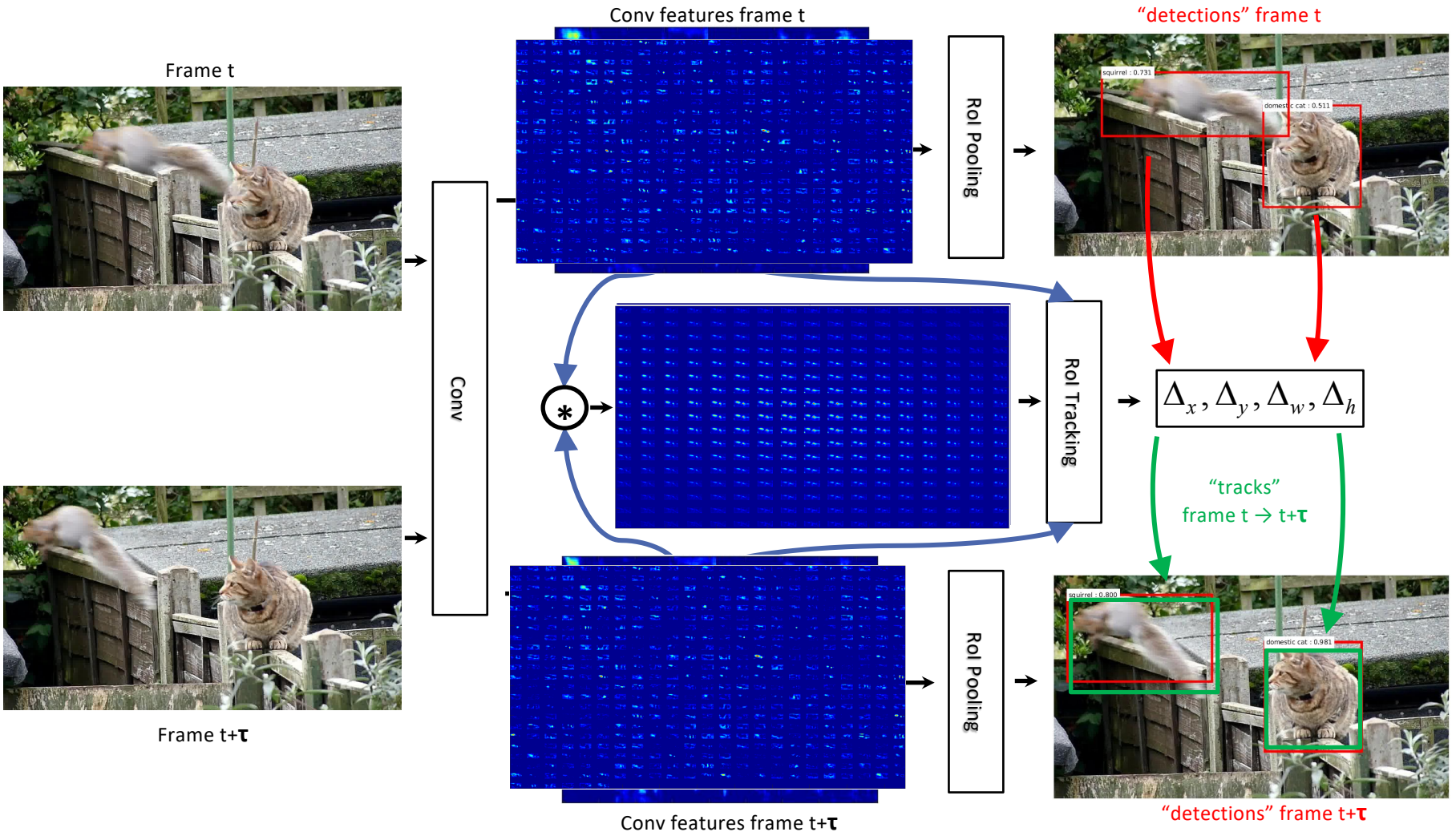
Object Detection from Video: ImageNet VID Challenges

- View point change
- Illumination variation
- Motion blur
- Occlusion



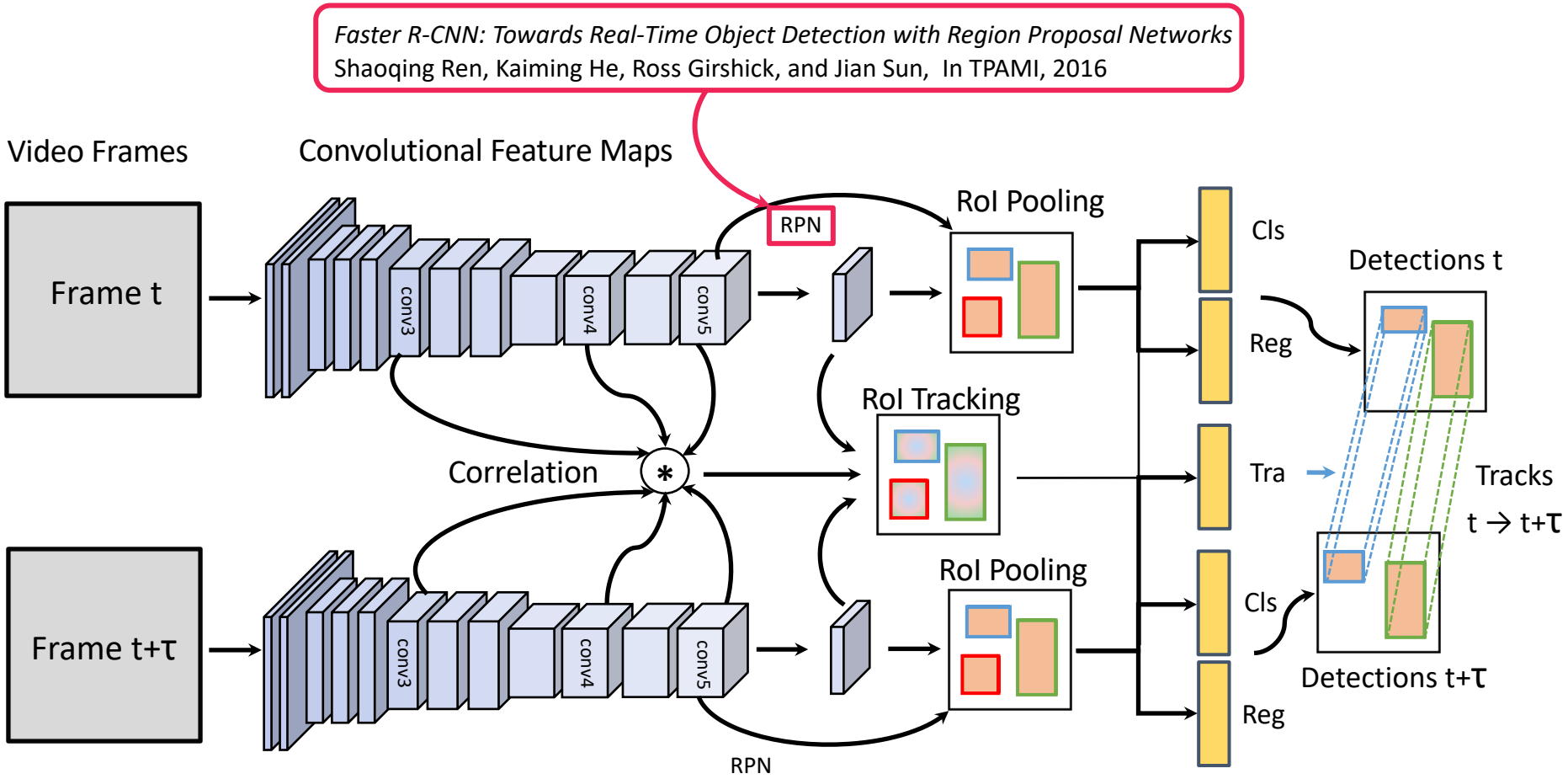
Feichtenhofer, Pinz, Zisserman, ICCV 2017

Detect & Track architecture



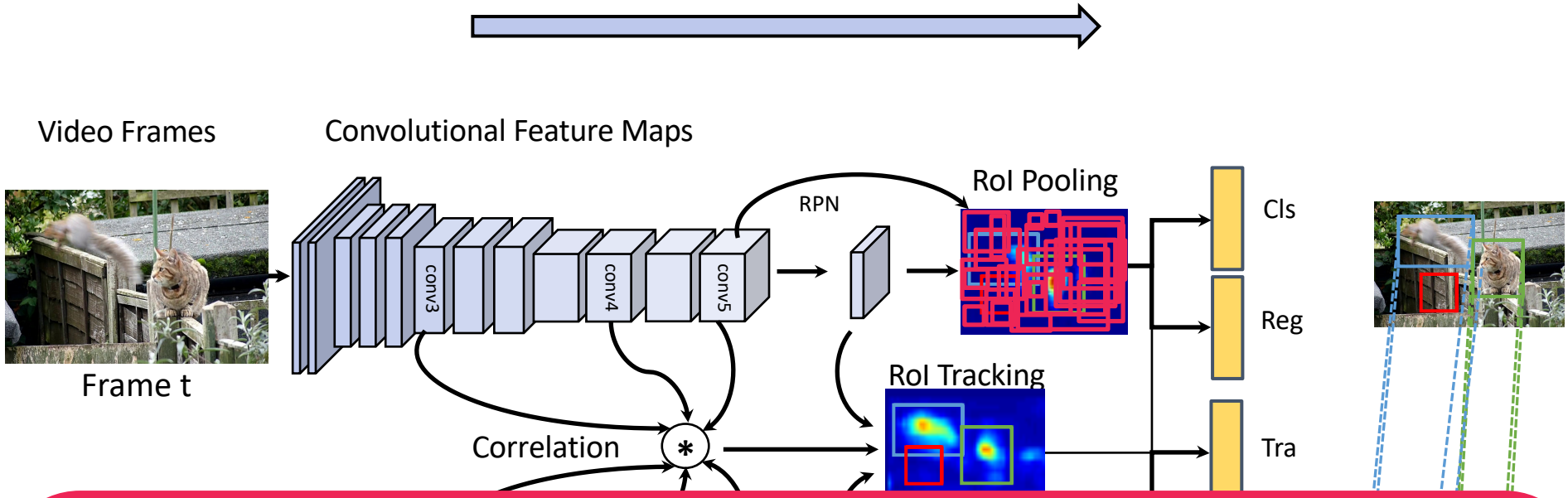
Feichtenhofer, Pinz, Zisserman, ICCV 2017

