CSE 252D: Advanced Computer Vision

Manmohan Chandraker

Lecture 4: Optical Flow

Virtual classrooms

- Virtual lectures on Zoom
  - Only host shares the screen
  - Keep video off and microphone muted
  - But please do speak up (remember to unmute!)
  - Slides uploaded on webpage just before class
- Virtual interactions on Zoom
  - Ask and answer plenty of questions
  - “Raise hand” feature on Zoom when you wish to speak
  - Post questions on chat window
  - Happy to try other suggestions!
- Lectures recorded and upload on Kaltura
  - Available under “My Media” on Canvas

Overall goals for the course

- Introduce the state-of-the-art in computer vision
- Study principles that make them possible
- Get understanding of tools that drive computer vision
- Enable one or all of several such outcomes
  - Pursue higher studies in computer vision
  - Join industry to do cutting-edge work in computer vision
  - Gain appreciation of modern computer vision technologies
- This is a great time to study computer vision!

Lighting Presentations

- Look out for email with steps for completing the presentation
  - Confirm the paper assignment when you receive the email
  - Send presentation to instructor and TA 3 days before class
  - Include script for narration
  - Incorporate feedback
  - Send final version and recorded video 1 day before class

- Order of presentation: alphabetic
  - Papers assigned by instructor
  - https://docs.google.com/spreadsheets/d/1JcM4V2GaPf7WF2YLFQ70Rn6BaLYEOqEzzj8rXOsb5bw/edit?usp=sharing

Lighting Presentations

- Time limit: 5 minutes ☺
  - **High-quality presentation:** well-practiced and fluent
  - Speak slowly, clearly and explain the content
- General rules
  - **Illustrate:** 1 image per slide, a few bullet points to explain it
  - **Related work:** don't list, rather contrast to 1-2 most important ones
  - **Experiments:** datasets, 1-2 key numbers or graphs
  - **Big no-no's:** giant tables of numbers, hyperparameters
- Template has been provided
  - Try to follow to the extent possible
  - Can replace prompts ("key design element") with relevant content
  - Avoid increasing the number of slides

Papers for Wed, Apr 14

- SuperPoint: Self-Supervised Interest Point Detection and Description
  - https://arxiv.org/abs/1712.07629

- AnchorNet: A Weakly Supervised Network to Learn Geometry-Sensitive Features for Semantic Matching
  - https://arxiv.org/abs/1704.04749

Papers for Fri, Apr 16

- PWC-Net: CNNs for Optical Flow Using Pyramid, Warping and Cost Volume
  - https://arxiv.org/abs/1709.02371

- SelFlow: Self-Supervised Learning of Optical Flow
  - https://arxiv.org/abs/1904.09117

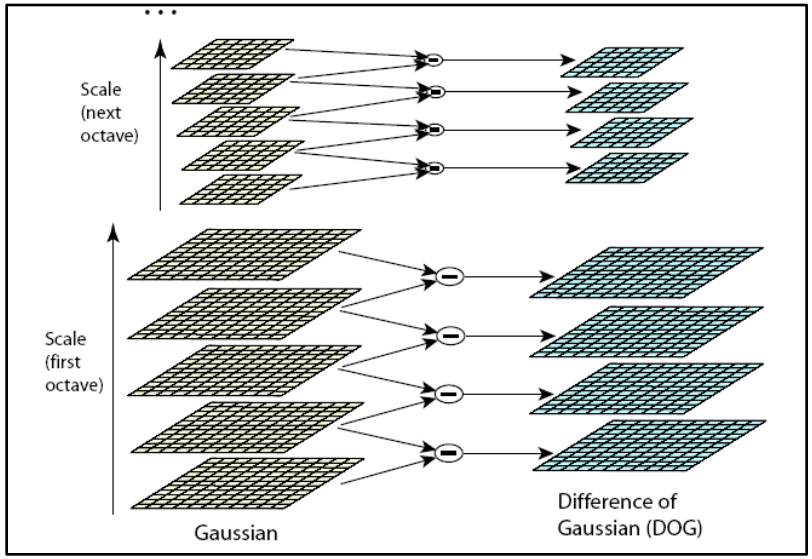
Papers for Wed, Apr 21

- GeoNet: Unsupervised Learning of Dense Depth, Optical Flow and Camera Pose
  - https://arxiv.org/abs/1803.02276

- Unsupervised Scale-Consistent Depth and Ego-Motion Learning from Monocular Video
  - https://arxiv.org/abs/1908.10553

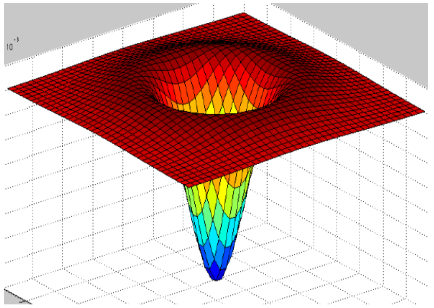
Recap

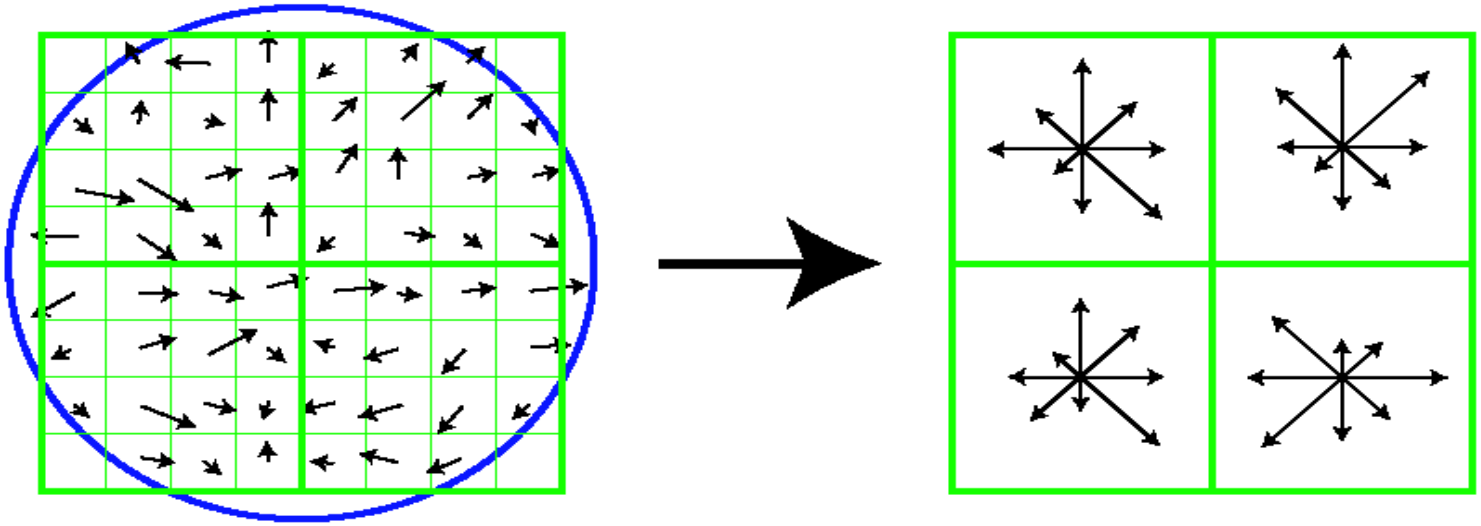
SIFT: scale space, keypoints, descriptors



$$G(x, y, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\frac{x^2+y^2}{\sigma^2}}$$

$$G(x, y, k\sigma) - G(x, y, \sigma) \approx (k-1)\sigma^2 \nabla^2 G.$$



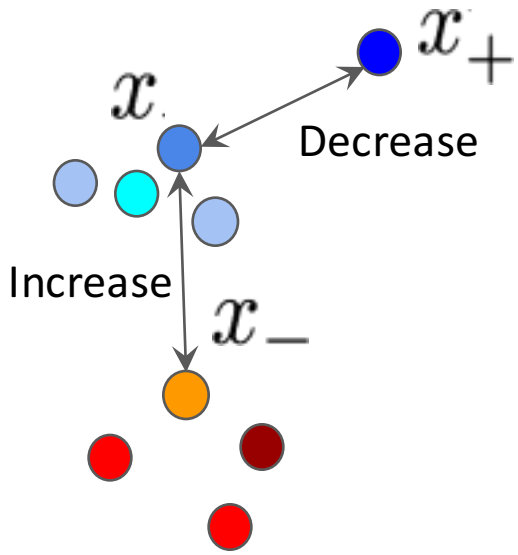


Need for a metric in correspondence learning

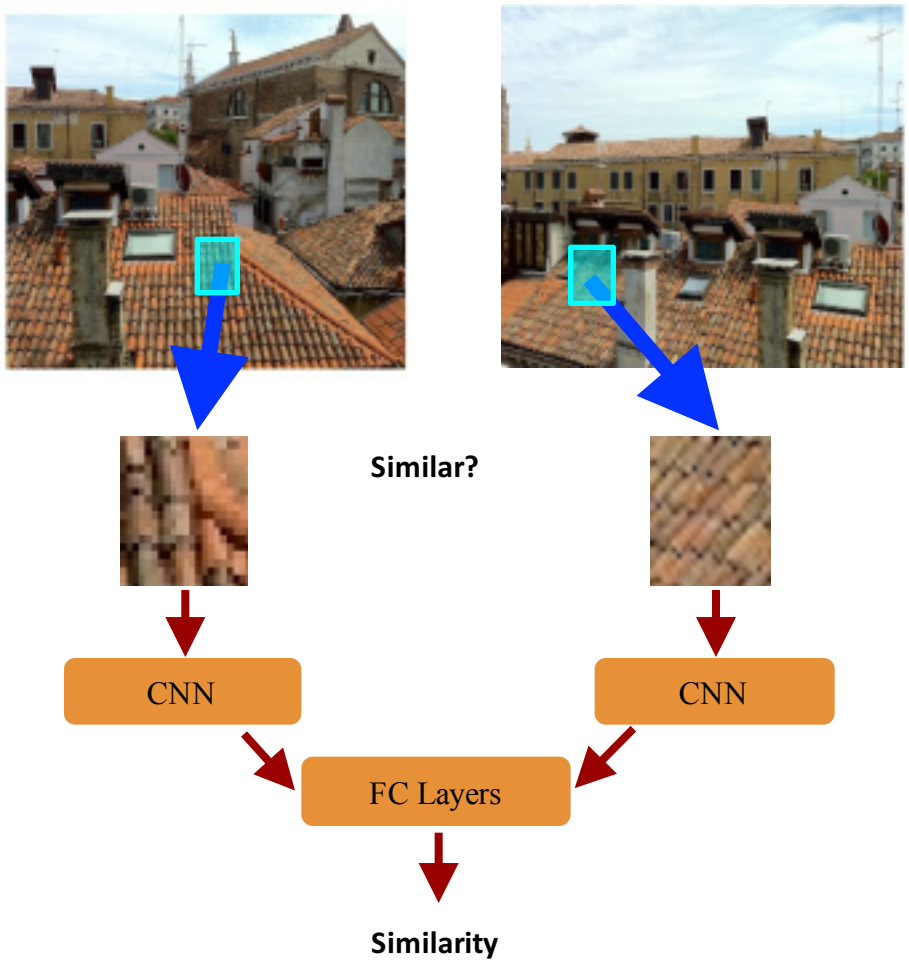
- Learn a feature space directly optimized for correspondence
- Intermediate activations of patch similarity are surrogate features
- Mapping an image to a metric space

$$\|f(x) - f(x_+)\| \to 0$$

$$\|f(x) - f(x_-)\| \geq m$$



Correspondence beyond similarity CNN



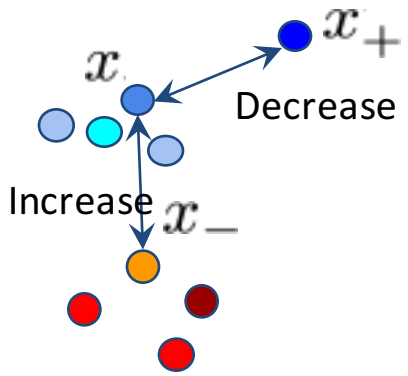
- Detection of interest points
- Normalization of patches
- Multiscale information
- Obtaining training data
- Efficient training and testing

Metric Learning

- Learn a feature space directly optimized for correspondence

$$\|f(x) - f(x_+)\| \to 0$$

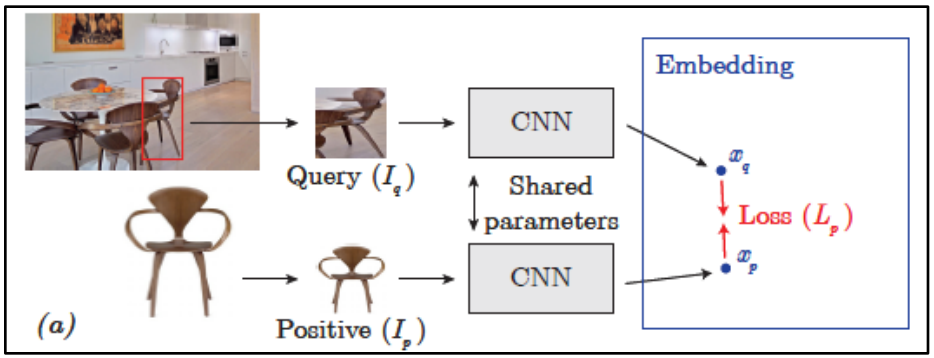
$$\|f(x) - f(x_-)\| \geq m$$

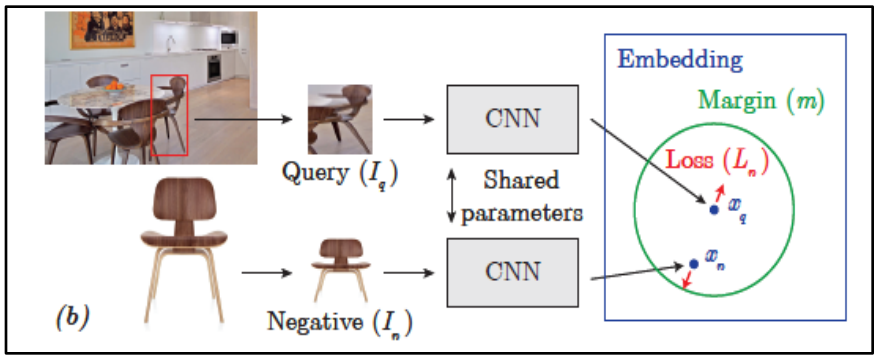


- Contrastive loss: Pair of training examples x_1, x_2 (with labels y_1, y_2)

$$L(x_1, x_2) = s_{12} \, \|x_1 - x_2\|^2 + (1 - s_{12}) \max(0, m^2 - \|x_1 - x_2\|^2)$$

where, $s_{12} = 1$ when $y_1 = y_2$, otherwise $s_{12} = 0$.





