CSE 252D: Advanced Computer Vision

Manmohan Chandraker

Lecture 5: Learning 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 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

Papers for Fri, Apr 23

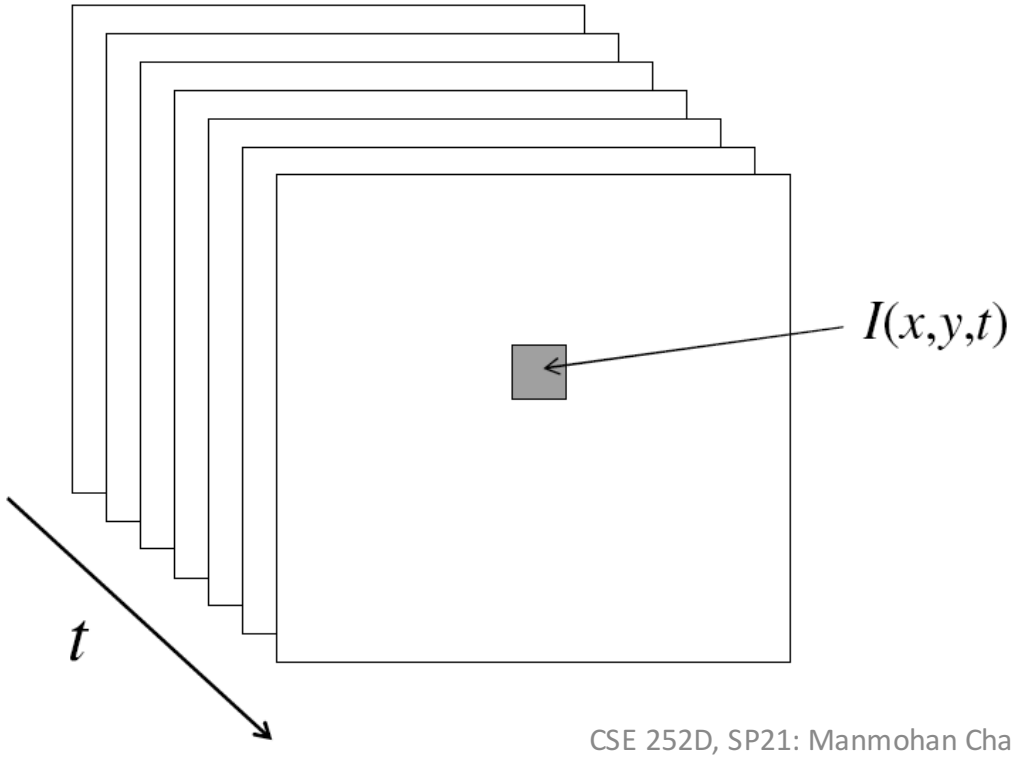
- Building Rome in a Day
  - https://grail.cs.washington.edu/rome/rome_paper.pdf
- CodeSLAM - Learning a Compact, Optimisable Representation for Dense Visual SLAM
  - https://arxiv.org/abs/1804.00874
- Beyond Tracking: Selecting Memory and Refining Poses for Deep Visual Odometry
  - https://arxiv.org/abs/1904.01892
- BA-Net: Dense Bundle Adjustment Network
  - https://arxiv.org/abs/1806.04807