Detect & Track Architecture



Feichtenhofer, Pinz, Zisserman, ICCV 2017

Detect & Track Training: Forward pass



- Multi-task objective for frame-based object detection and across-frame track regression

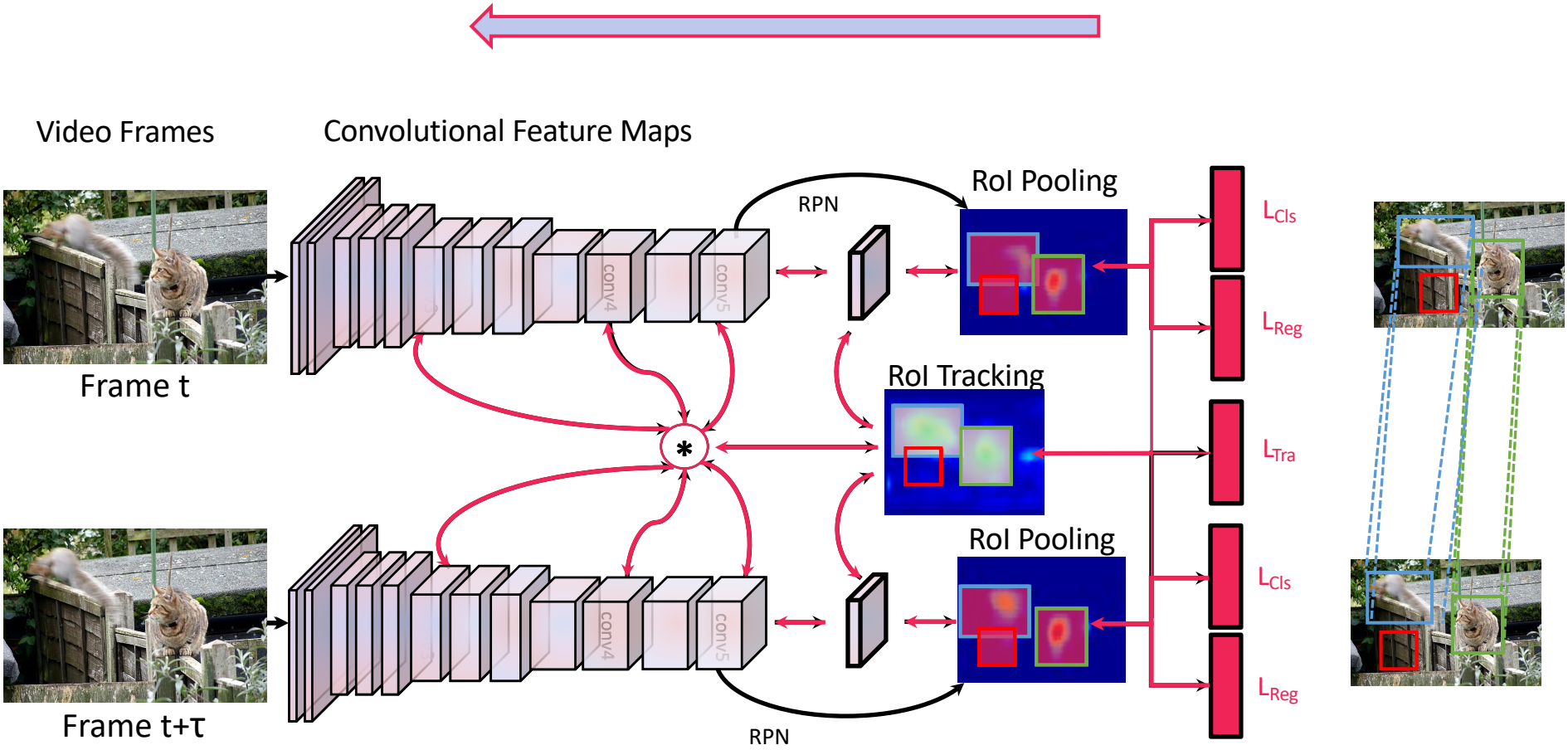
$$L(\{p_i\},\{b_i\},\{\Delta_i\}) = \frac{1}{N}\sum_{i=1}^{N} L_{cls}(p_{i,c^*}) + \lambda\frac{1}{N_{fg}}\sum_{i=1}^{N}[c_i^* > 0]L_{reg}(b_i, b_i^*) + \lambda\frac{1}{N_{tra}}\sum_{i=1}^{N_{tra}} L_{tra}(\Delta_i^{t+\tau}, \Delta_i^{*,t+\tau})$$

Classification softmax scores

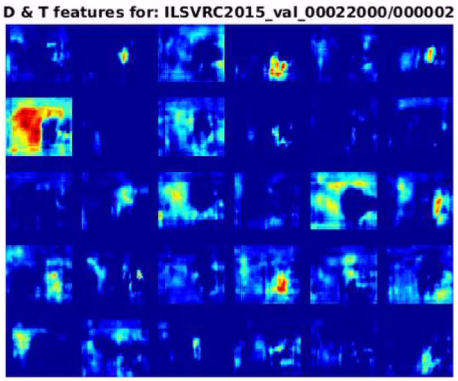
Frame RoIs

Track RoIs

Detect & Track Training: Backward pass

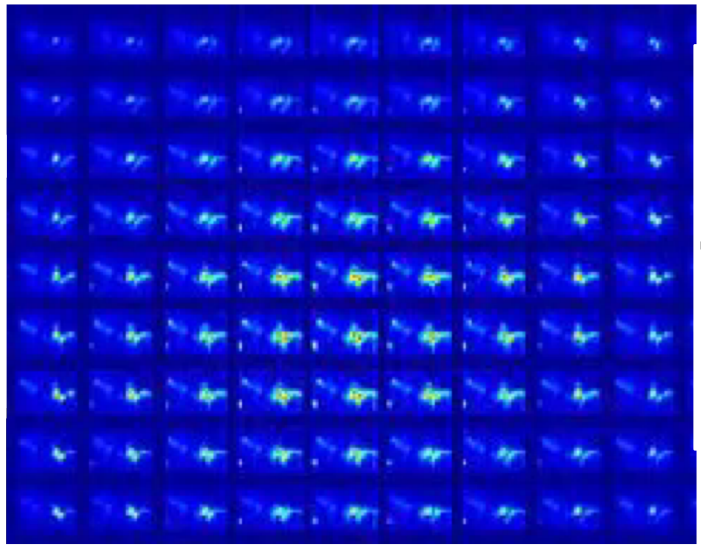


Detect & Track: Testing



RoI Pooling

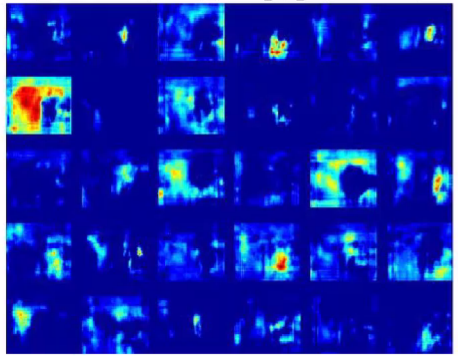
"detections" frame t



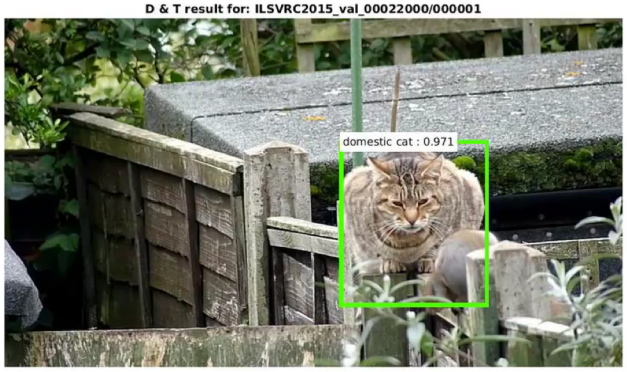
RoI Tracking

$\Delta_x, \Delta_y, \Delta_w, \Delta_h$

"tracks"
frame t → t+τ



RoI Pooling

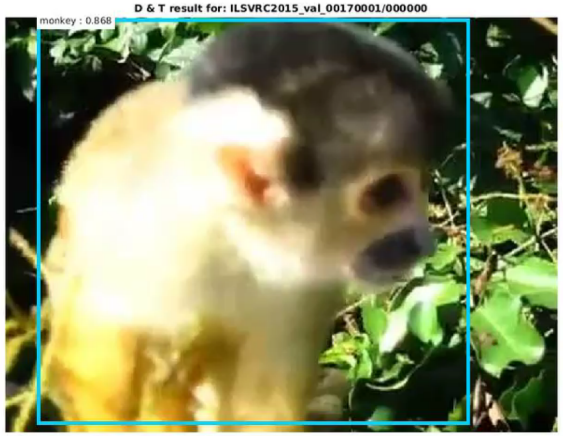


"detections" frame t+τ

Feichtenhofer, Pinz, Zisserman, ICCV 2017

Qualitative Results

Code & Models: **github.com/feichtenhofer/Detect-Track/**



Results: **robots.ox.ac.uk/~vgg/research/Detect-Track/**

Feichtenhofer, Pinz, Zisserman, ICCV 2017

Detect & Track: Qualitative Results

Difficult cases

Feichtenhofer, Pinz, Zisserman, ICCV 2017

Detect & Track: Quantitative Results

- ImageNet video object detection challenge (VID):
  - 30 classes
  - ~1.3M frames
  - 3862 training and 555 validation videos