Triplet Loss for Metric Learning

- At each training iteration, sample a set of triplets

$$\mathcal{T} = (\mathbf{x}_a, \mathbf{x}_p, \mathbf{x}_n) \qquad y_a = y_p \neq y_n$$

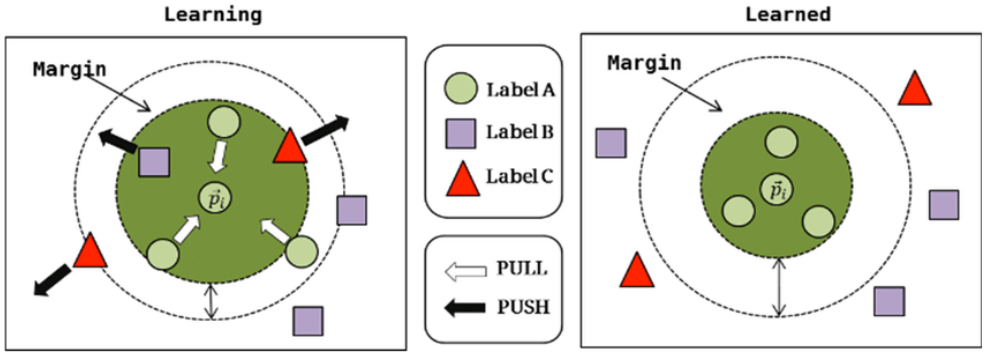
Anchor Positive Negative

- Goal: push negative a margin further from anchor than positive

$$\|\mathbf{x}_a - \mathbf{x}_p\|^2 + m \leq \|\mathbf{x}_a - \mathbf{x}_n\|^2$$

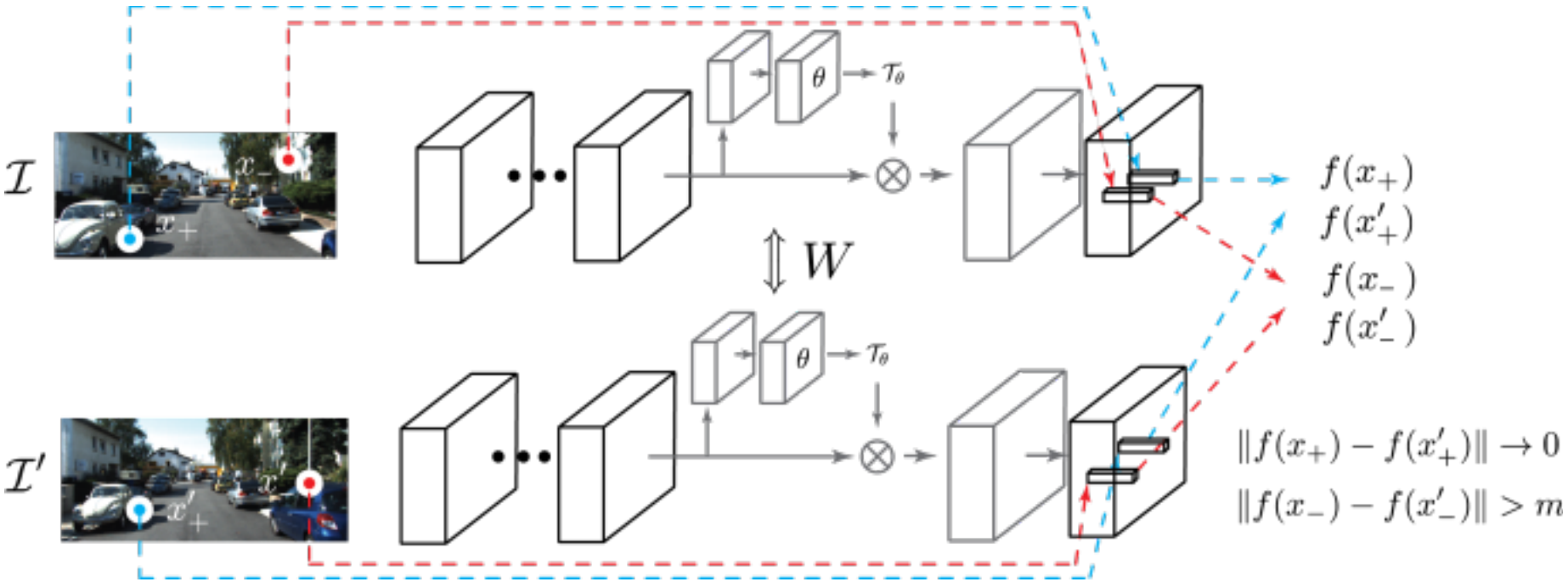
- Triplet loss:

$$l_{tri}(\mathcal{T}) = \left[\|\mathbf{x}_a - \mathbf{x}_p\|^2 - \|\mathbf{x}_a - \mathbf{x}_n\|^2 + m\right]_+$$

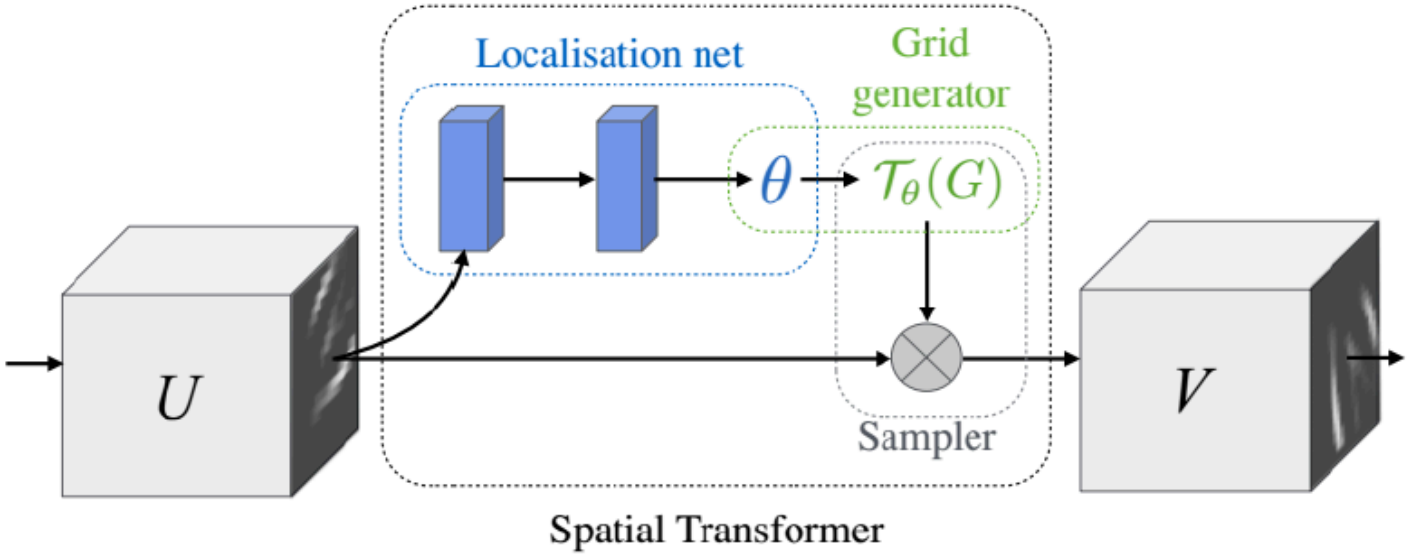


UCN Architecture

- Extract convolutional features over a pair of images
- Ground truth positives available as matching points
- Points further away are all negatives
- Train with metric learning to learn a feature with meaningful distances



Spatial Transformers

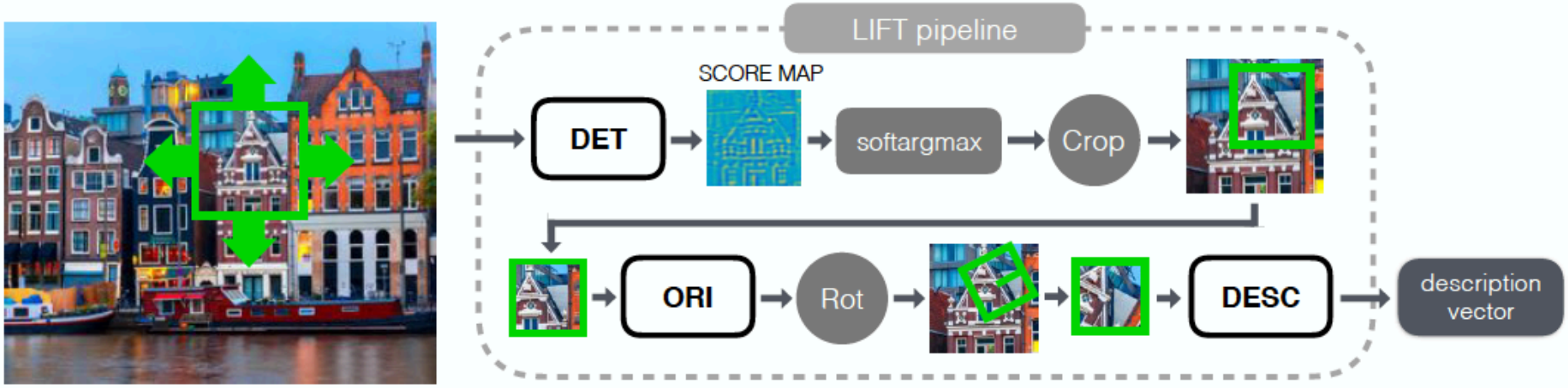


- **Localization net** takes input feature map U and outputs transformation parameters

- **Grid generator** takes uniform grid in output map and deforms it by transformation
- This determines a deformed grid to sample the input feature map U

- **Sampler** takes U and deformed grid as input, then produces V by sampling

Keypoint detection and description



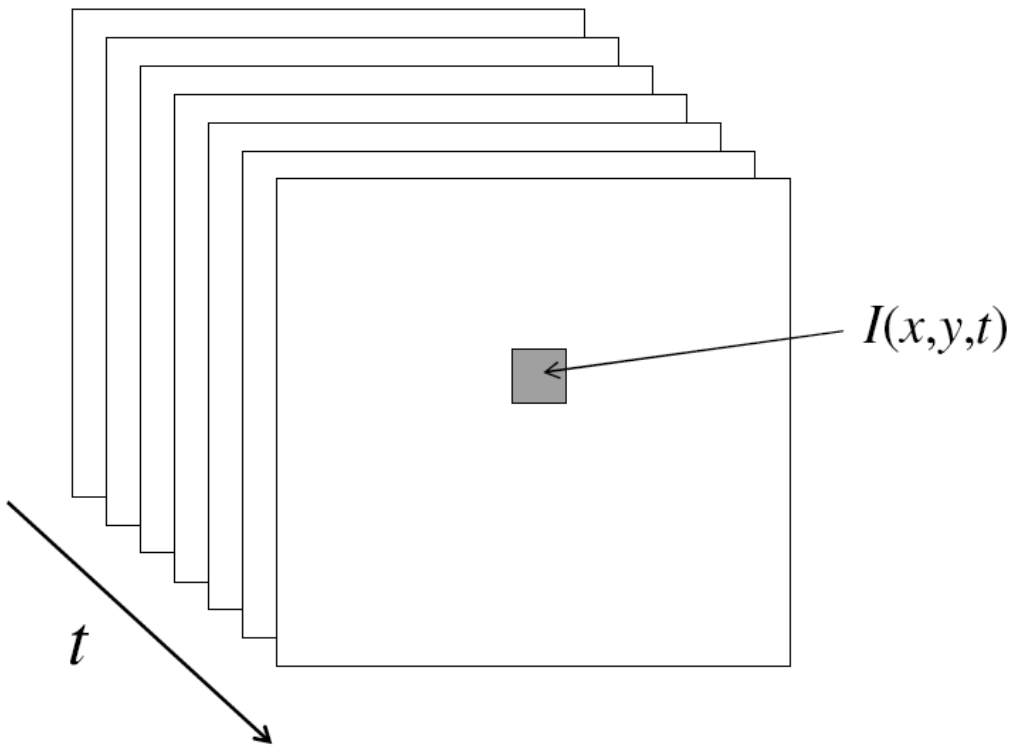
- Start with a big patch
- Compute score map for possible keypoint locations
- Crop small patch around most likely keypoint location
- Normalize cropped patch orientation
- Compute descriptor for the normalized patch

[LIFT: Learned Invariant Feature Transform, ECCV 2016]

Optical Flow

Video

- A video is a sequence of frames captured over time
- Now our image data is a function of space (x, y) and time (t)