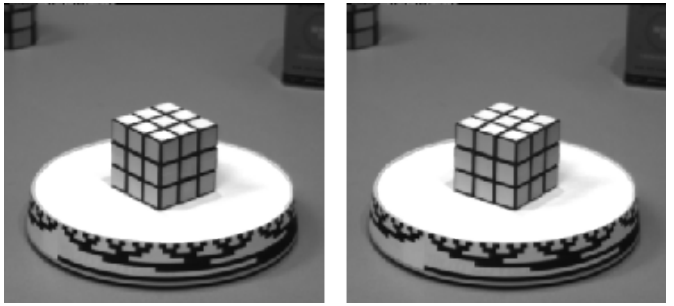
Recap

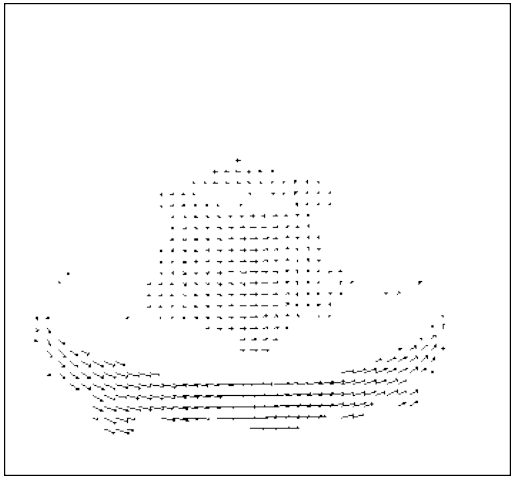
Optical flow

Brightness constancy constraint equation

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

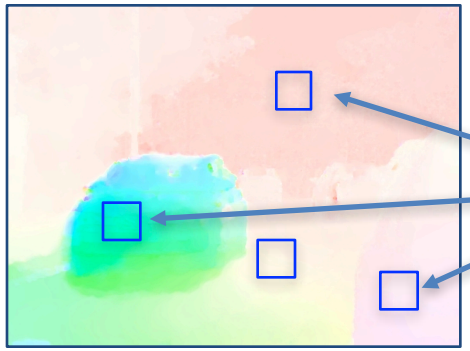






Optical flow: Lucas-Kanade

- Overconstrained linear system through patch coherence



Assume a constant flow in K x K patch

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

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

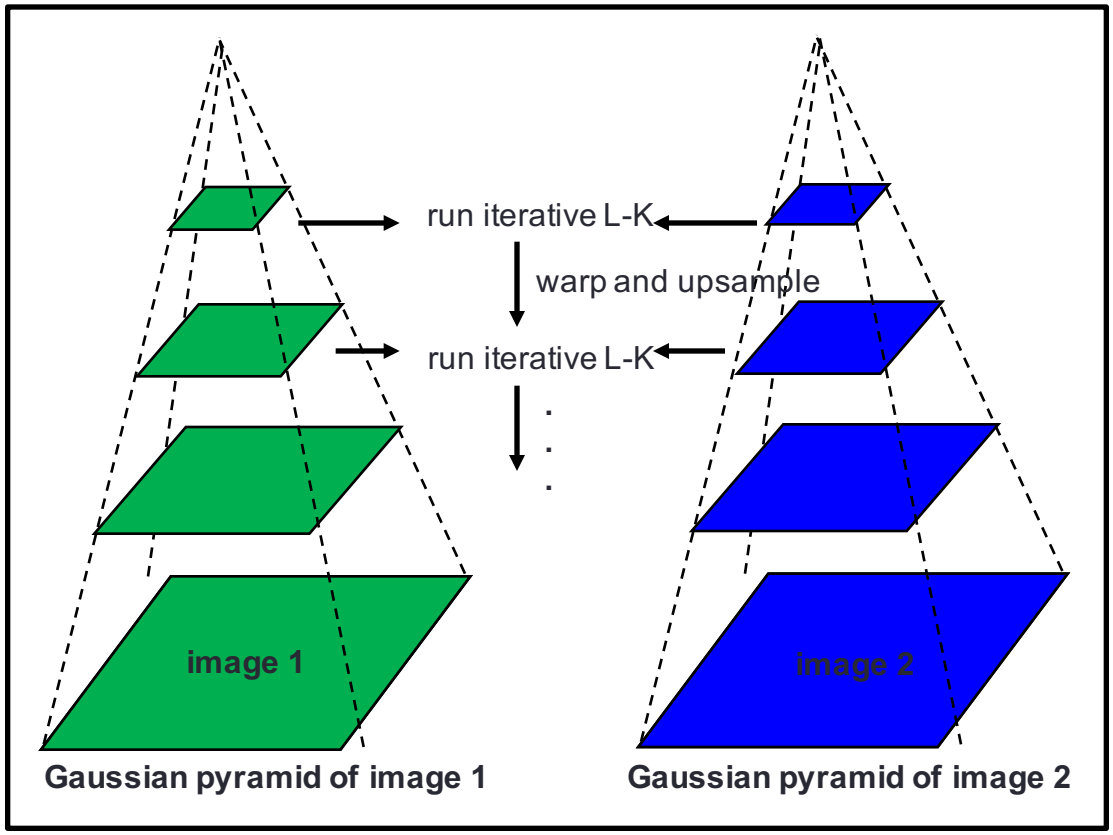
Least squares solution for *d* given by $(A^T A)\ d = A^T b$