Methods	airplane	antelope	bear	bicycle	bird	bus	car	cattle	dog	d. cat	elephant	fox	g. panda	hamster	horse	lion
TCN [4]	72.7	75.5	42.2	39.5	25.0	64.1	36.3	51.1	24.4	48.6	65.6	73.9	61.7	82.4	30.8	34.4
TPN+LSTM [2]	84.6	78.1	72.0	67.2	68.0	80.1	54.7	61.2	61.6	78.9	71.6	83.2	78.1	91.5	66.8	21.6
Winner ILSVRC'15 [3]	83.7	85.7	84.4	74.5	73.8	75.7	57.1	58.7	72.3	69.2	80.2	83.4	80.5	93.1	84.2	67.8
D (R-FCN)	87.4	79.4	84.5	67.0	72.1	84.6	54.6	72.9	70.9	77.3	76.7	89.7	77.6	88.5	74.8	57.9
D (& T loss)	89.4	80.4	83.8	70.0	71.8	82.6	56.8	71.0	71.8	76.6	79.3	89.9	83.3	91.9	76.8	57.3
D&T ($\tau = 1$)	90.2	82.3	87.9	70.1	73.2	87.7	57.0	80.6	77.3	82.6	83.0	97.8	85.8	96.6	82.1	66.7
D&T ($\tau = 10$)	89.1	79.8	87.5	68.8	72.9	86.1	55.7	78.6	76.4	83.4	82.9	97.0	85.0	96.0	82.2	66.0

Methods	lizard	monkey	motorcycle	rabbit	red panda	sheep	snake	squirrel	tiger	train	turtle	watercraft	whale	zebra	mAP (%)
TCN [4]	54.2	1.6	61.0	36.6	19.7	55.0	38.9	2.6	42.8	54.6	66.1	69.2	26.5	68.6	47.5
TPN+LSTM [2]	74.4	36.6	76.3	51.4	70.6	64.2	61.2	42.3	84.8	78.1	77.2	61.5	66.9	88.5	68.4
Winner ILSVRC'15 [3]	80.3	54.8	80.6	63.7	85.7	60.5	72.9	52.7	89.7	81.3	73.7	69.5	33.5	90.2	73.8
Winner ILSVRC'16 [5]	(single model performance)														76.2
D (R-FCN)	76.8	50.1	80.2	61.3	79.5	51.9	69.0	57.4	90.2	83.3	81.4	68.7	68.4	90.9	74.2
D (& T loss)	79.0	54.1	80.3	65.3	85.3	56.9	74.1	59.9	91.3	84.9	81.9	68.3	68.9	90.9	75.8
D&T ($\tau = 1$)	83.4	57.6	86.7	74.2	91.6	59.7	76.4	68.4	92.6	86.1	84.3	69.7	66.3	95.2	**79.8**
D&T ($\tau = 10$)	83.1	57.9	79.8	72.7	90.0	59.4	75.6	65.4	90.5	85.6	83.3	68.3	66.5	93.2	78.6

Table 1. Performance comparison on the ImageNet VID validation set. The average precision (in %) for each class and the mean average precision over all classes is shown. τ corresponds to the temporal sampling stride.

[1] J. Dai, Yi Li, K. He, and J. Sun R-FCN: Object Detection via Region-based Fully Convolutional Networks. In NIPS, 2016

[2] K. Kang, H. Li, T. Xiao, W. Ouyang, J. Yan, X. Liu, and X. Wang. Object detection in videos with tubelet proposal networks. In CVPR, 2017.

[3] K. Kang, H. Li, J. Yan, X. Zeng, B. Yang, T. Xiao, C. Zhang, Z. Wang, R. Wang, X. Wang, and W. Ouyang. T-CNN: tubelets with convolutional neural networks for object detection from videos. arXiv preprint, 2016.

[4] K. Kang, W. Ouyang, H. Li, and X. Wang. Object detection from video tubelets with convolutional neural networks. In CVPR, 2016.

[5] J. Yang, H. Shuai, Z. Yu, R. Fan, Q. Ma, Q. Liu, and J. Deng. ILSVRC2016 object detection from video: Team NUIST. http://image-net.org/challenges/talks/2016/Imagenet%202016%20VID.pptx , 2016.

Revolution of Depth for Recognition

2012: VGG-M
5 conv. layers

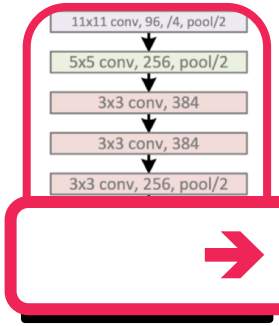
2014: VGG-16
16 conv. layers

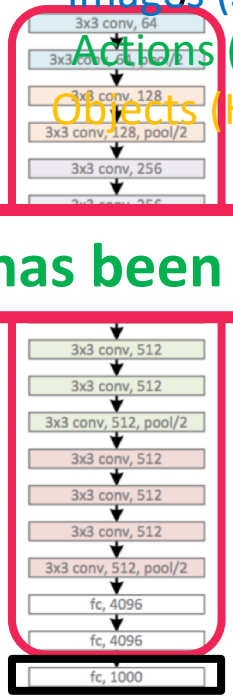
2016: ResNet
>50 conv. layers

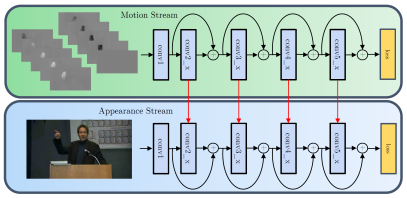
deeper

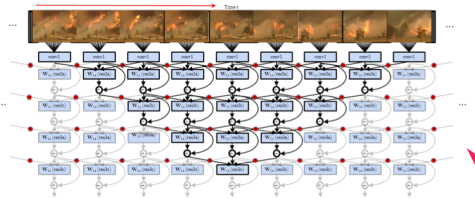
Pre-ConvNet accuracy:
Images (SIFT+FV): 73.8%
Actions (IDT+FV): 85.8%
Objects (HOG+DPM): 34%

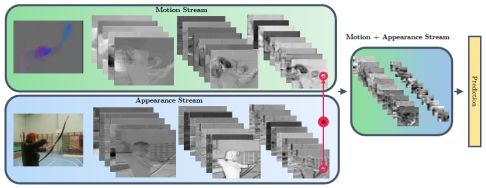
➔ **What has been learned internally?**











ImageNet: 84.7%
UCF101: 88.8%
Pascal VOC: 58%

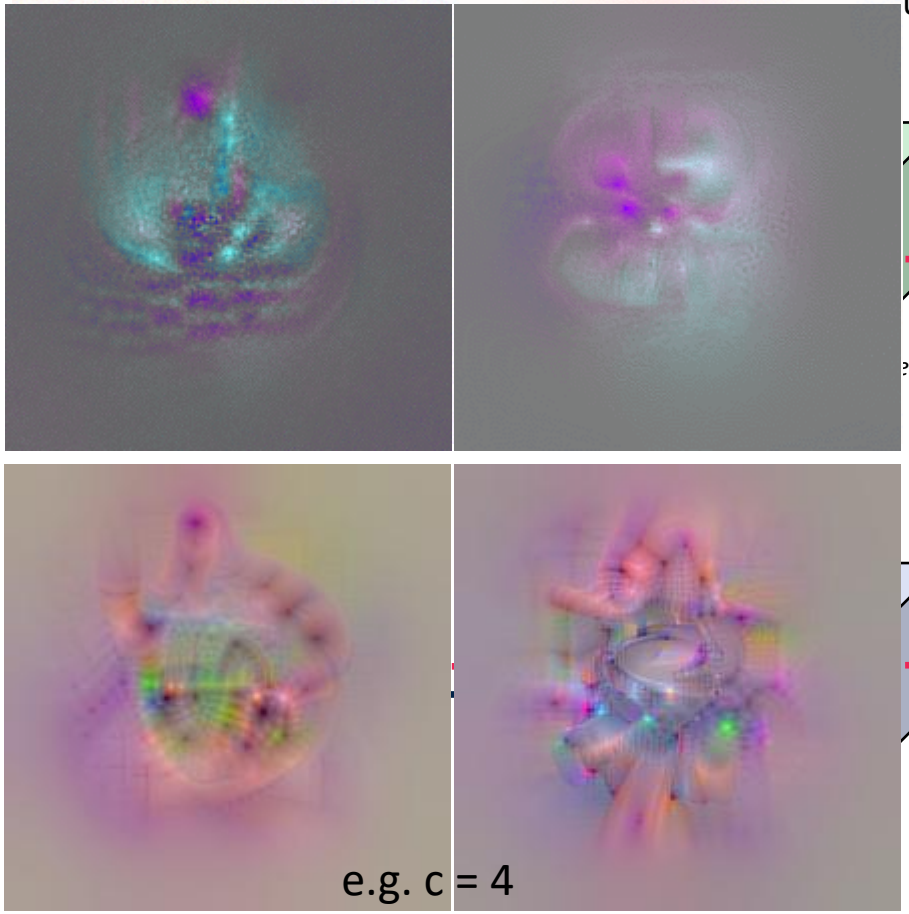
ImageNet: 91.5%
UCF101: 91.8%
Pascal VOC: 66%