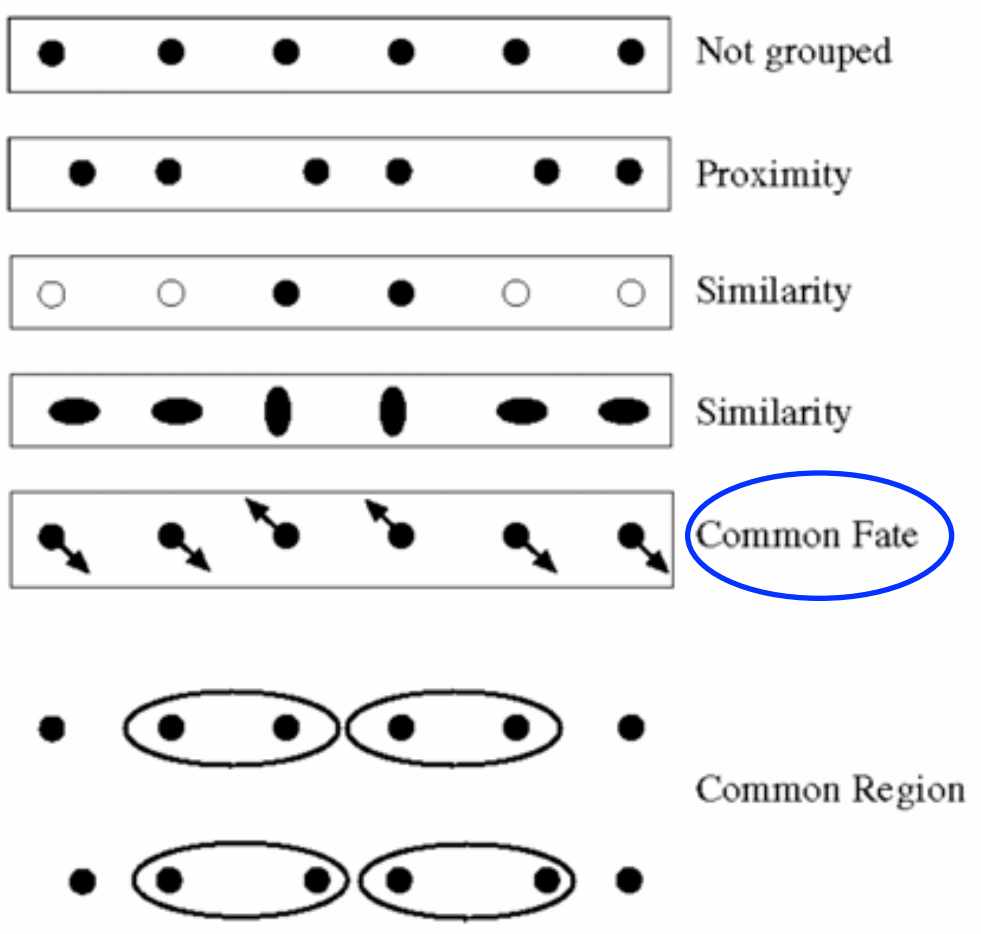
Motion and perceptual organization

- Even “impoverished” motion data can evoke a strong percept

Motion and perceptual organization

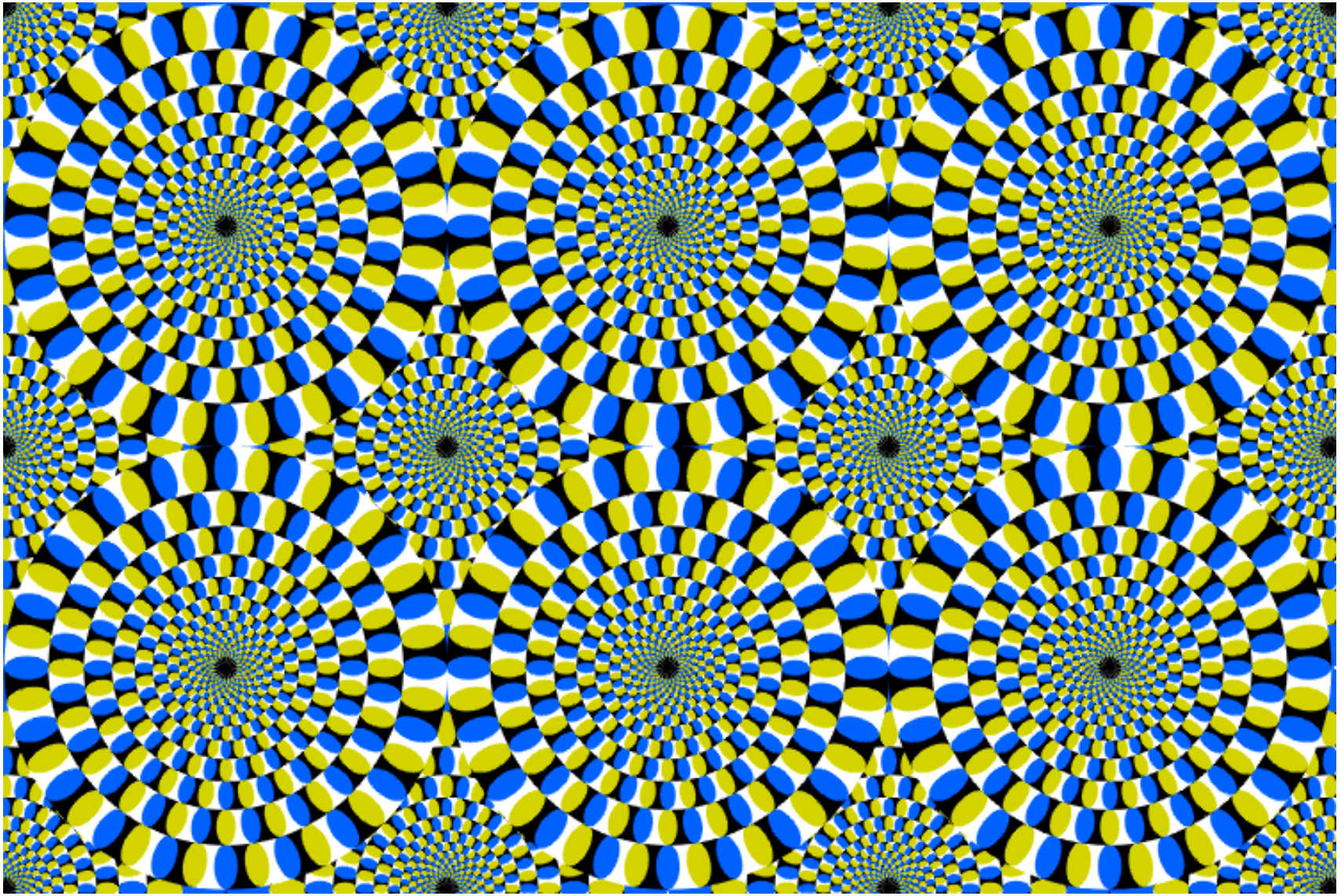
- Even “impoverished” motion data can evoke a strong percept

Motion and perceptual organization

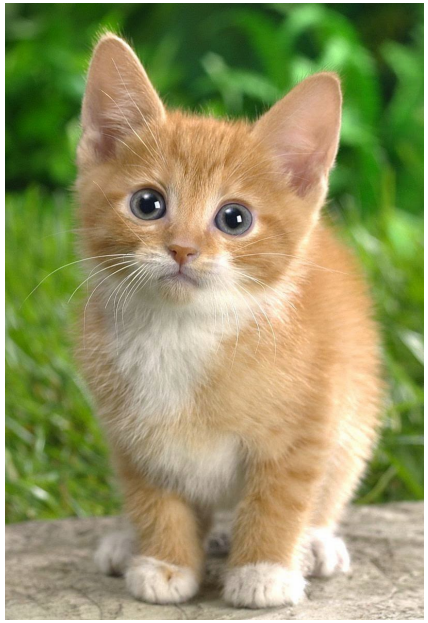


Gestalt psychology (Max Wertheimer, 1880-1943)

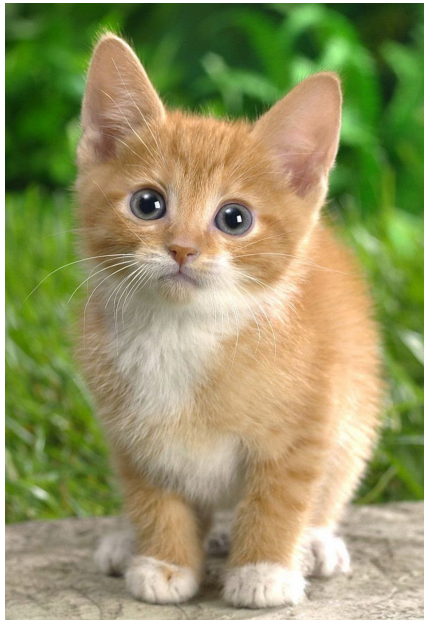
Peripheral drift illusion

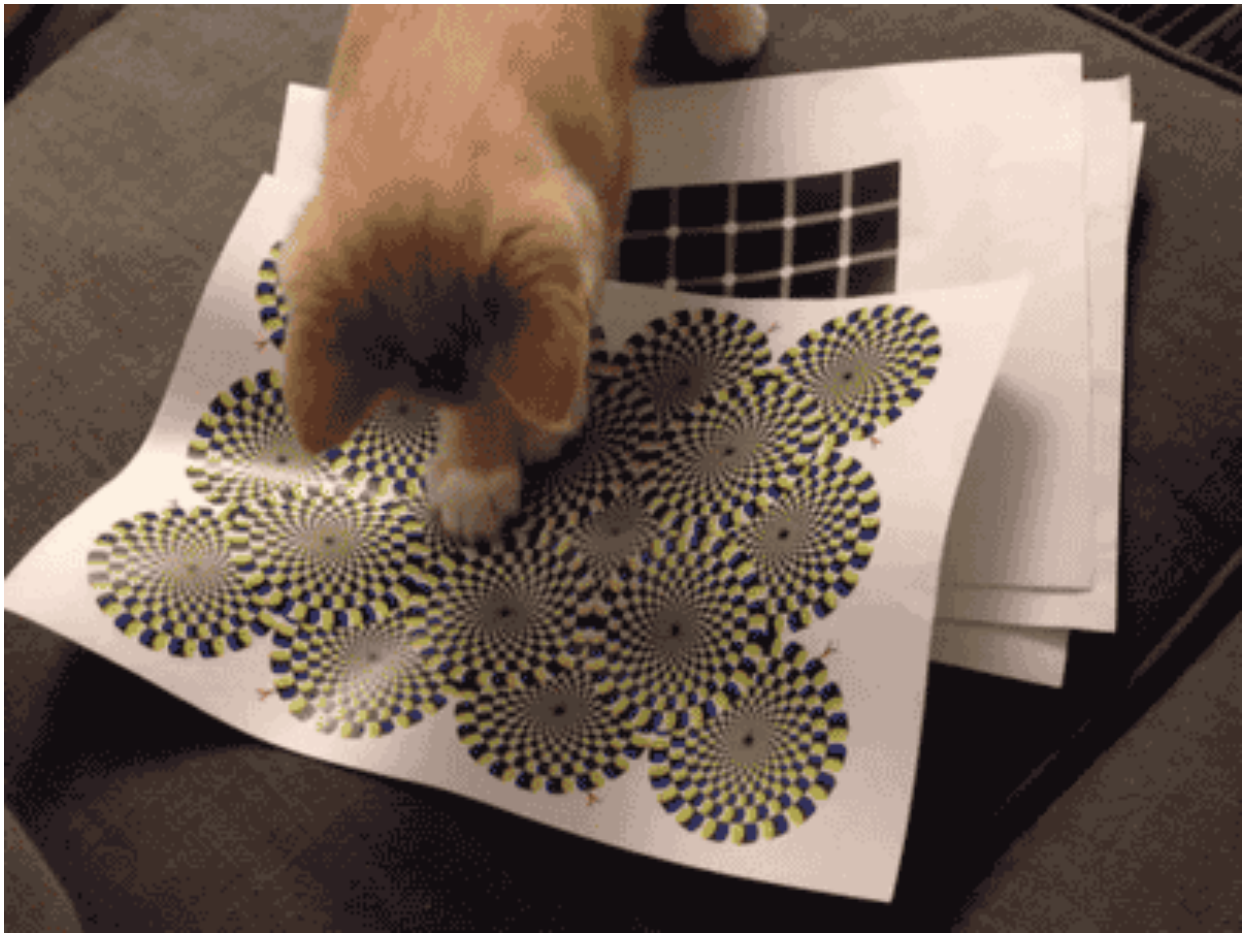


Does it work on cats?

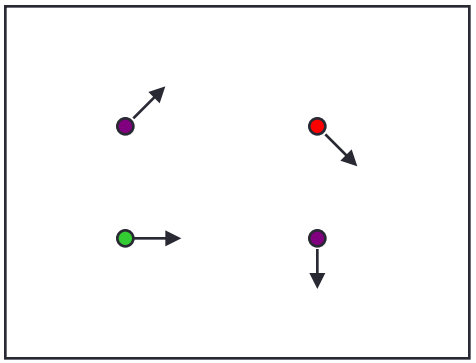


Does it work on cats?

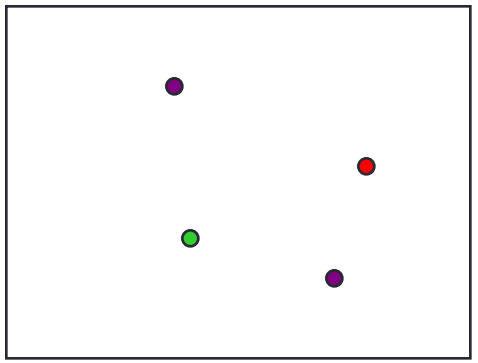




Problem definition: optical flow



$I(x, y, t)$



$I(x, y, t+1)$

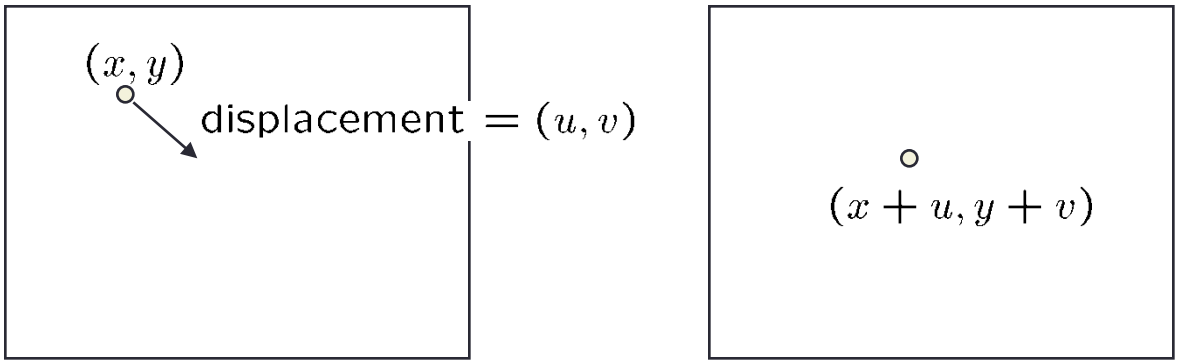
How to estimate pixel motion from image *I(x,y,t)* to *I(x,y,t+1)* ?

- Solve pixel correspondence problem
- Given a pixel in I(x,y,t), look for nearby pixels of the same color in *I(x,y,t+1)*

Key assumptions

- Small motion: Points do not move very far
- Color constancy: A point in *I(x,y,t)* looks the same in *I(x,y,t+1)*
- For grayscale images, this is brightness constancy

Optical flow constraints (grayscale images)



- Let's look at these constraints more closely
  - Brightness constancy constraint (equation)

$$I(x, y, t) = I(x+u, y+v, t+1)$$

  - Small motion: Taylor series expansion of *I*:

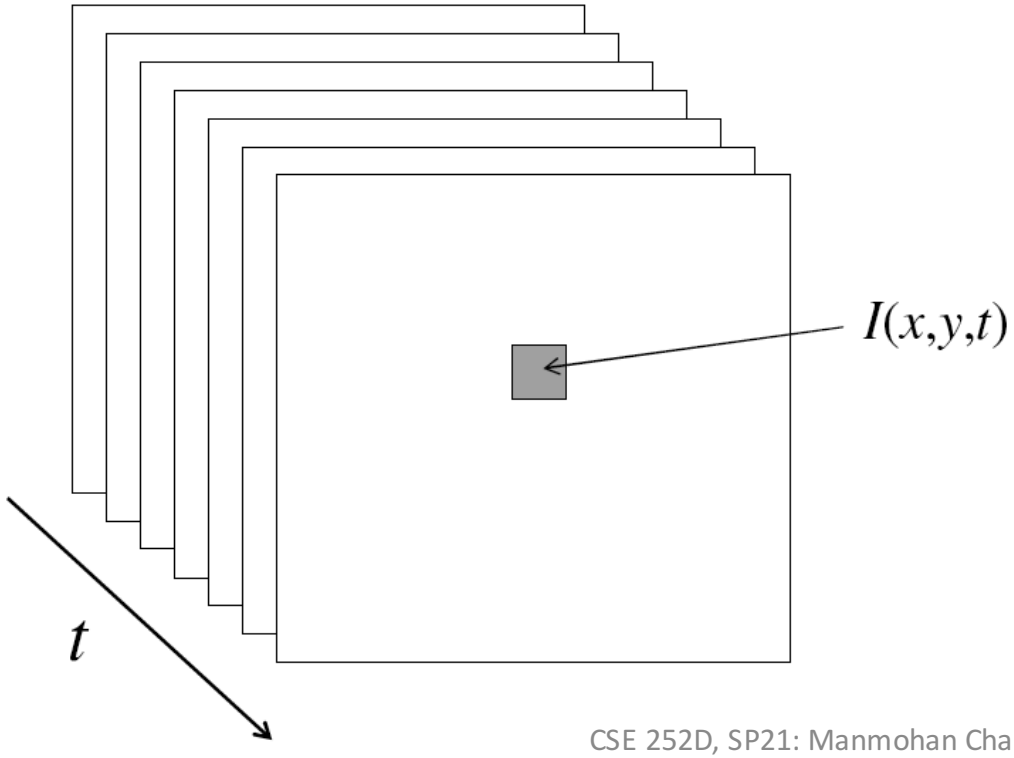
$$I(x+u, y+v) = I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \text{ higher order terms}$$

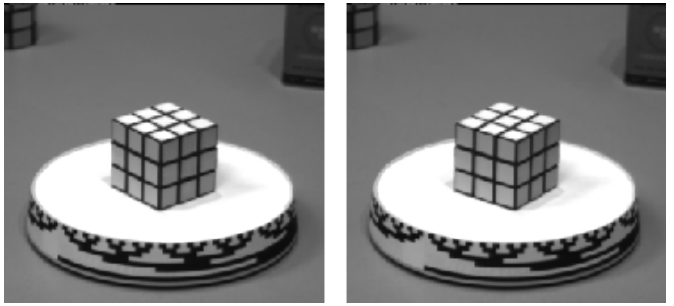
$$\approx I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v$$