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

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

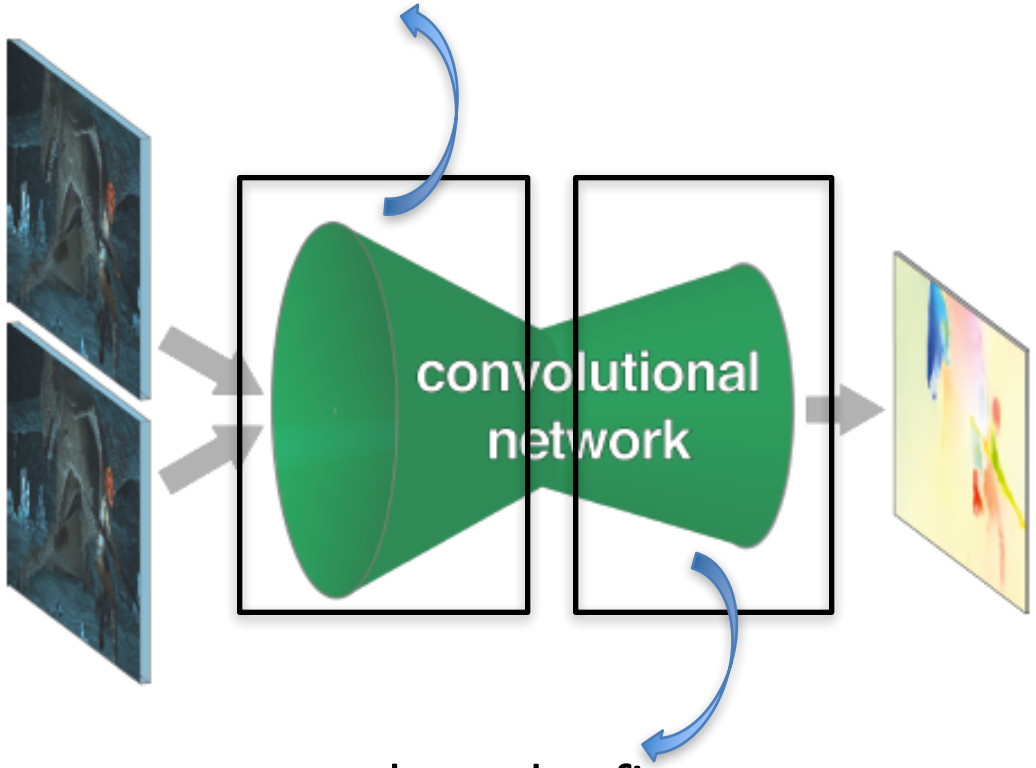
The summations are over all pixels in the K x K window

Optical flow: Coarse-to-Fine



FlowNet

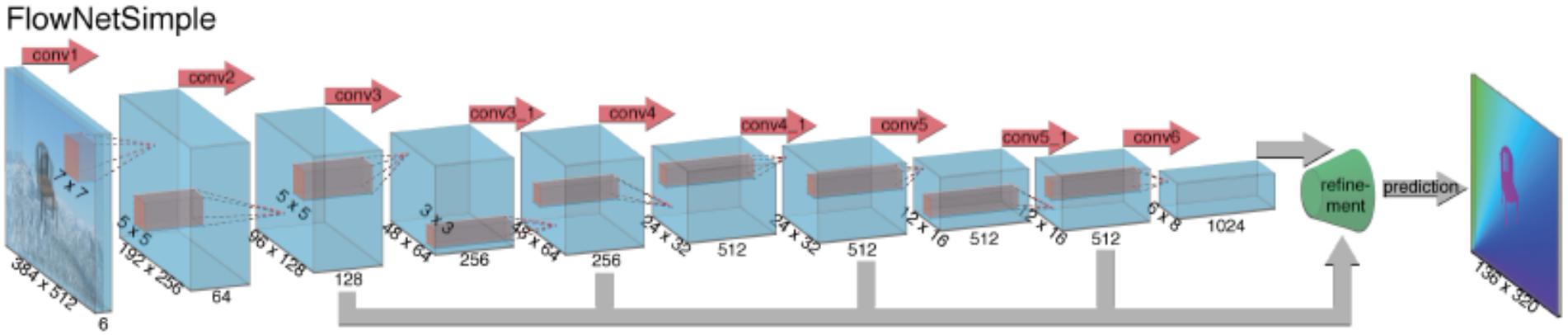
- Contracting part: extract a rich hierarchical feature representation
  - Flow is local entity, need for abstract features?