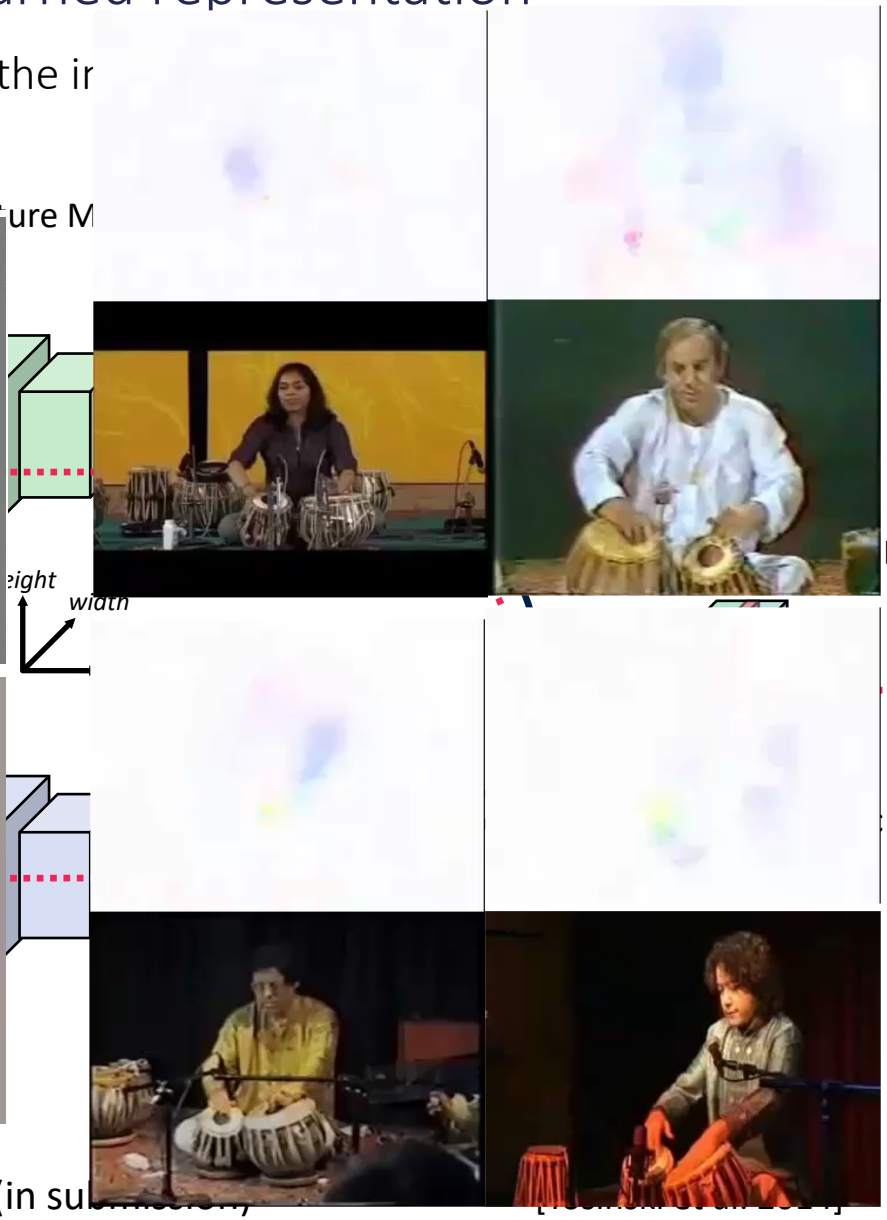
ImageNet: 97.1%
UCF101: 94.2%
Pascal VOC: 86%

Visualizing the learned representation

Slow motion

Fast motion





Feichtenhofer, Pinz, Wildes, Zisserman, CVPR 2018 (in sub

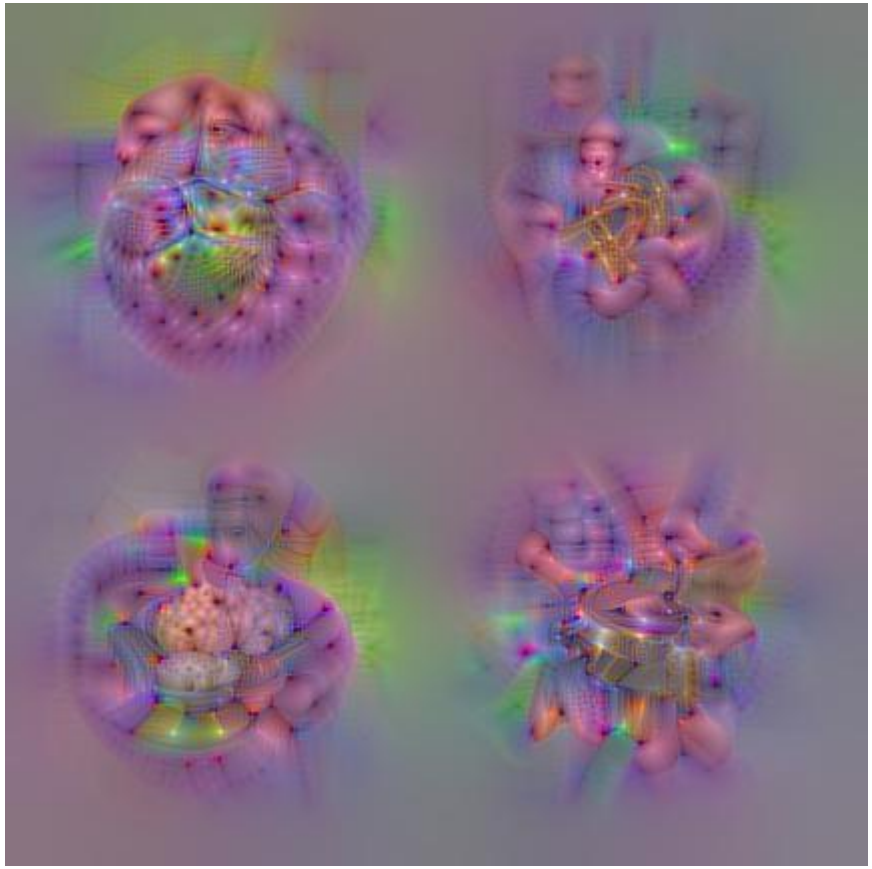
2014]

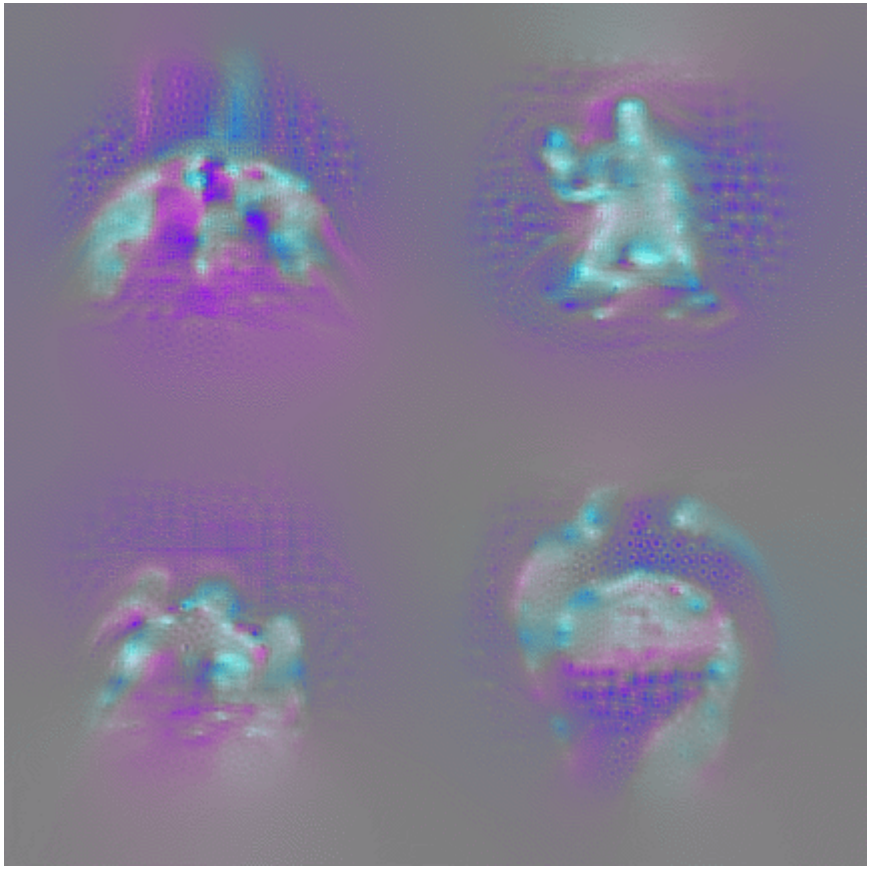
Going through the conv layers of VGG-16 (first four filters of each layer are shown)