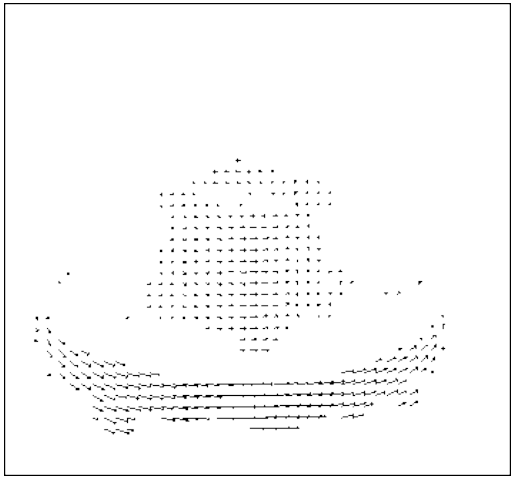
Optical flow

Brightness constancy constraint equation

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







Brightness constancy

Brightness constancy constraint equation

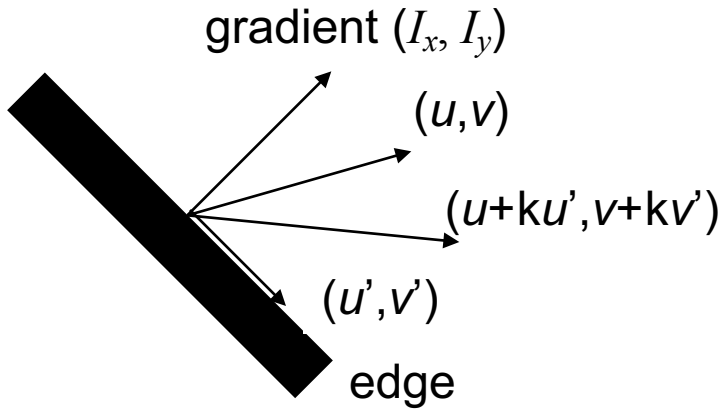
$$I_x\, u + I_y\, v + I_t = 0$$

- How many equations and unknowns per pixel?
  - One equation two unknowns (u, v)

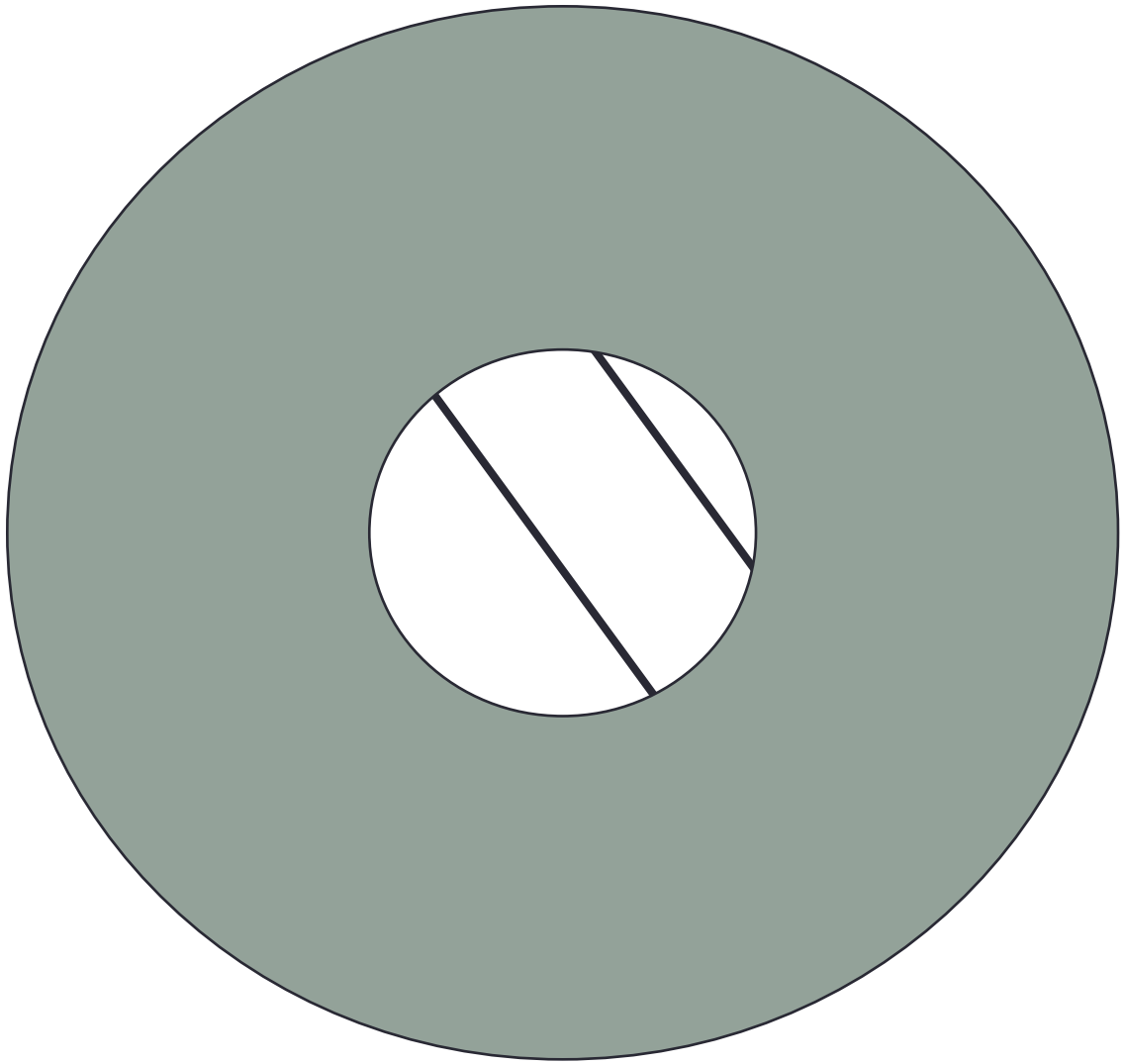
The component of the motion perpendicular to the gradient (that is, parallel to the edge) cannot be measured

If (*u*, *v*) satisfies the flow equation, so does (*u*+*ku*', *v*+*kv*') for any *k* if

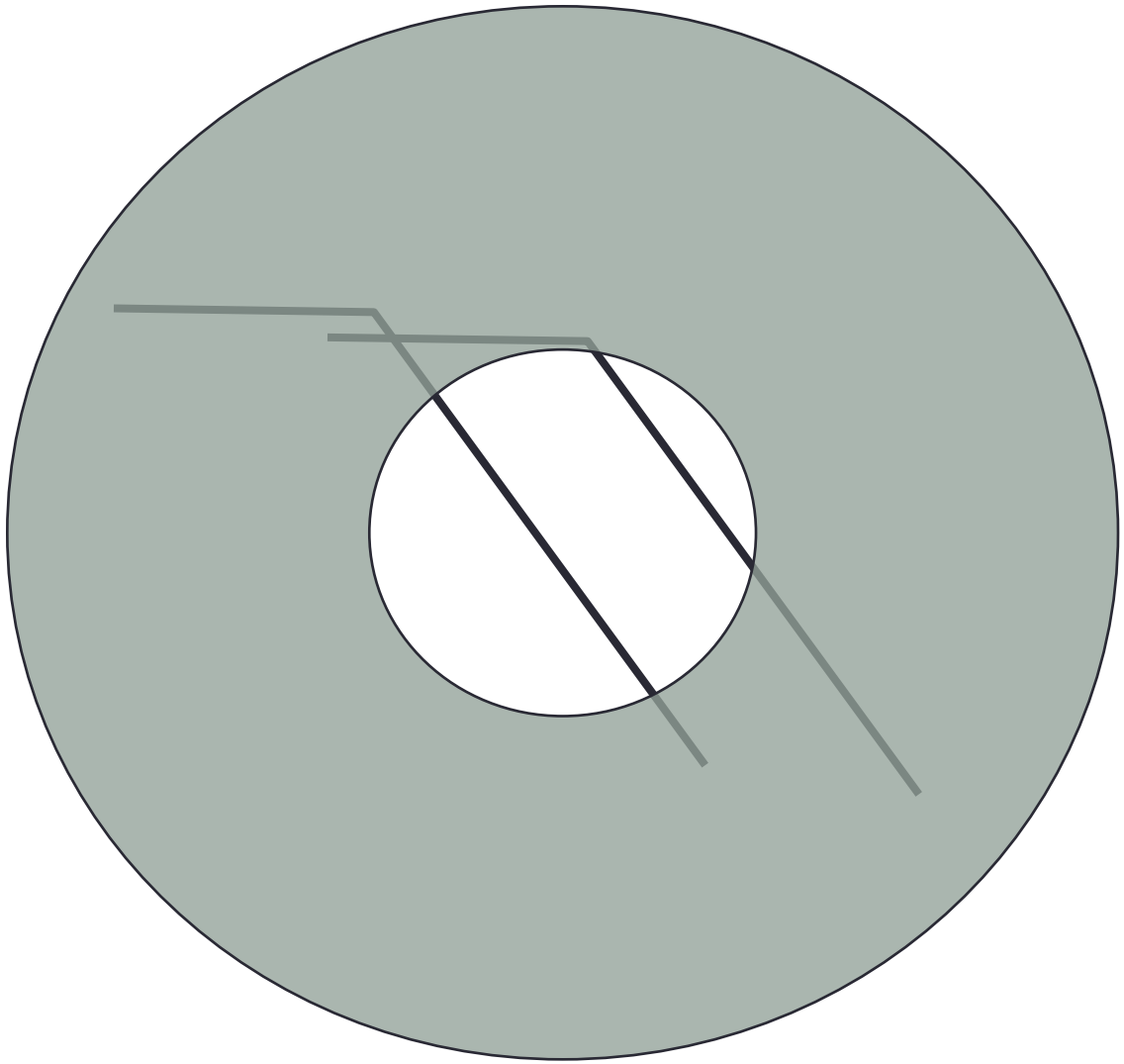
$$I_x u' + I_y v' = 0$$



Aperture problem

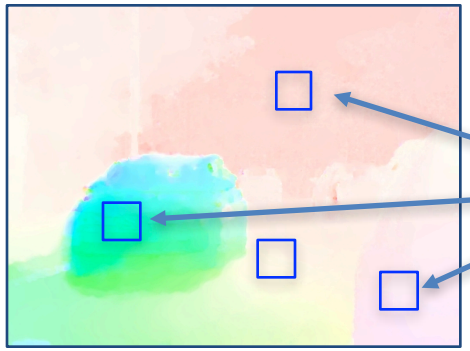


Aperture problem



Optical flow: Lucas-Kanade

- Overconstrained linear system through patch coherence



Assume a constant flow in K x K patch

$$\begin{bmatrix} I_x(\mathbf{p}_1) & I_y(\mathbf{p}_1) \\ I_x(\mathbf{p}_2) & I_y(\mathbf{p}_2) \\ \vdots & \vdots \\ I_x(\mathbf{p}_{25}) & I_y(\mathbf{p}_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{p}_1) \\ I_t(\mathbf{p}_2) \\ \vdots \\ I_t(\mathbf{p}_{25}) \end{bmatrix}$$

$$\underset{25\text{x}2}{A} \quad \underset{2\text{x}1}{d} = \underset{25\text{x}1}{b}$$

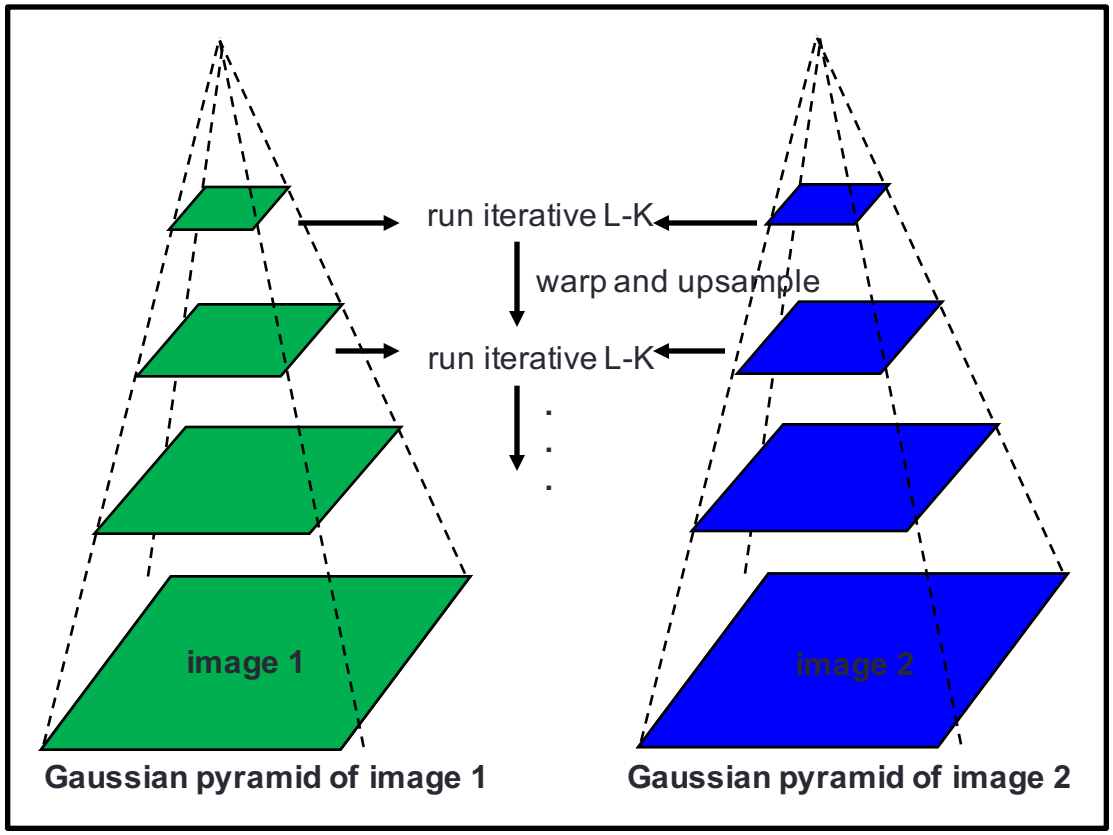
Least squares solution for *d* given by $(A^T A)\ d = A^T b$