- 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]

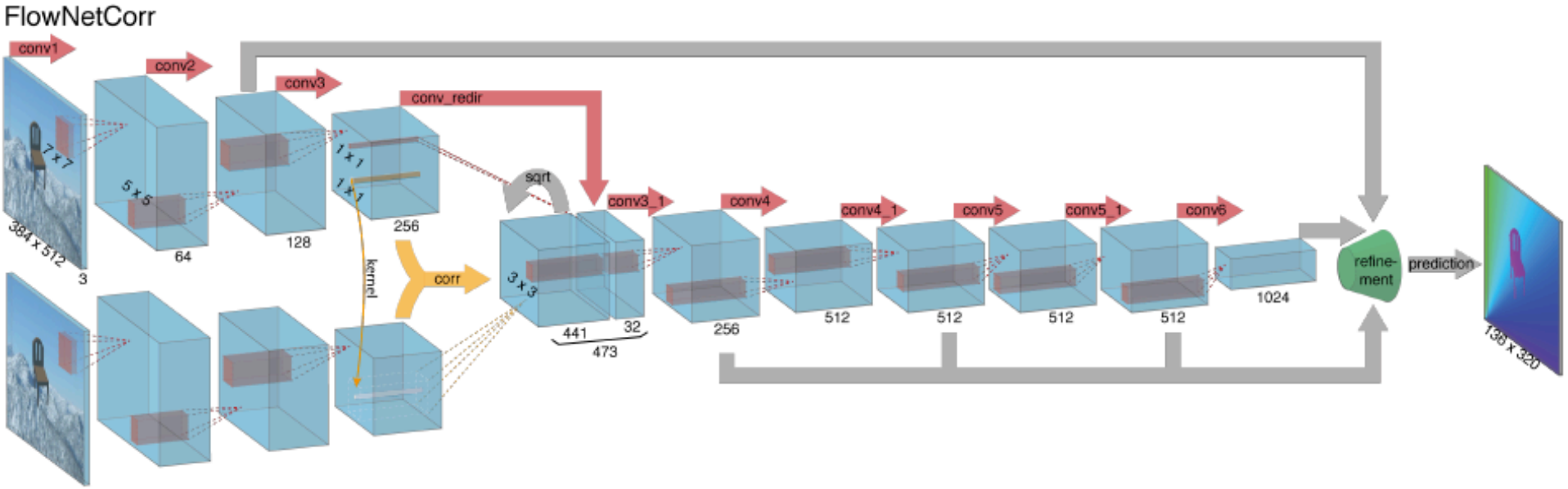
FlowNet: Simple Architecture



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

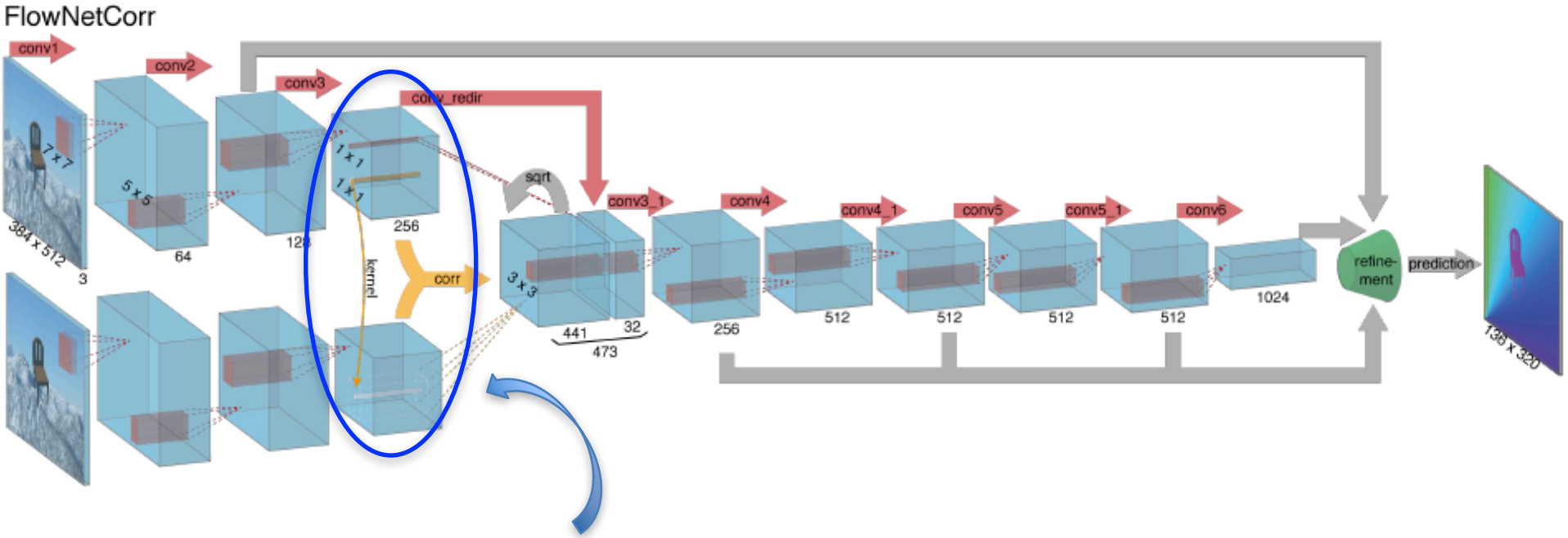
- Advantage:
  - Simple!
- Disadvantage:
  - Unclear how well local SGD-based optimization can do