Appearance

conv[illegible] f1-4

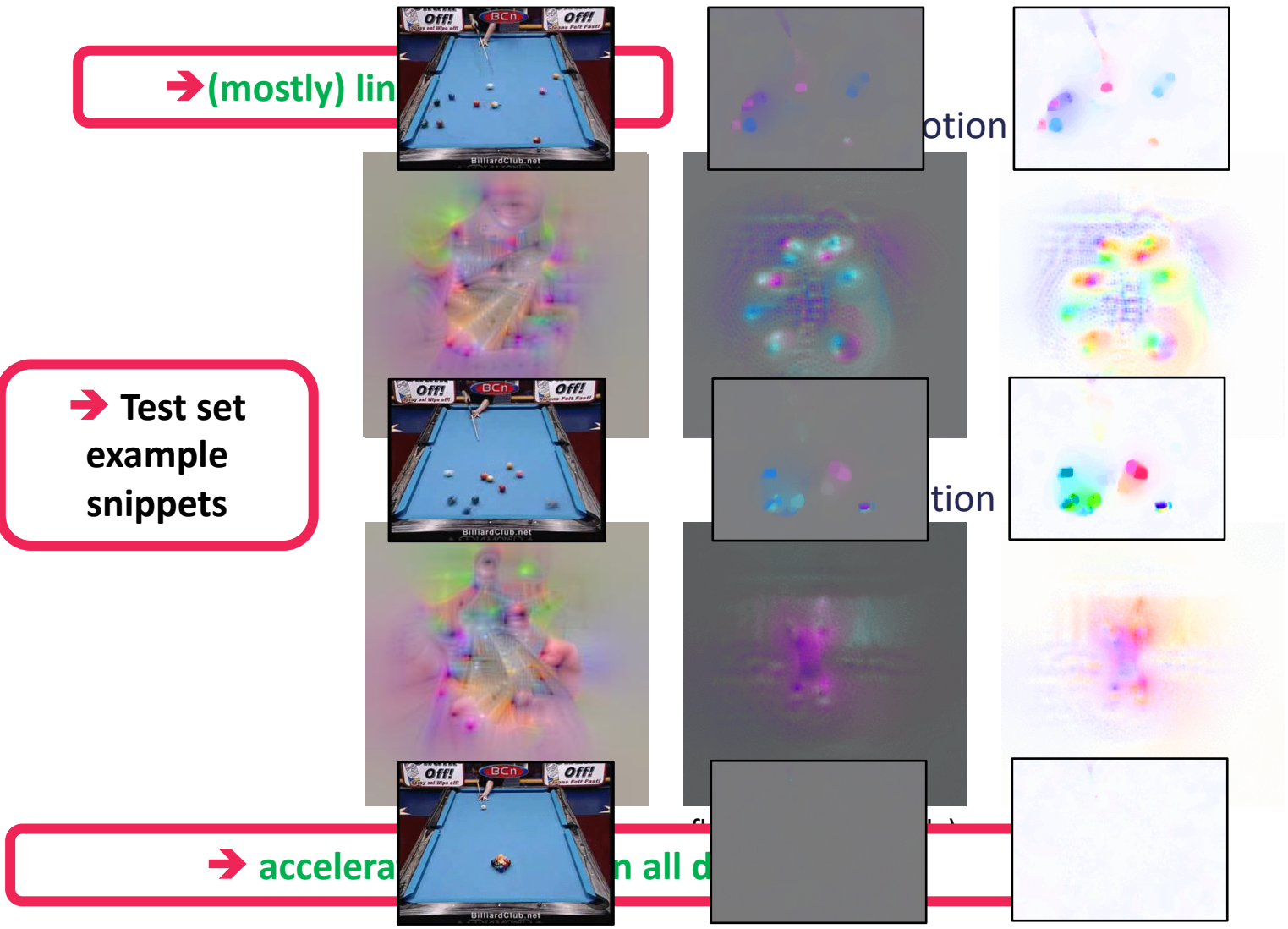
Slow motion



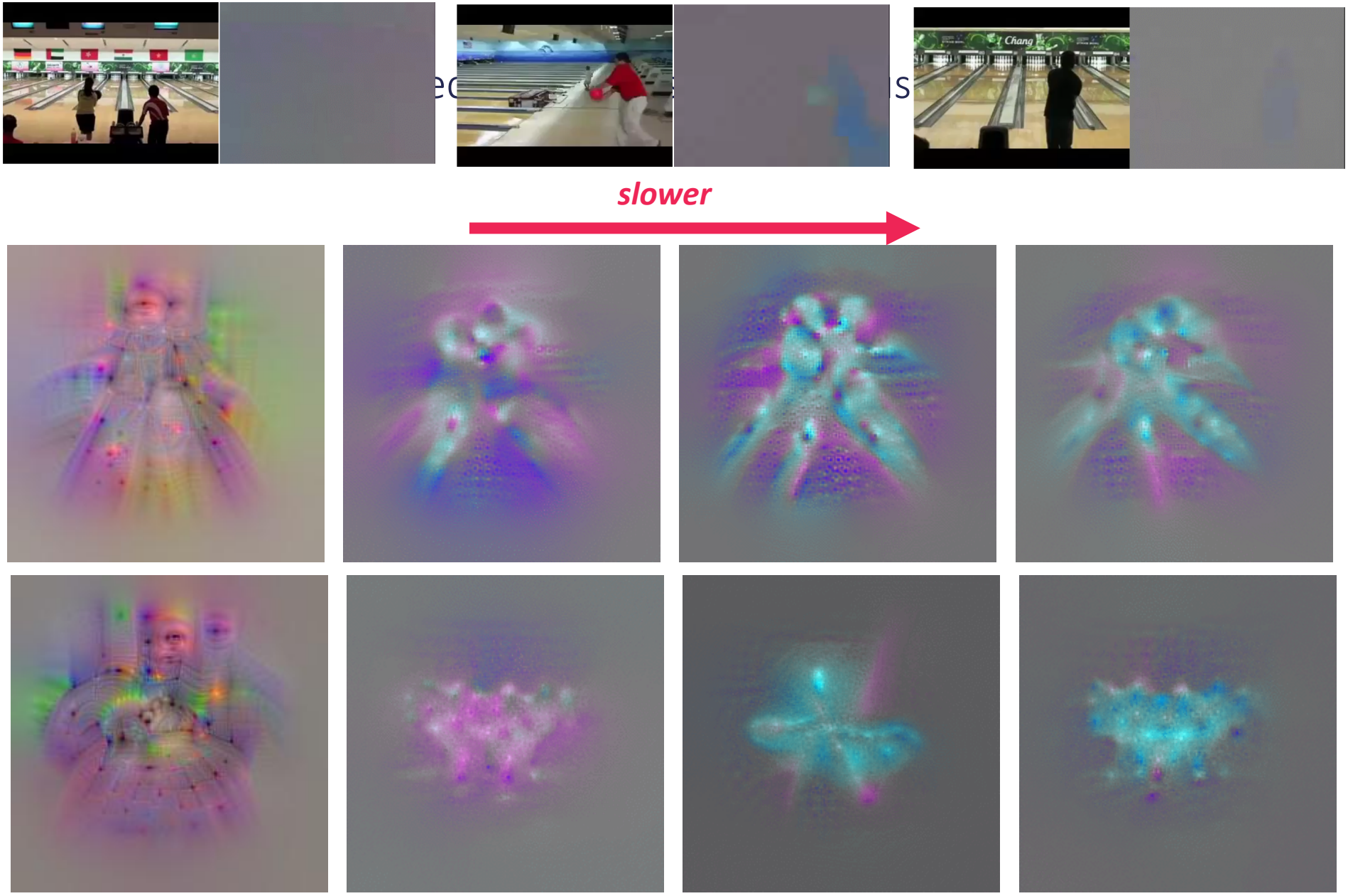


Feichtenhofer, Pinz, Wildes, Zisserman, CVPR 2018 (in submission)

Filter #21 at conv5 fusion – a local Billiard neuron ?