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

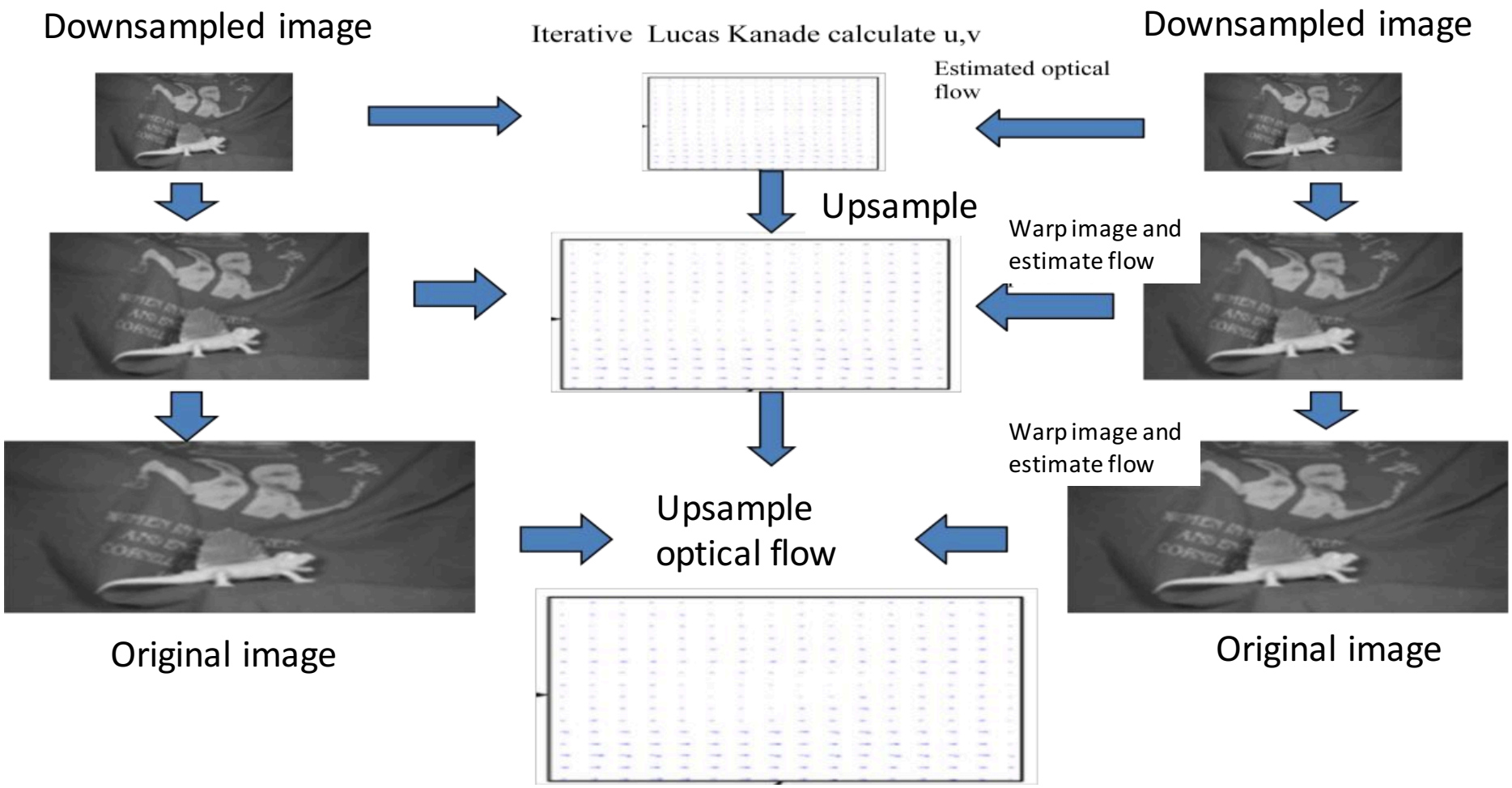
$A^T A$ $\qquad\qquad$ $A^T b$

The summations are over all pixels in the K x K window

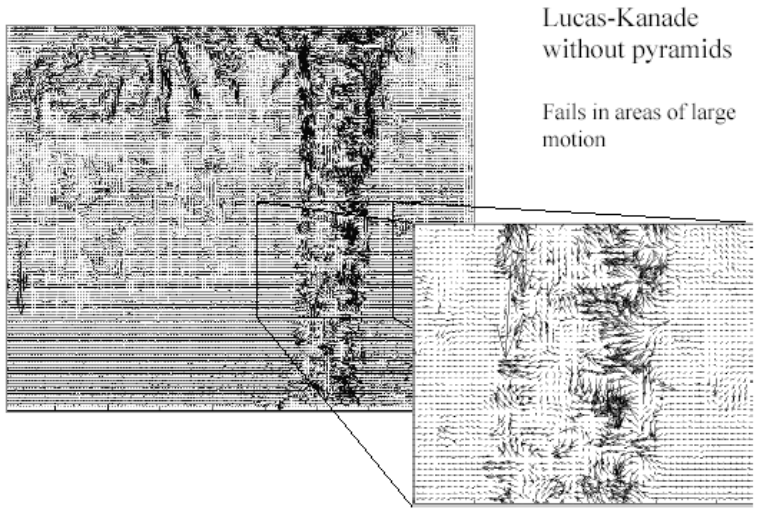
Optical flow: Coarse-to-Fine

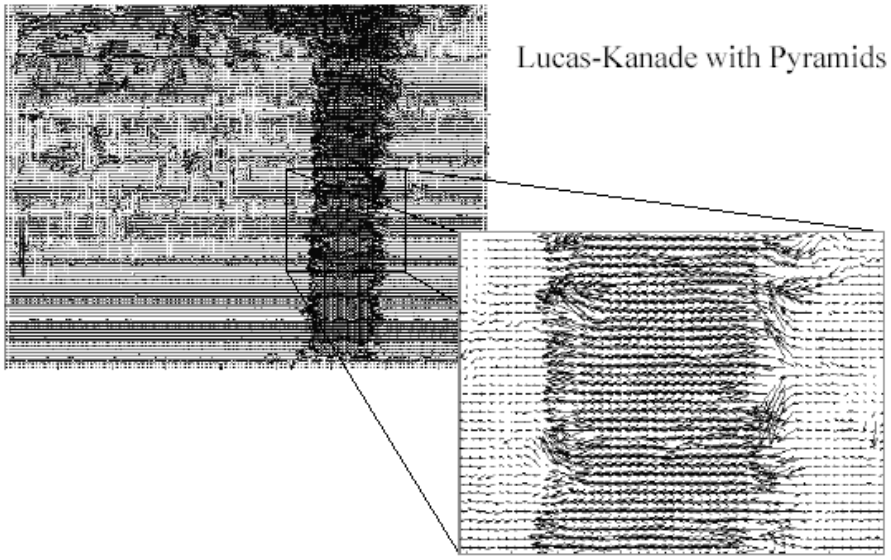


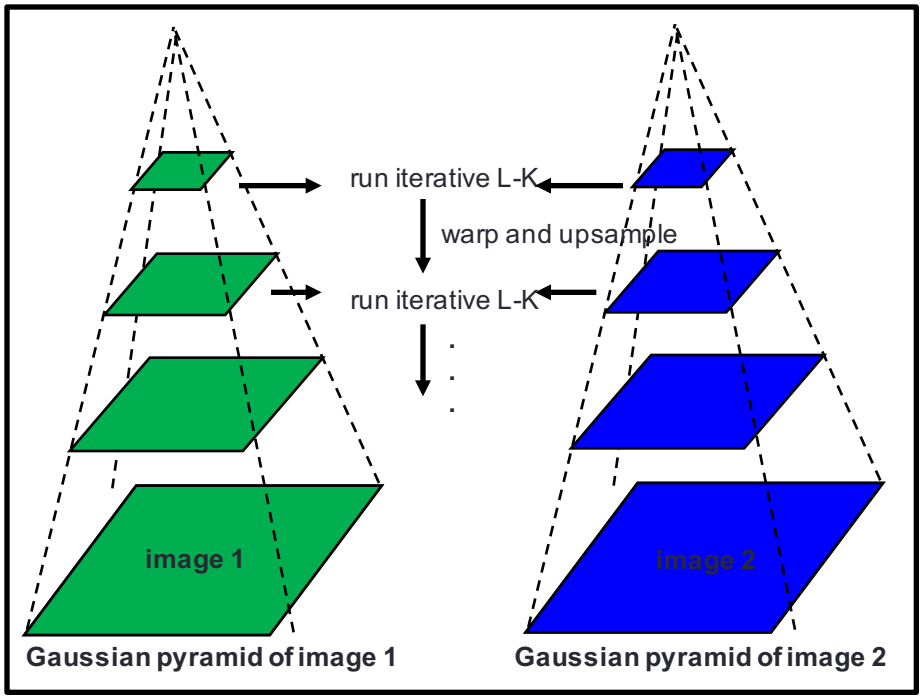
Optical flow: Coarse-to-Fine



Optical flow: Coarse-to-Fine



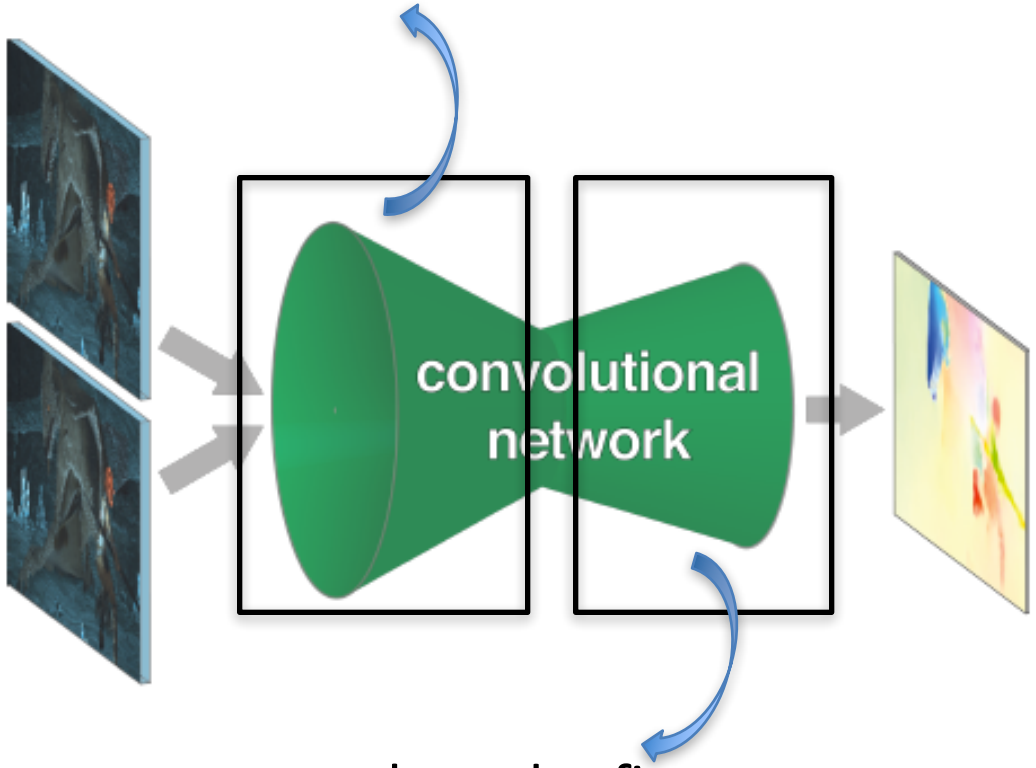




* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

FlowNet

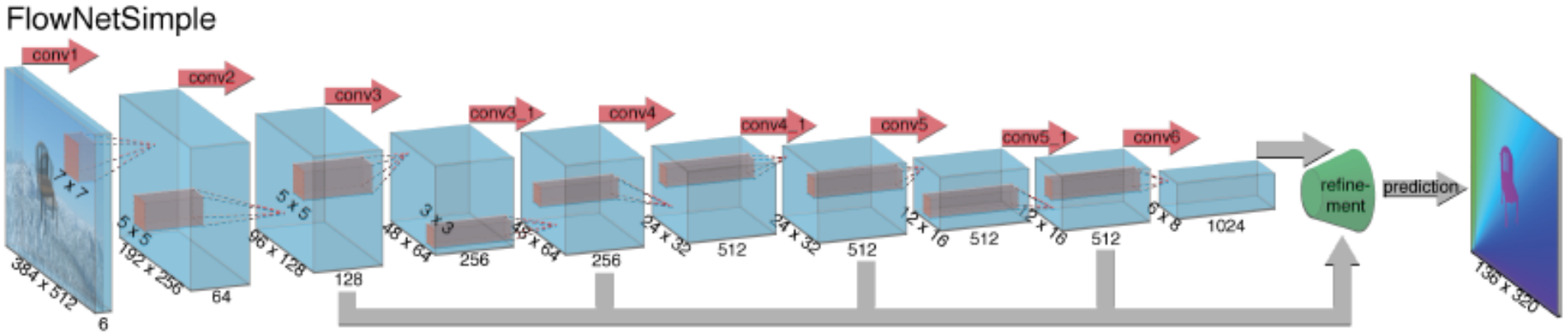
- Contracting part: extract a rich hierarchical feature representation
  - Flow is local entity, need for abstract features?



- Expanding part: upsample and refine
  - Flow is at image resolution: progressively upsample feature maps

[FlowNet: Learning Optical Flow with Convolutional Networks, Fischer at al., ICCV 2015]

FlowNetSimple

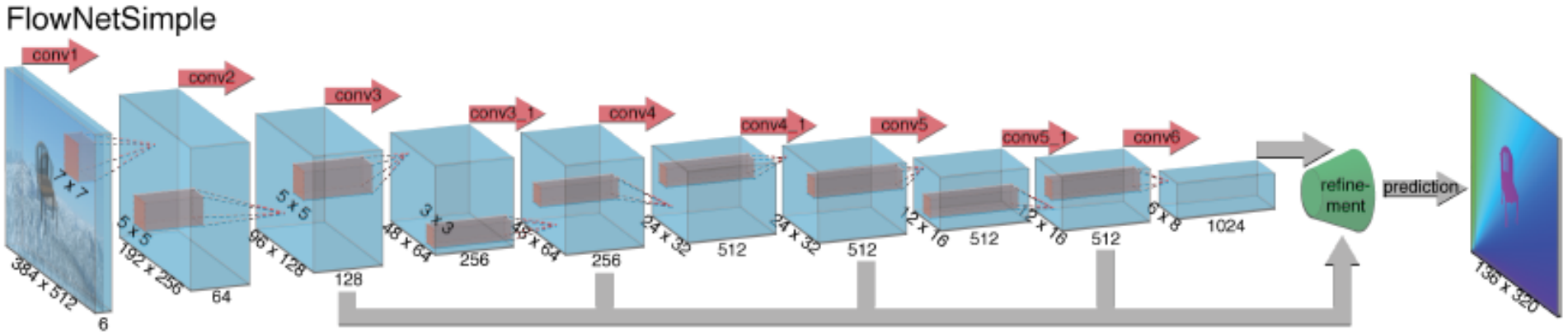


- Concatenate two images, feed through a generic CNN
- Let network figure out how to map image pair to optical flow

- Advantage:

- Disadvantage:

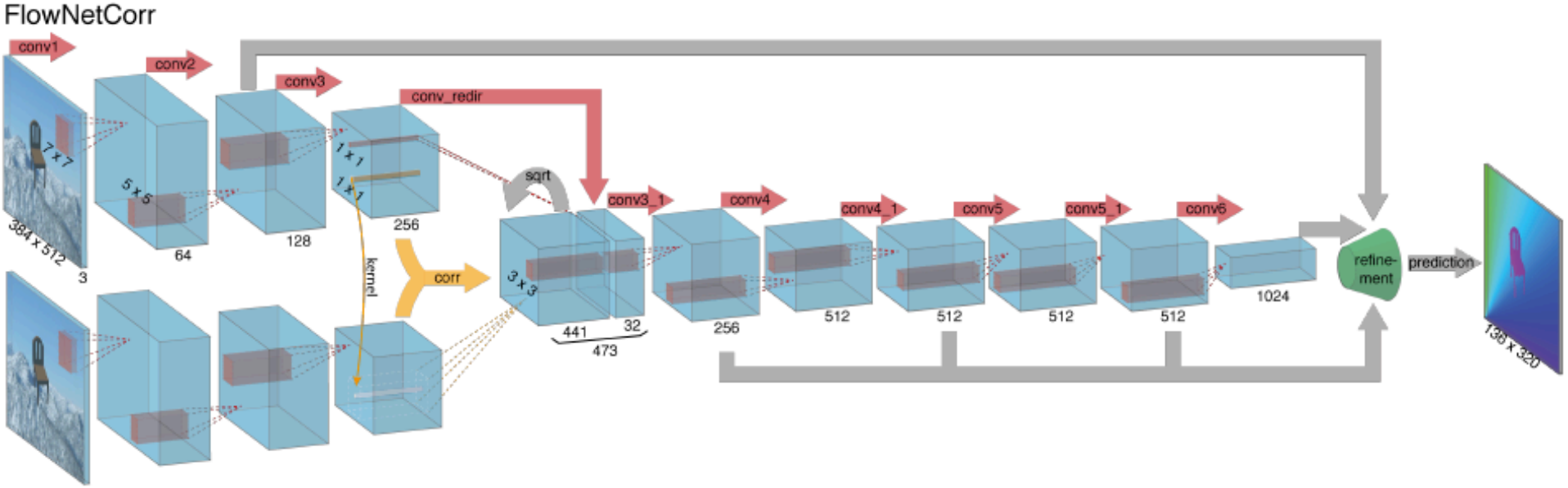
FlowNetSimple



- Concatenate two images, feed through a generic CNN
- Let network figure out how to map image pair to optical flow