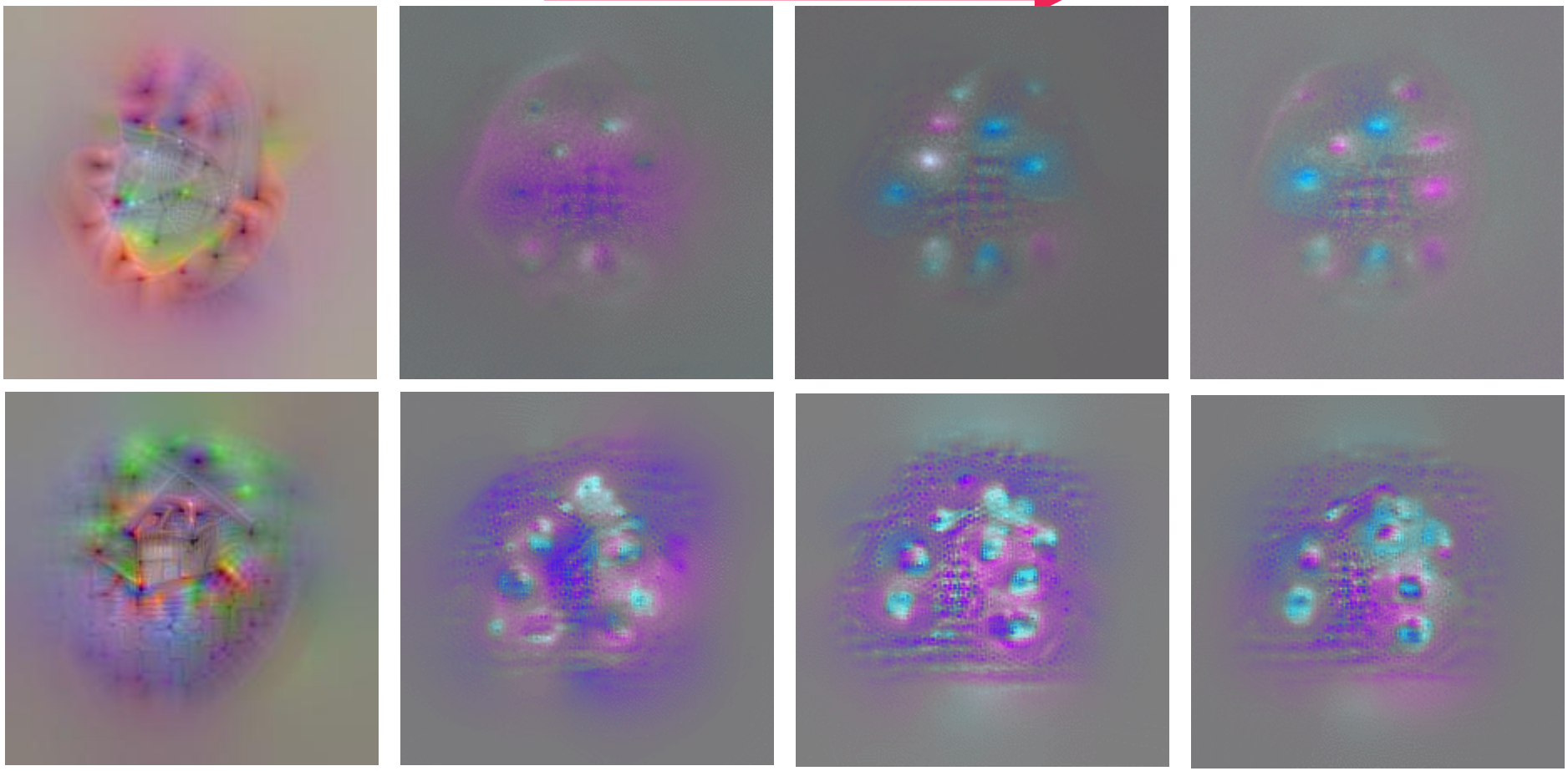


Feichtenhofer, Pinz, Wildes, Zisserman, CVPR 2018 (in submission)



Feichtenhofer, Pinz, Wildes, Zisserman, CVPR 2018 (in submission)

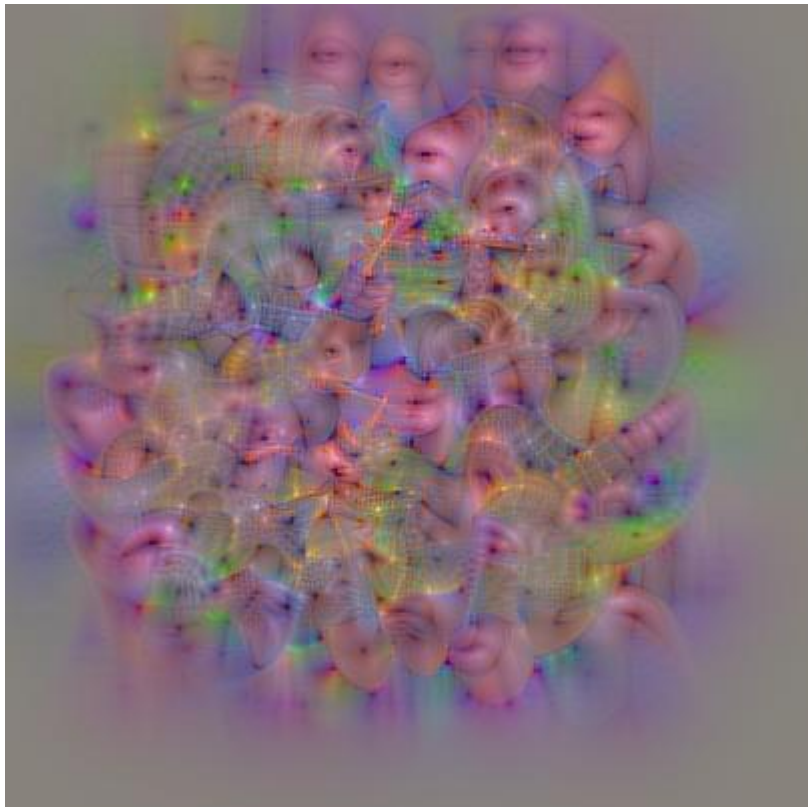
slower



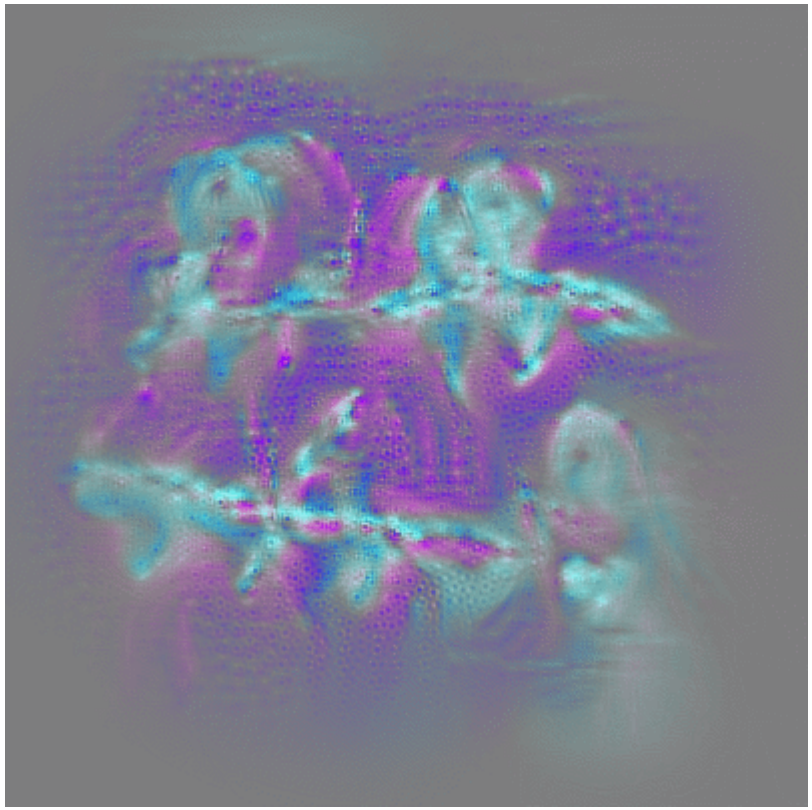
Feichtenhofer, Pinz, Wildes, Zisserman, CVPR 2018 (in submission)

FC 6 (4096 features; RF 404x404)

Appearance



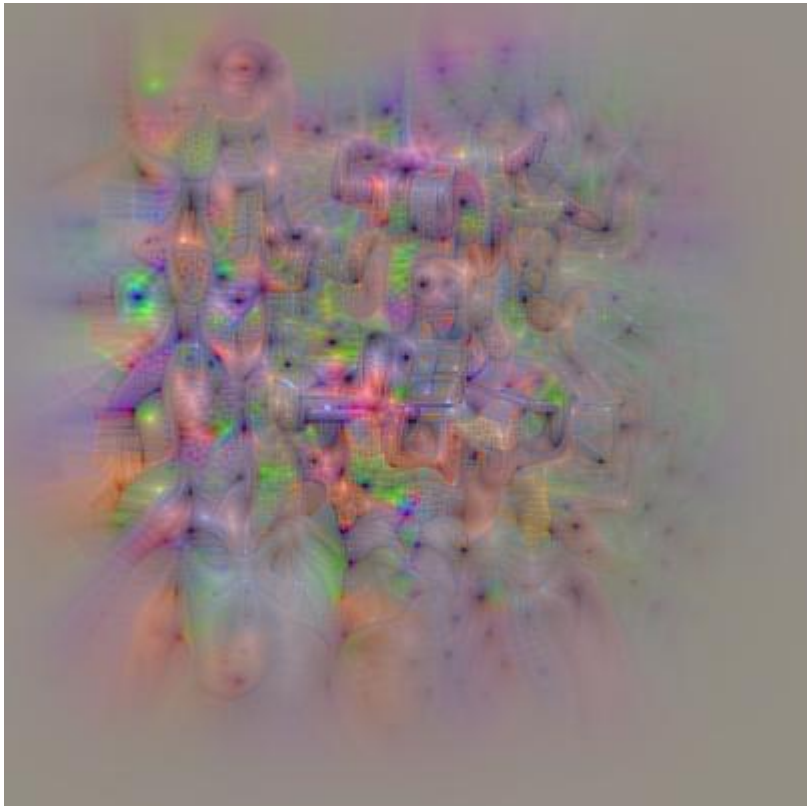
Slow motion



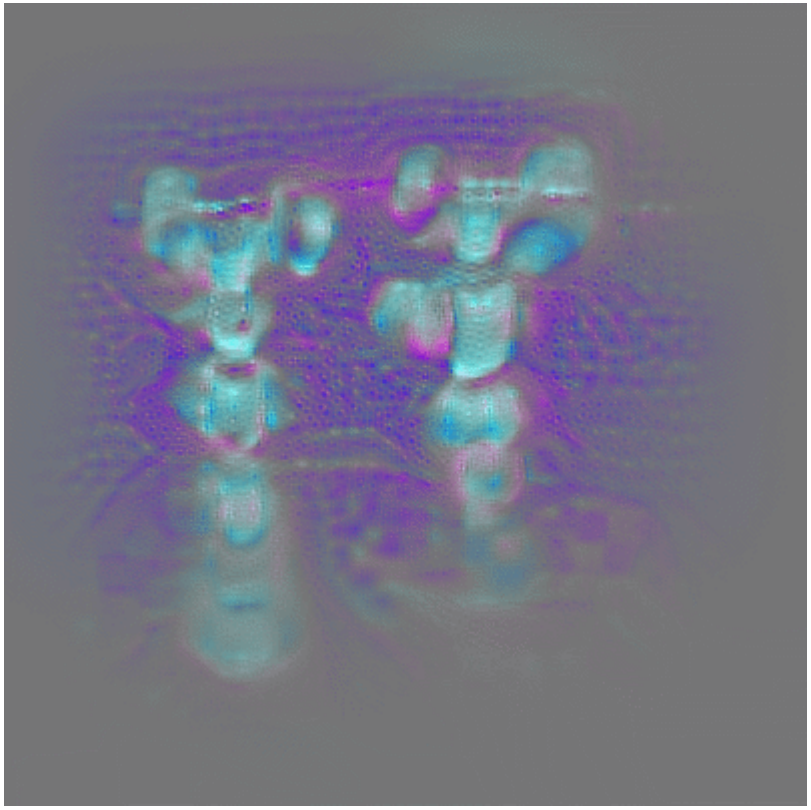
Feichtenhofer, Pinz, Wildes, Zisserman, CVPR 2018 (in submission)