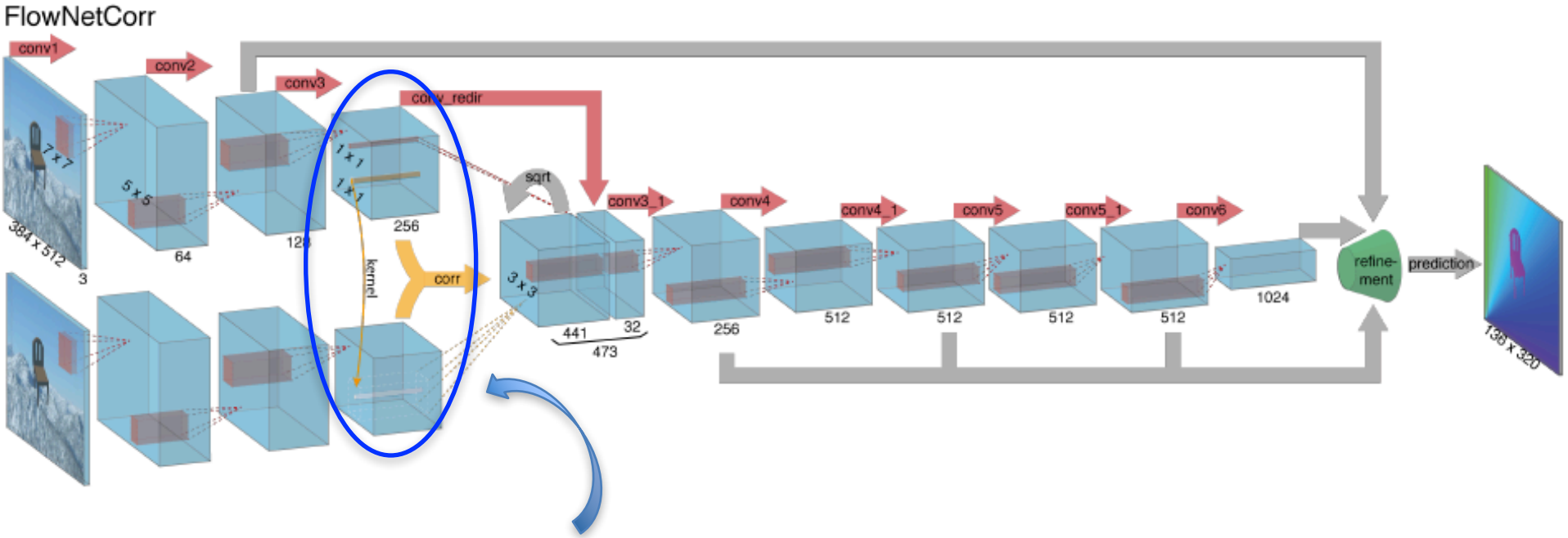
- Advantage:
  - Simple!
- Disadvantage:
  - Unclear how well local SGD-based optimization can do

FlowNetCorr



- Design architecture to encourage spatial matching
  - Meaningful image representation, then match at higher feature layer
- Correlation layer
  - Perform a multiplicative patch comparison

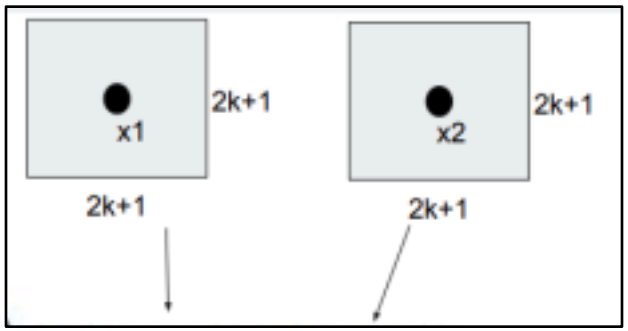
FlowNetCorr: Correlation Layer



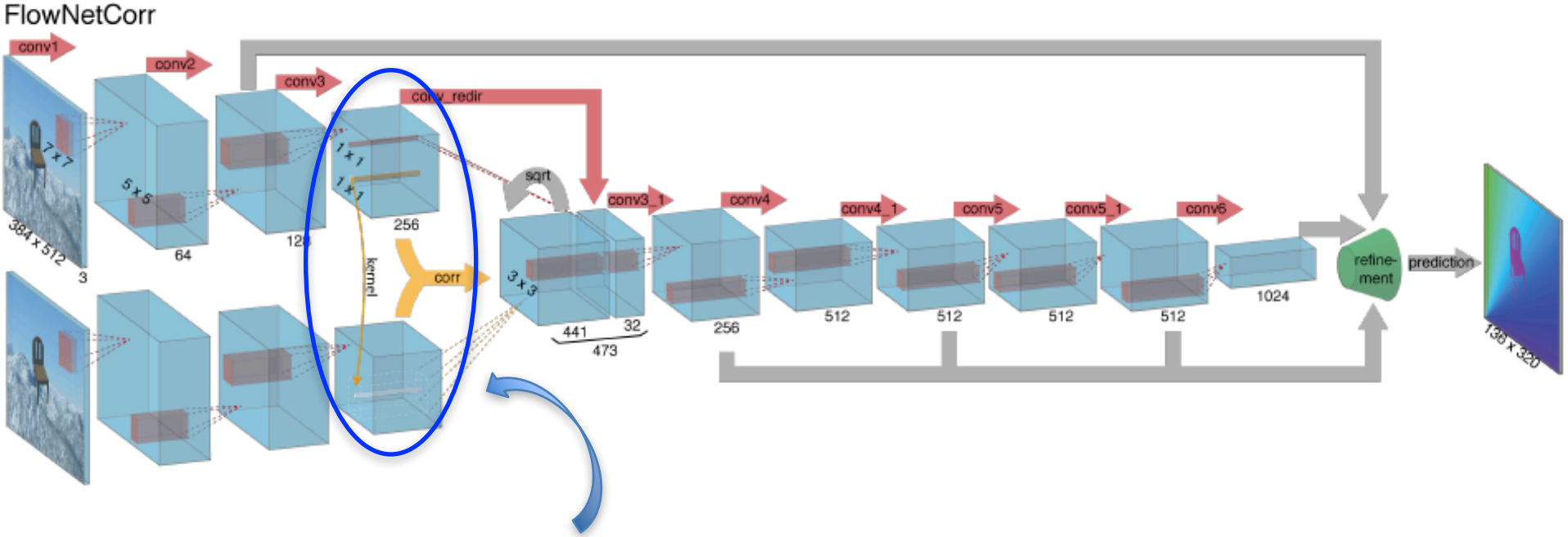
- Multi-channel feature maps $\mathbf{f}_1$ and $\mathbf{f}_2$, of size *w* x *h* and *n* channels
- Correlation of patches centered at $\mathbf{x}_1$ and $\mathbf{x}_2$:

$$c(\mathbf{x}_1, \mathbf{x}_2) = \sum_{\mathbf{o} \in [-k,k] \times [-k,k]} \langle \mathbf{f}_1(\mathbf{x}_1 + \mathbf{o}), \mathbf{f}_2(\mathbf{x}_2 + \mathbf{o}) \rangle$$

- Number of trainable parameters?
- Cost of computing correlations?



FlowNetCorr: Correlation Layer

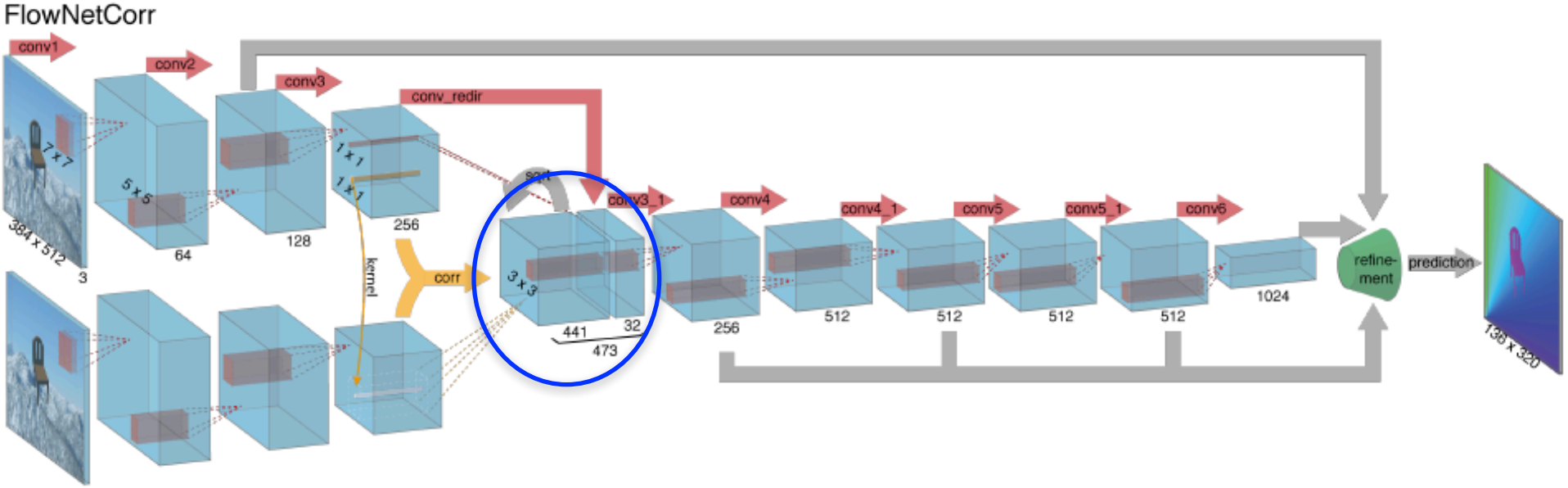


- Multi-channel feature maps $\mathbf{f}_1$ and $\mathbf{f}_2$, of size w x h and n channels
- Correlation of patches centered at $\mathbf{x}_1$ and $\mathbf{x}_2$:

$$c(\mathbf{x}_1, \mathbf{x}_2) = \sum_{\mathbf{o} \in [-k,k] \times [-k,k]} \langle \mathbf{f}_1(\mathbf{x}_1 + \mathbf{o}), \mathbf{f}_2(\mathbf{x}_2 + \mathbf{o}) \rangle$$

- Number of trainable parameters? None.
- Cost of computing correlations? $(2k+1)^2$ x $(w$ x $h)^2$ x n.

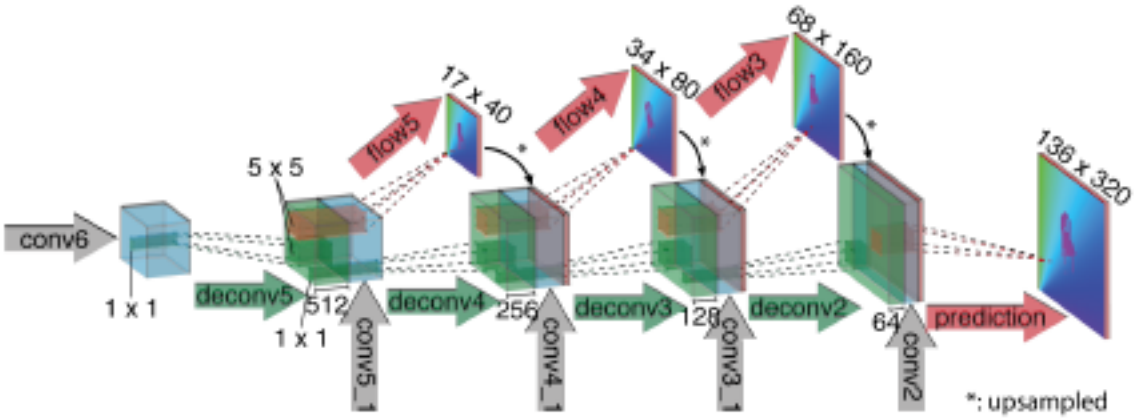
FlowNetCorr: Correlation Layer



- Efficiency considerations:
  - Limit $\mathbf{x}_2$ to lie within maximum disparity of d pixels from $\mathbf{x}_1$
  - Stride on $\mathbf{f}_1$ to limit number of pixels where correlation is computed

- Stack each of $(2d+1)^2$ "disparity" maps as output channels
- Also stack 1 x 1 output of input feature to retain image information

FlowNetCorr: Refinement

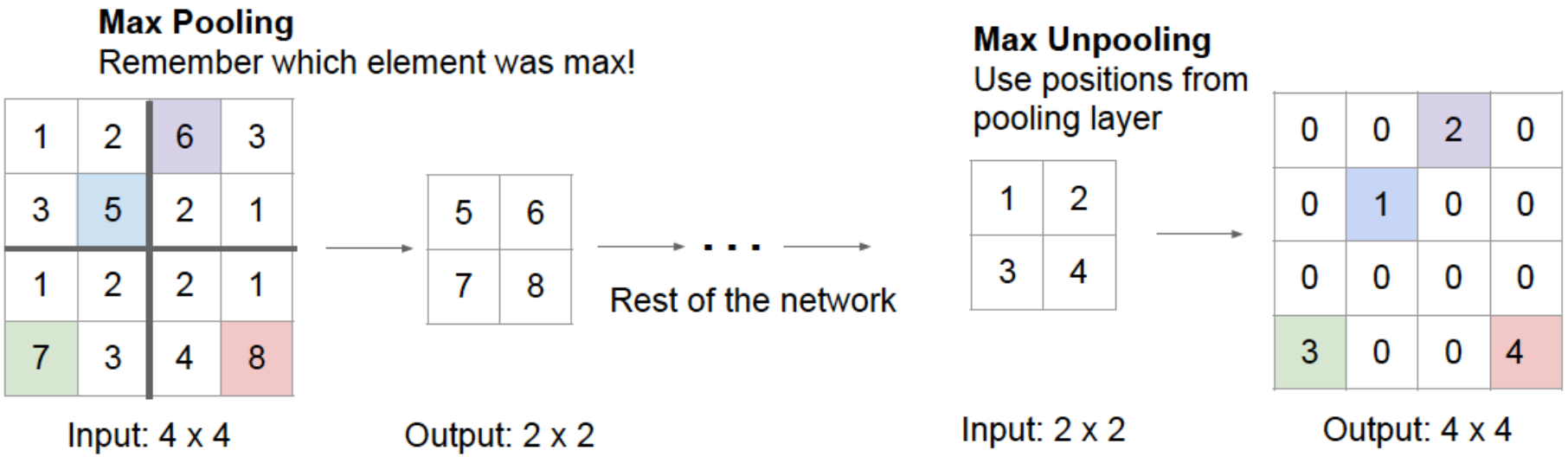


- Gradually upsample the low-dimensional feature.
- Concatenate with encoder feature of corresponding scale, to recover details.

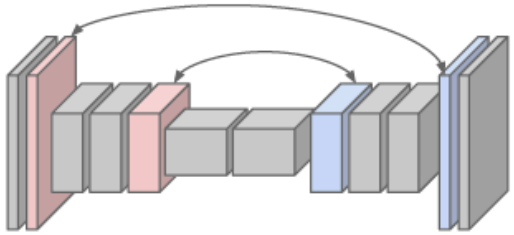
- In simplest form, upsampling can be implemented as bilinear interpolation.

- Can be learned as unpooling followed by convolution
- Can be learned as a transposed convolution filter

Refinement: Unpooling followed by convolution



Corresponding pairs of downsampling and upsampling layers



[Figure from: L. Fei-Fei et al., CS 231n]

Normal Convolutions

1	2
3	4

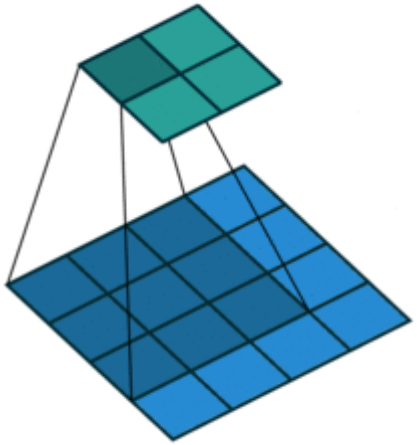
$\star$

1	2	3
4	5	6
7	8	9

$=$

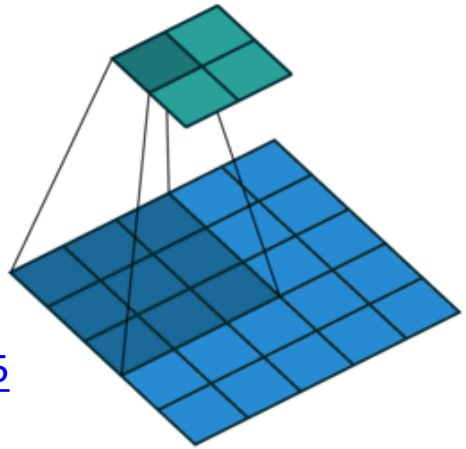
Filter, a

Input, x



https://arxiv.org/abs/1603.07285

[Dumolin and Visin, 2018]



Normal Convolutions

1	2
3	4

$\star$

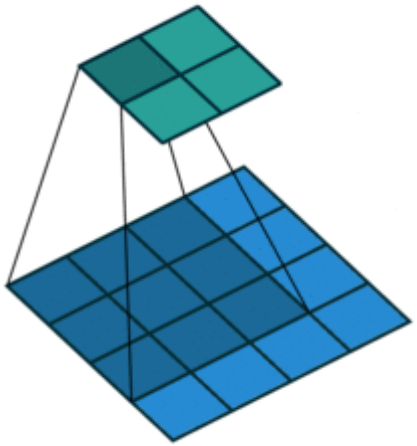
1	2	3
4	5	6
7	8	9

$=$

37	47
67	77

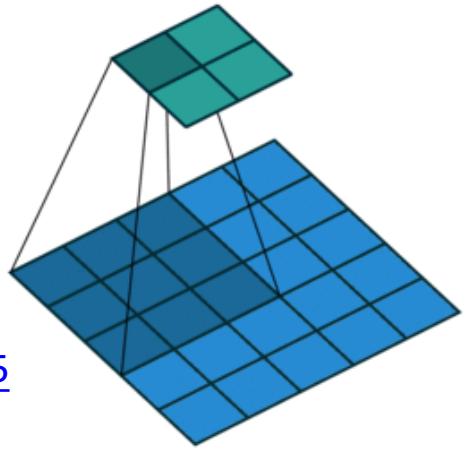
Filter, a

Input, x



https://arxiv.org/abs/1603.07285

[Dumolin and Visin, 2018]



Normal Convolutions

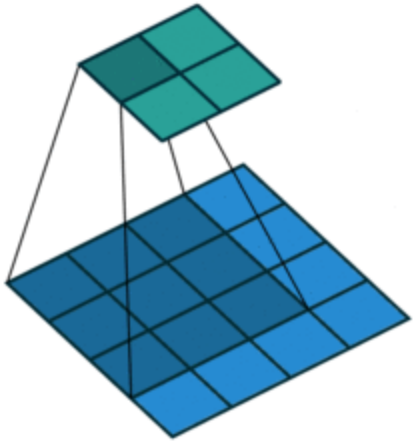
1	2
3	4

☆

1	2	3
4	5	6
7	8	9

=

37	47
67	77



Filter, a

Input, x

1	2	0	3	4	0	0	0	0
0	1	2	0	3	4	0	0	0
0	0	0	1	2	0	3	4	0
0	0	0	0	1	2	0	3	4

A

1
2
3
4
5
6
7
8
9

X

=

37
47
67
77

Transposed Convolutions

1	2
3	4

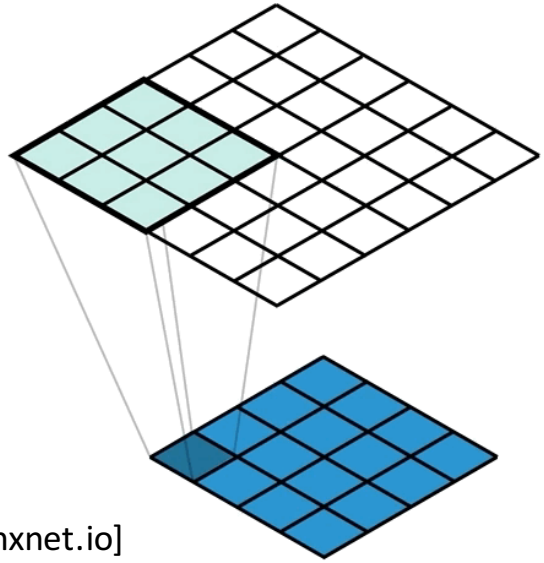
Filter, a

$\overset{T}{\star}$

1	0
2	1

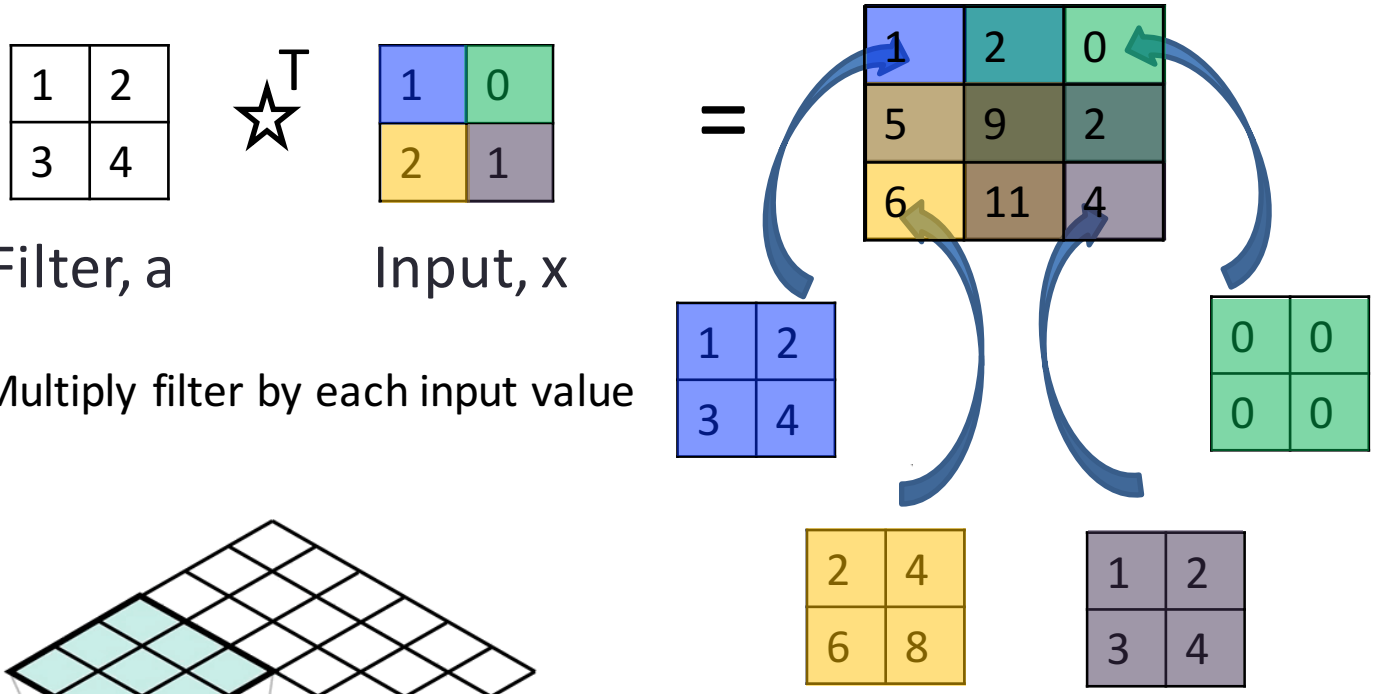
Input, x

$=$



[Thom Lane, mxnet.io]

Transposed Convolutions

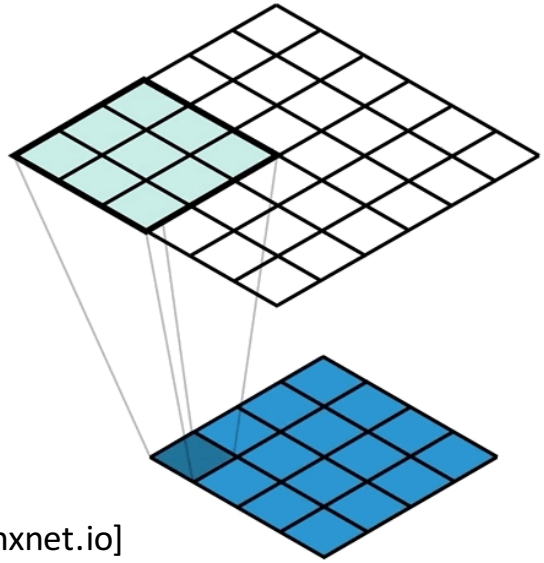


Filter, a

Input, x

Multiply filter by each input value

Tile the scaled filter at output locations
Add overlapping values



[Thom Lane, mxnet.io]

Transposed Convolutions

1	2
3	4

Filter, a

$\overset{T}{\star}$

1	0
2	1

Input, x

$=$

1	2	0
5	9	2
6	11	4

1	0	0	0
2	1	0	0
0	2	0	0
3	0	1	0
4	3	2	1
0	4	0	2
0	0	3	0
0	0	4	3
0	0	0	4

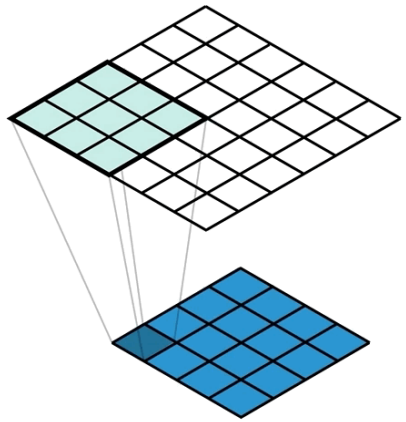
A^T

X

1
0
2
1

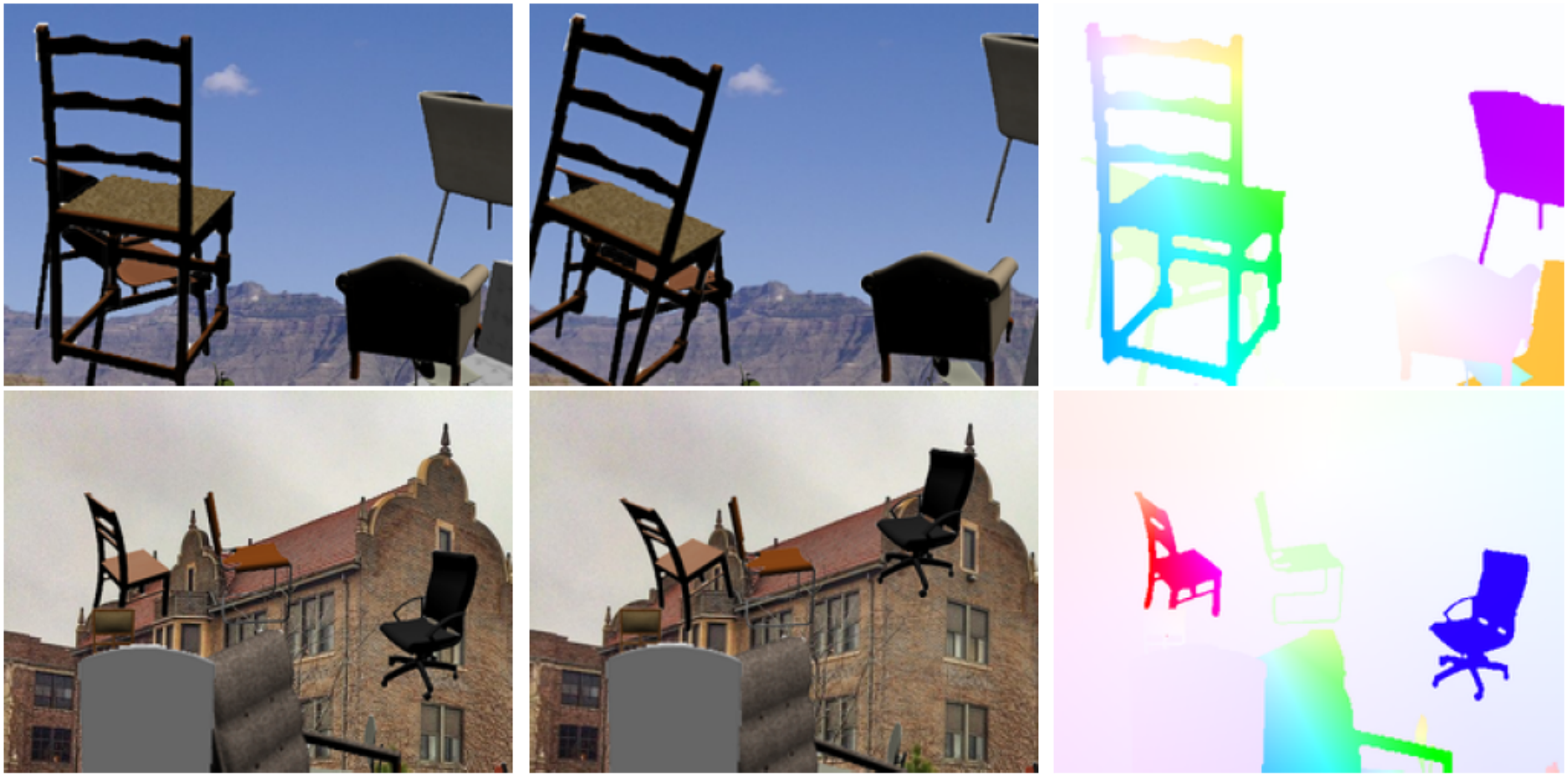
$=$

1
2
0
5
9
2
6
11
4



Synthetic Training Data

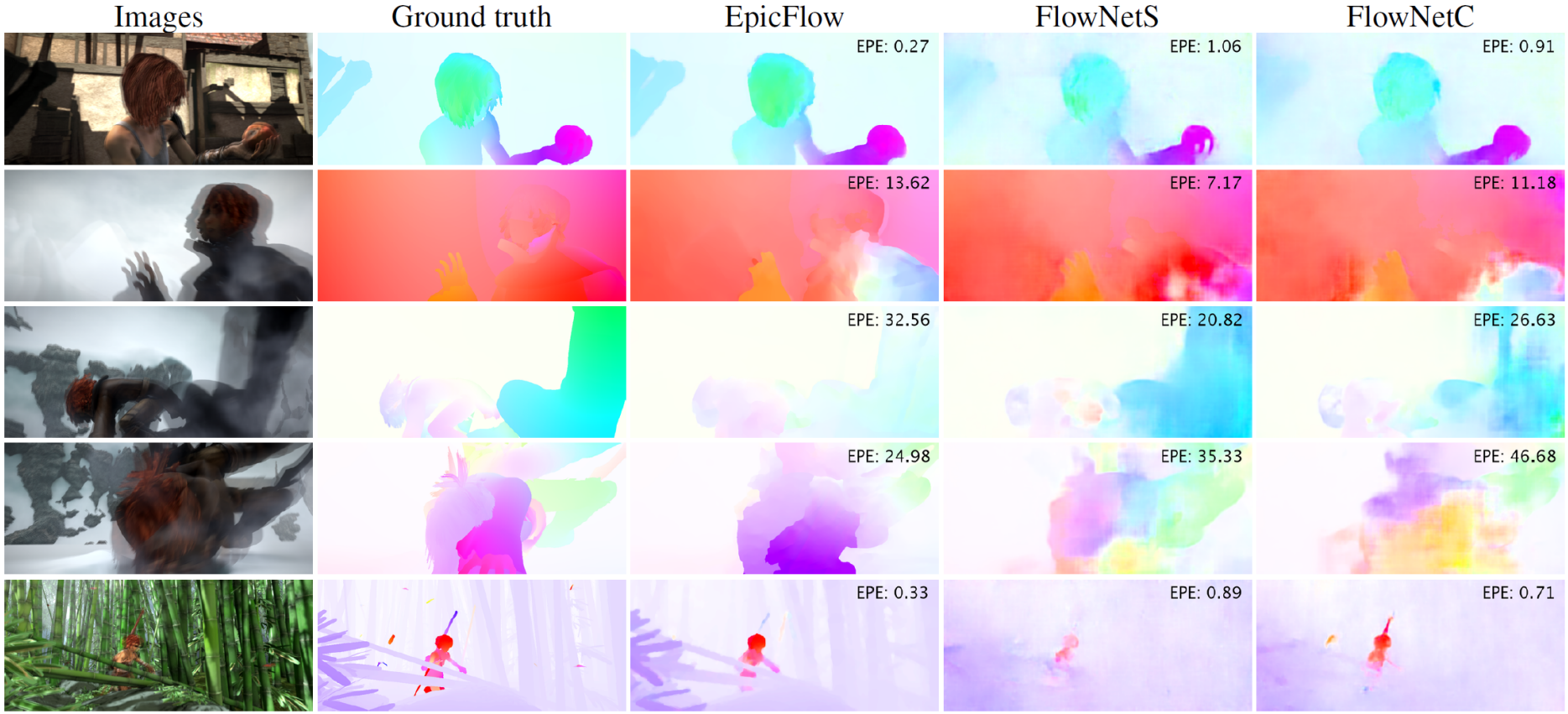
	Frame pairs	Frames with ground truth
Middlebury	72	8
KITTI	194	194
Sintel	1,041	1,041
Flying Chairs	22,872	22,872