FC 7 (4096 features; RF 404x404)

Appearance



Slow motion



Feichtenhofer, Pinz, Wildes, Zisserman, CVPR 2018 (in submission)

Last layer

➔ “CleanAndJerk”

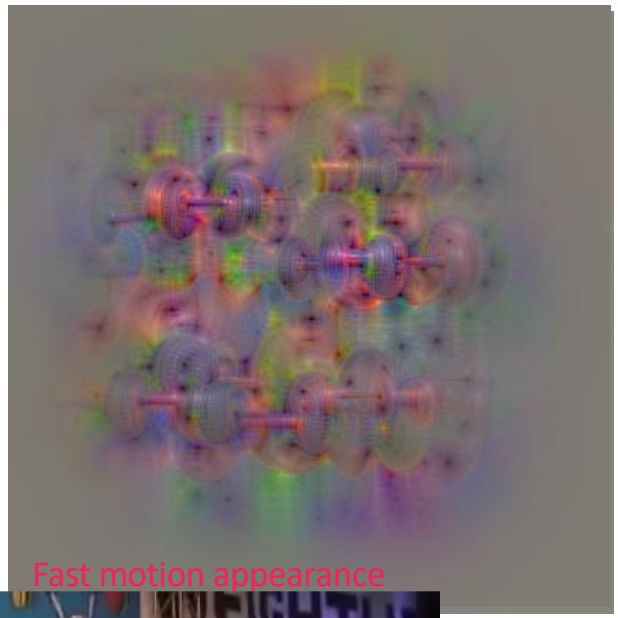
Appearance

Slow motion

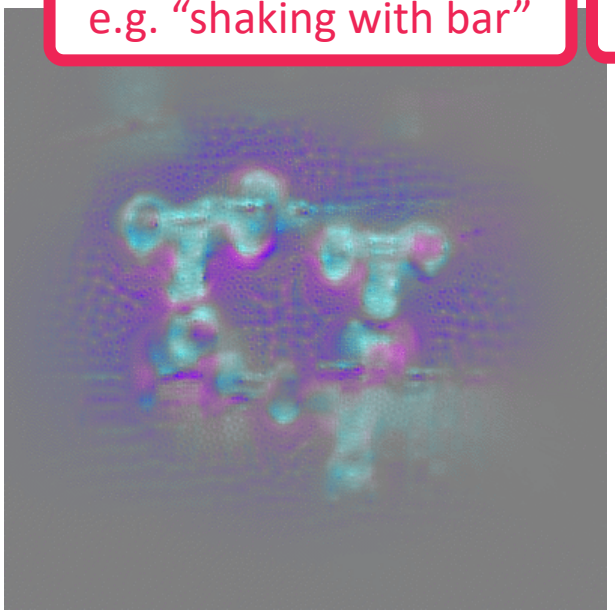
e.g. “shaking with bar”

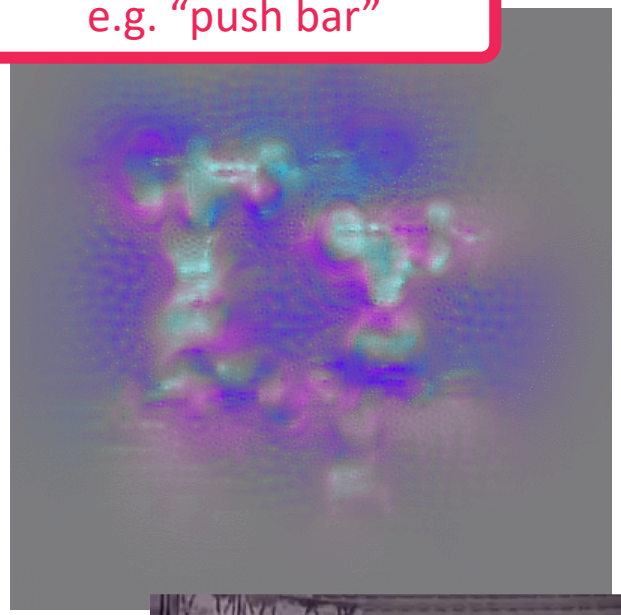
Fast motion

e.g. “push bar”



Fast motion appearance

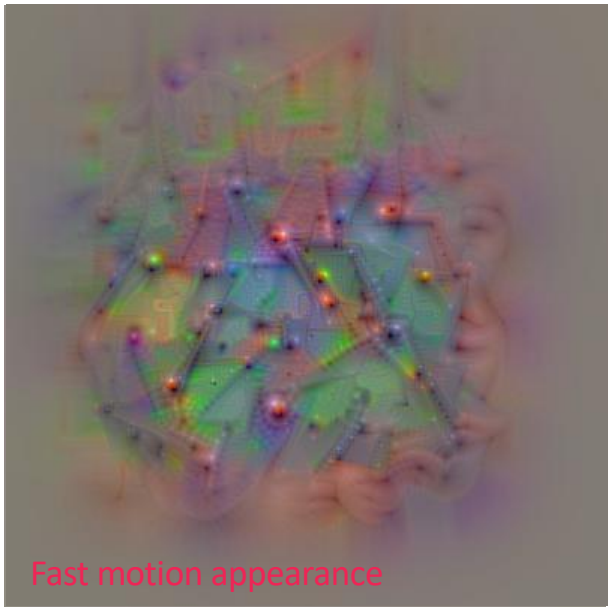




Last layer

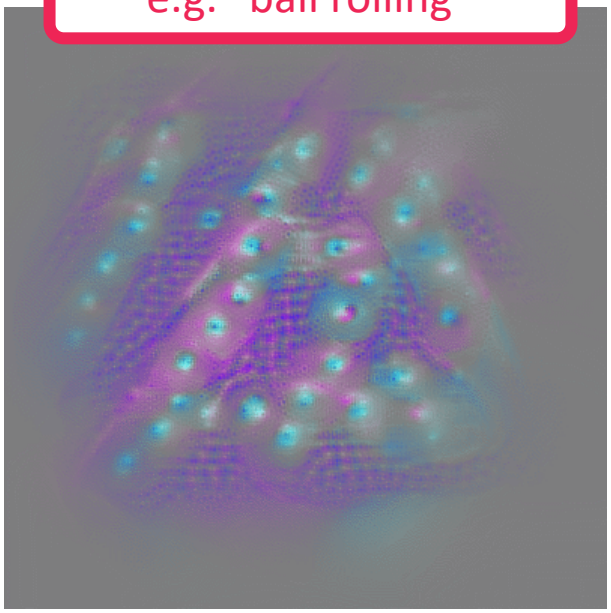
➔ “Billiards”

Appearance



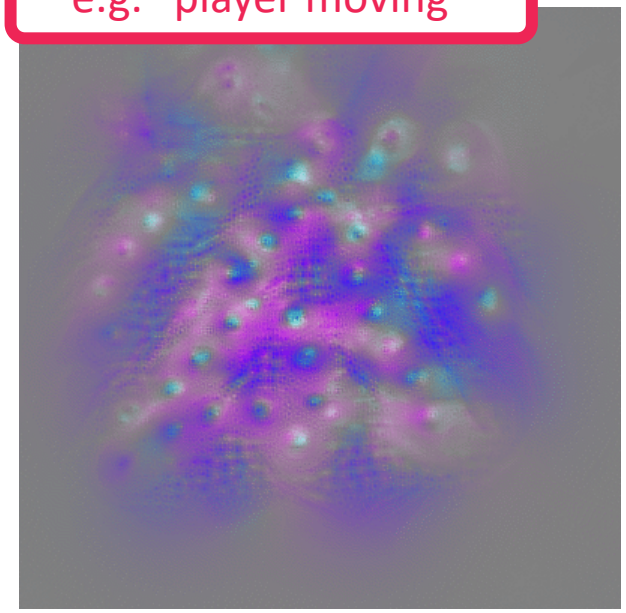
Slow motion

e.g. “ball rolling”



Fast motion

e.g. “player moving”



es, Zisserman, CVPR 2018 (in submission)