Results on Sintel Dataset

$$L_{\text{epe}} = \frac{1}{N} \sum \sqrt{(U - U')^2 + (V - V')^2}$$

- U, V: Ground truth flow
- U', V': Estimated flow
- N: Number of pixels

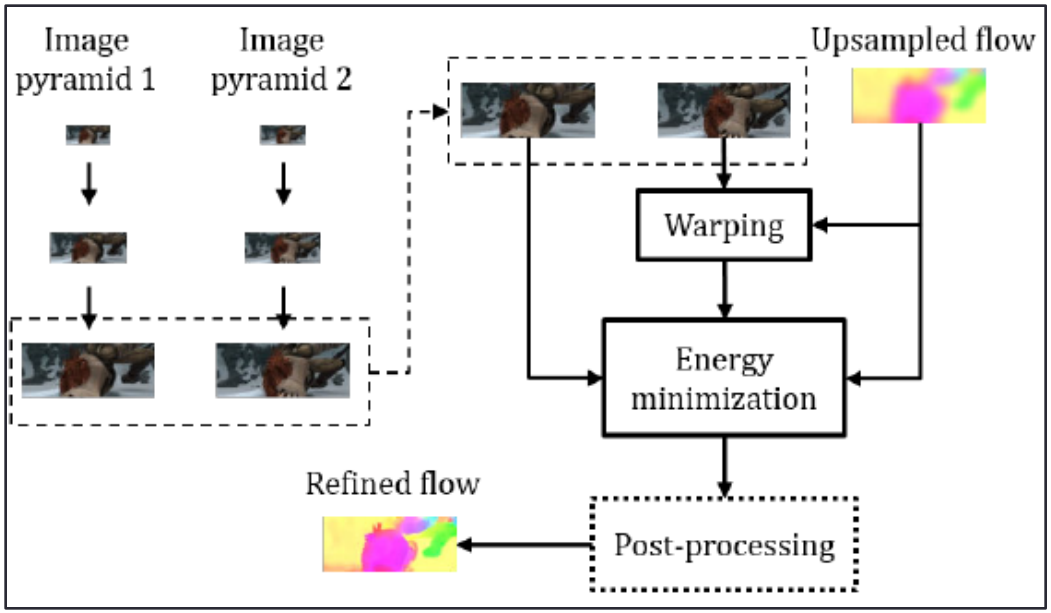


Insights from Spatial Pyramids

- Issue with learning flow: handle both large and small displacements
  - Spatio-temporal convolutions not enough to handle large motions
  - Detailed, sub-pixel flow estimation and precise motion boundaries

- Large motions: this is exactly what spatial pyramids are designed to handle!

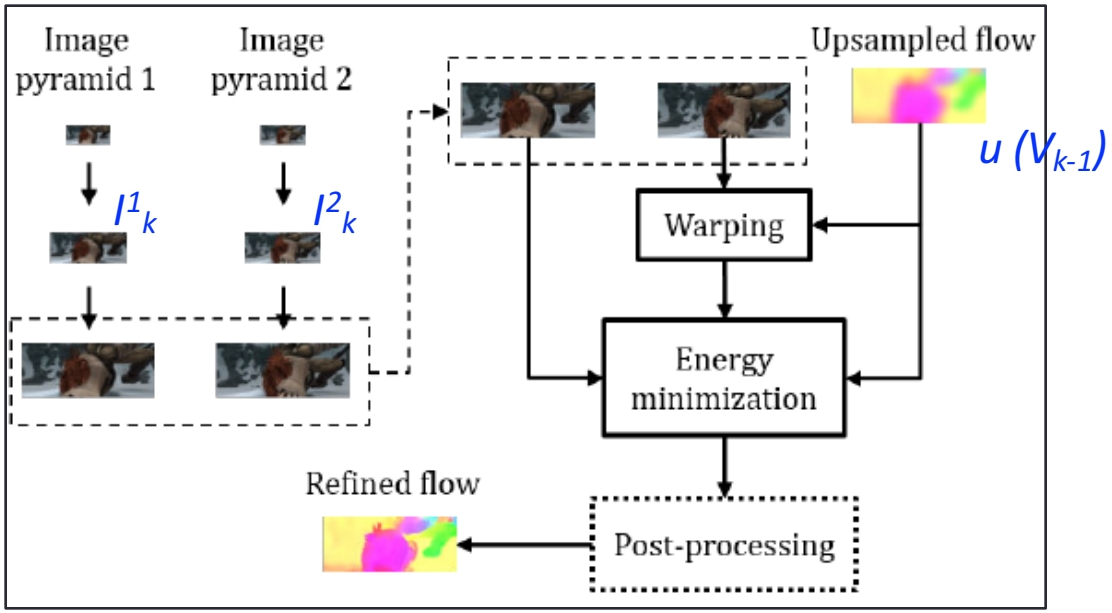
- Instead of flow, learn increment over upsampled flow at each pyramid level
- Residual flow has small magnitude, easier to learn



Insights from Spatial Pyramids

- Upsampling function: *u*
- Warping function *w(I, V)* bilinearly interpolates image *I* using flow *V*
- At pyramid level k, we have:
  - Images at appropriate resolution : I^1_k and I^2_k
  - Upsampled flow field from level *k-1* : $u\ (V_{k-1})$
- Residual flow field: difference between true and upsampled flows at level *k*

$$V_k = u(V_{k-1}) + v_k$$



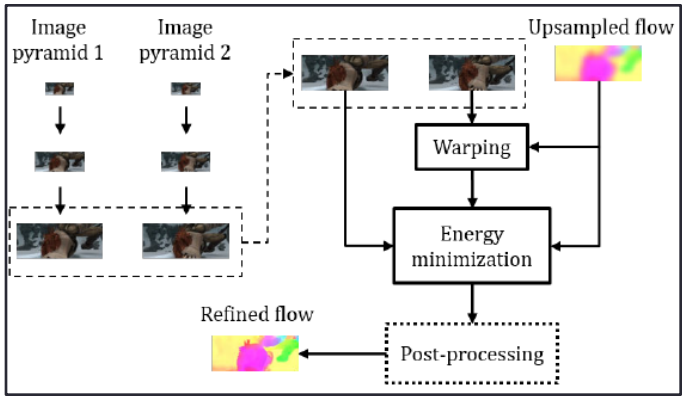
Insights from Spatial Pyramids

- Upsampling function: *u*
- Warping function *w(I, V)* bilinearly interpolates image *I* using flow *V*
- At pyramid level k, we have:
  - Images at appropriate resolution : I^1_k and I^2_k
  - Upsampled flow field from level *k-1* : $u\,(V_{k-1})$
- Residual flow field: difference between true and upsampled flows at level *k*

$$V_k = u(V_{k-1}) + v_k$$

- Can have two notions of "deep" here:
  - Deep spatial pyramids to handle large displacements
  - Deep CNNs to predict residuals at every pyramid level

$$v_k = G_k(I^1_k, w(I^2_k, u(V_{k-1})), u(V_{k-1}))$$

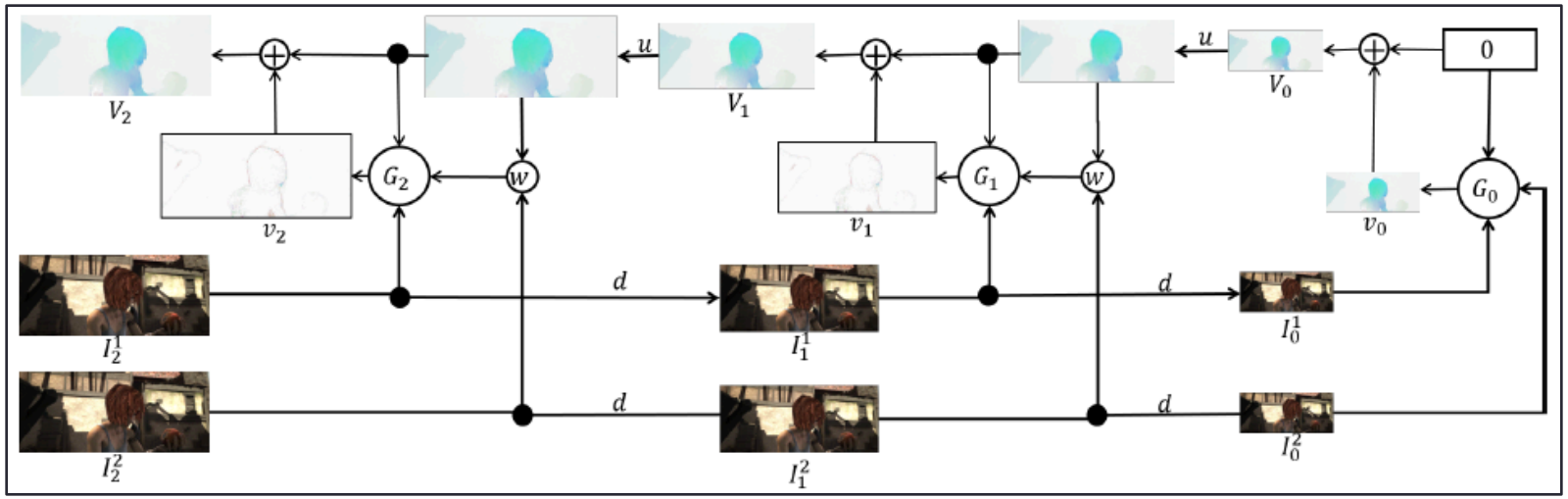


Spatial Pyramid Network

- Basic goal: learn a CNN G_k to predict residual flow at each level:

$$v_k = G_k(I_k^1, w(I_k^2, u(V_{k-1})), u(V_{k-1}))$$

- At level *k*:
  - Use I^1_k and warped I^2_k, with upsampled flow from level k-1, to predict residual
  - Add residual to upsampled flow at level *k-1* to obtain flow at level *k*



[Optical Flow Estimation Using a Spatial Pyramid Network, CVPR 2017]

Spatial Pyramid Network

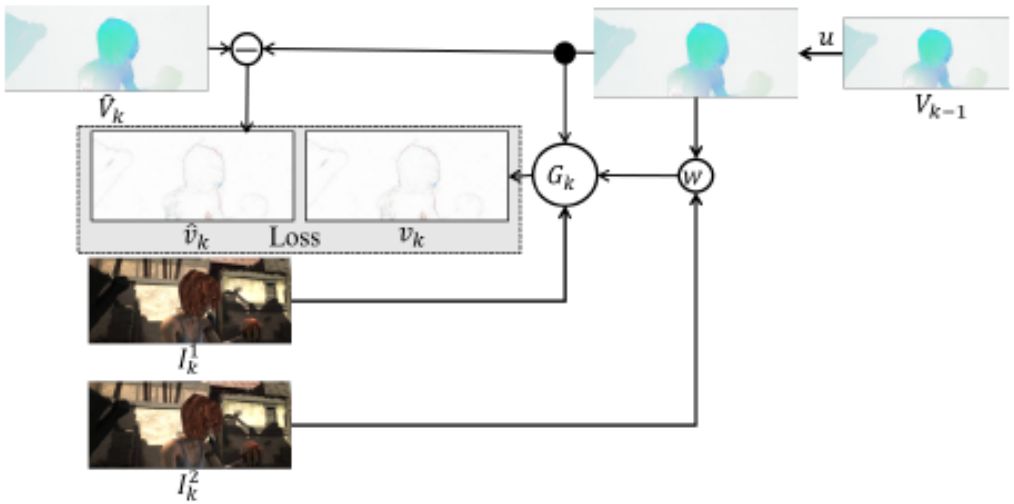
- Basic goal: learn a CNN G_k to predict residual flow at each level:

$$v_k = G_k(I_k^1, w(I_k^2, u(V_{k-1})), u(V_{k-1}))$$

- At level *k*:
  - Use I^1_k and warped I^2_k, with upsampled flow from level k-1, to predict residual
  - Add residual to upsampled flow at level *k-1* to obtain flow at level *k*

- Train each level G_k sequentially to predict residual at level *k*, given trained G_{k-1}
  - Ground truth residual = (Downsampled ground truth flow) – (Upsampled prediction)

$$\hat{v}_k = \hat{V}_k - u(V_{k-1})$$



[Optical Flow Estimation Using a Spatial Pyramid Network, CVPR 2017]

Spatial Pyramid Network

- Basic goal: learn a CNN G_k to predict residual flow at each level:

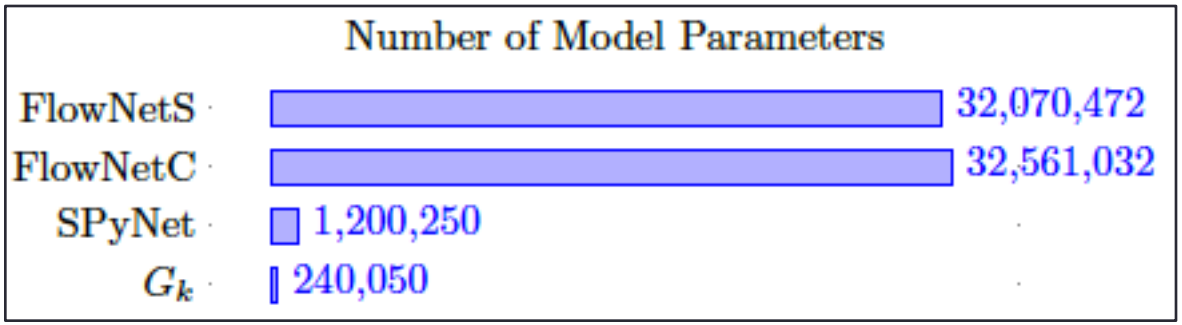
$$v_k = G_k(I_k^1, w(I_k^2, u(V_{k-1})), u(V_{k-1}))$$

- At level *k*:
  - Use I^1_k and warped I^2_k, with upsampled flow from level k-1, to predict residual
  - Add residual to upsampled flow at level *k-1* to obtain flow at level *k*

- Train each level G_k sequentially to predict residual at level *k*, given trained G_{k-1}
  - Ground truth residual = (Downsampled ground truth flow) – (Upsampled prediction)

$$\hat{v}_k = \hat{V}_k - u(V_{k-1})$$

- Each level solves a simple problem, so each level G_k can have simple architecture



[Optical Flow Estimation Using a Spatial Pyramid Network, CVPR 2017]