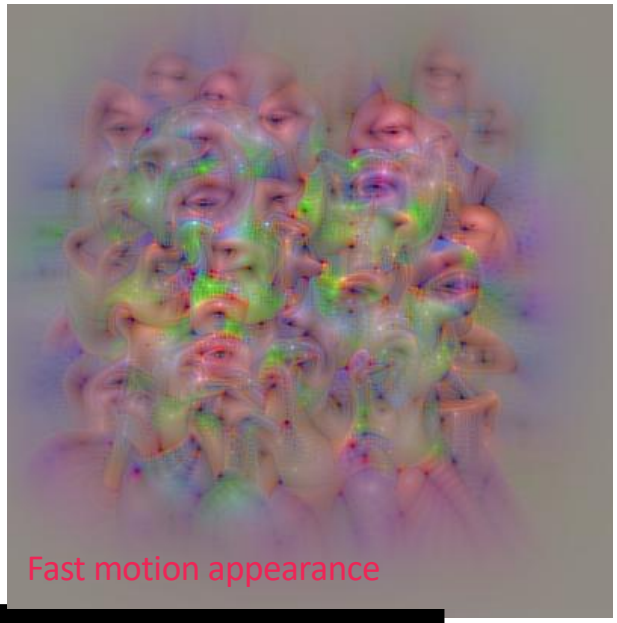
Revealing idiosyncracies in data

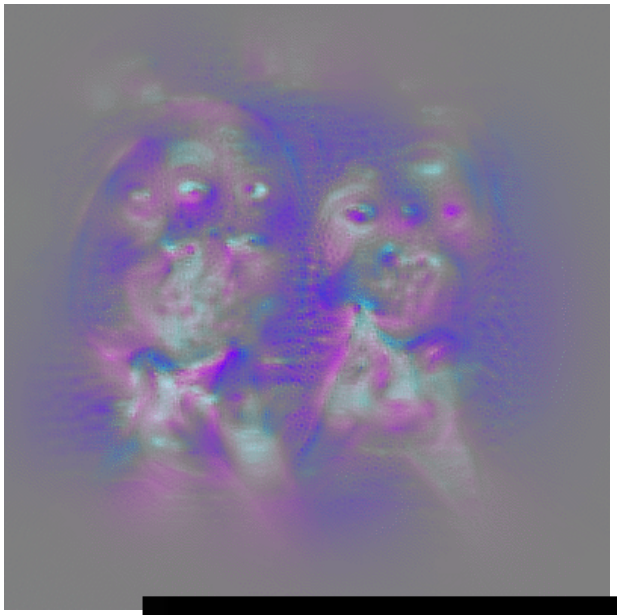
➔ “ApplyLipstick”

Appearance

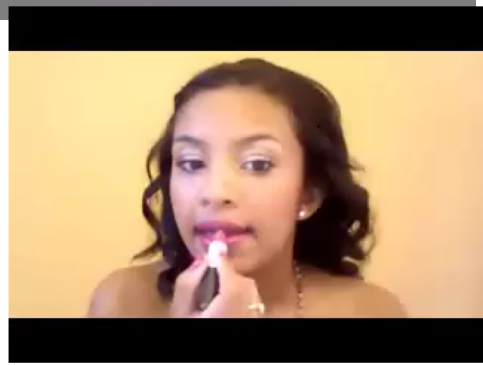
Slow motion

Fast motion





s, Zisserman, CVPR 2018 (in submission)



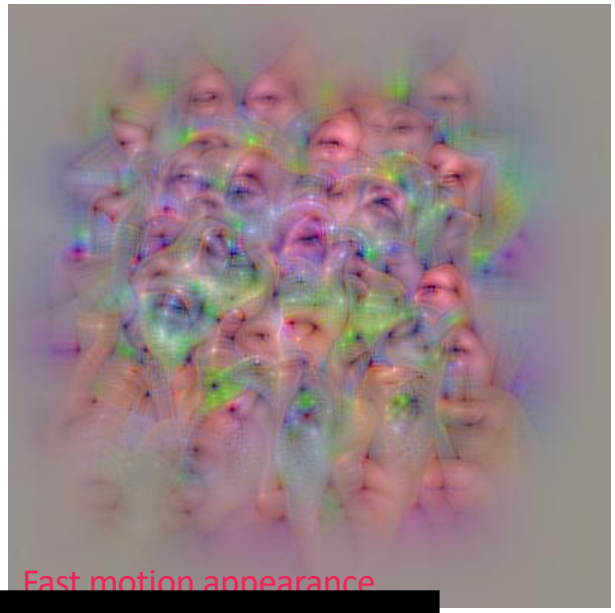
Revealing idiosyncracies in data

➔ “ApplyEyeMakeup”

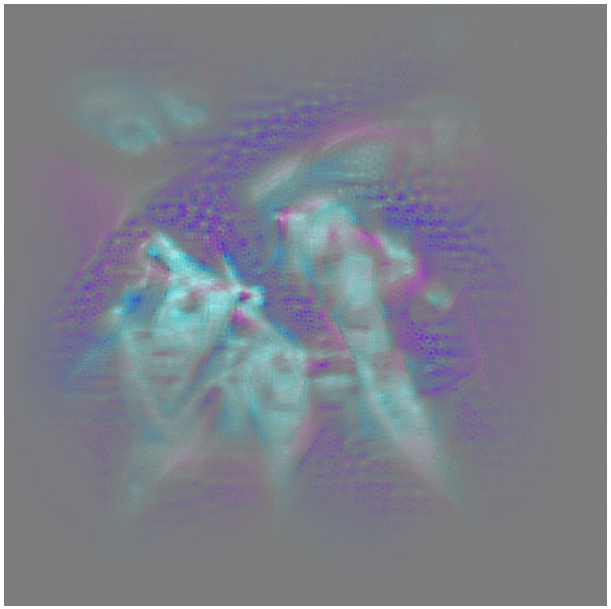
Appearance

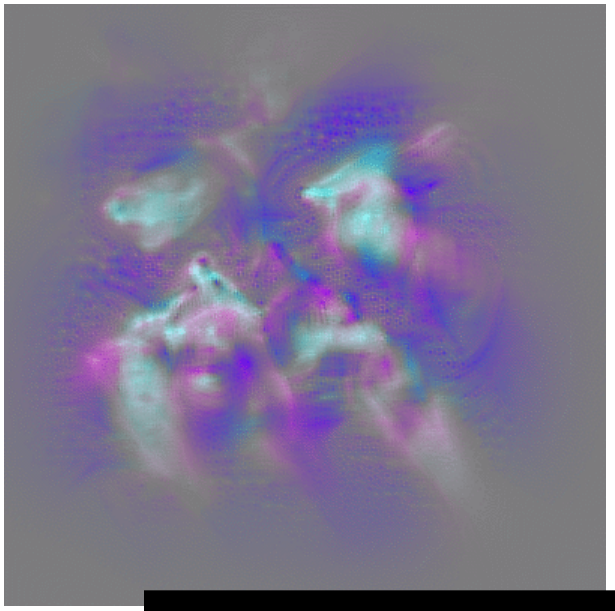
Slow motion

Fast motion



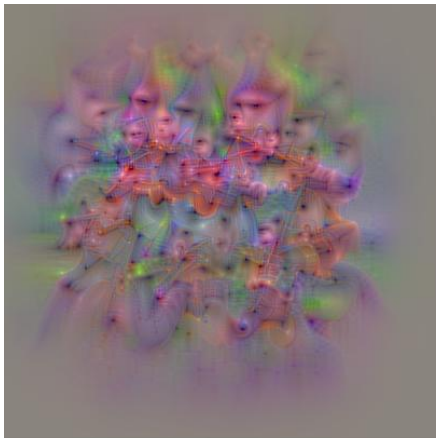
Fast motion appearance

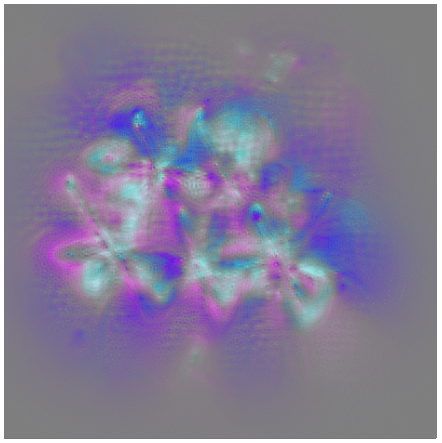


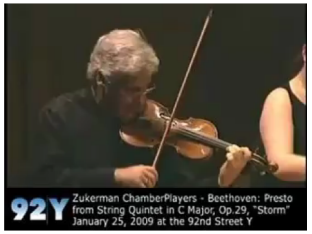


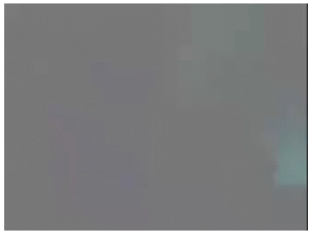
s, Zisserman, CVPR 2018 (in submission)

Explaining failure cases:



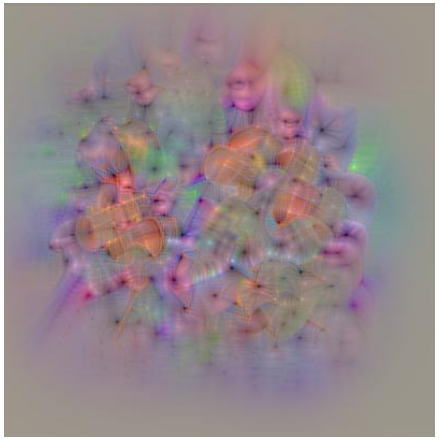


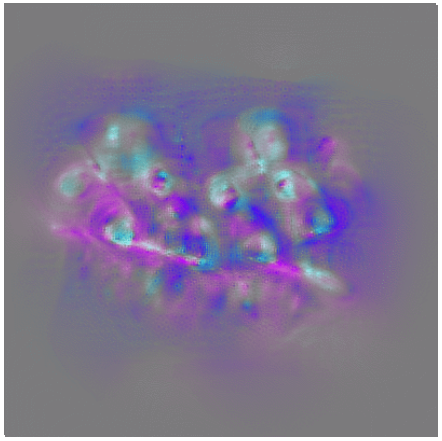


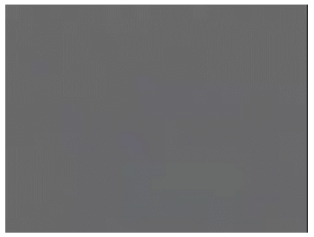


PlayingViolin 84%

(11% confused with PlayingChello)







Feichtenhofer, Pinz, Wildes, Zisserman, CVPR 2018 (in submission)