FlowNet: Correlation Architecture



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

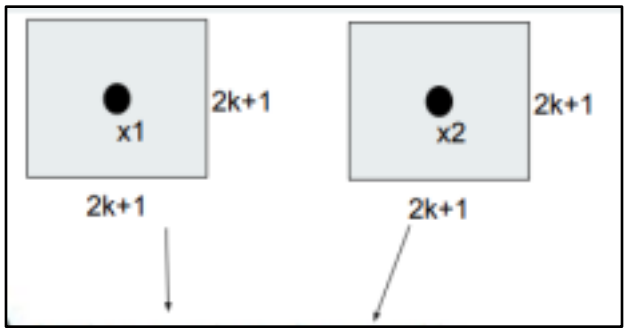
FlowNet: Correlation Layer



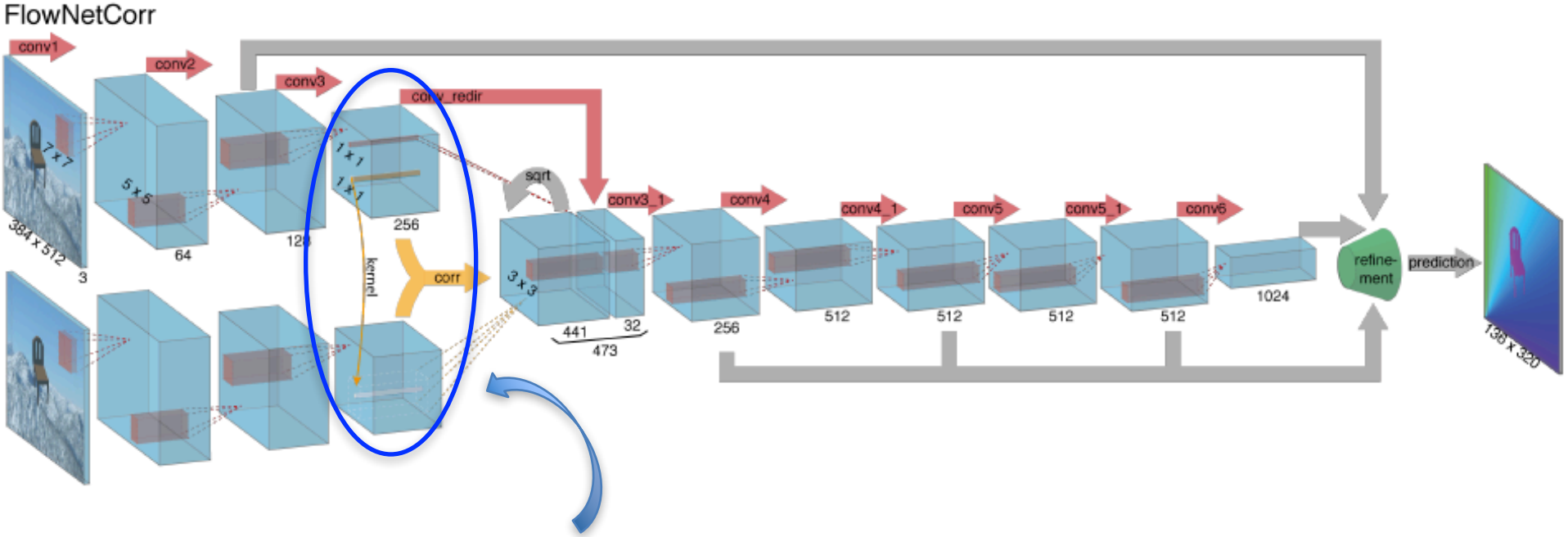
- Multi-channel feature maps $\mathbf{f}_1$ and $\mathbf{f}_2$, of size *w* x *h* and *n* channels
- Correlation of patches centered at $\mathbf{x}_1$ and $\mathbf{x}_2$:

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

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



FlowNet: Correlation Layer

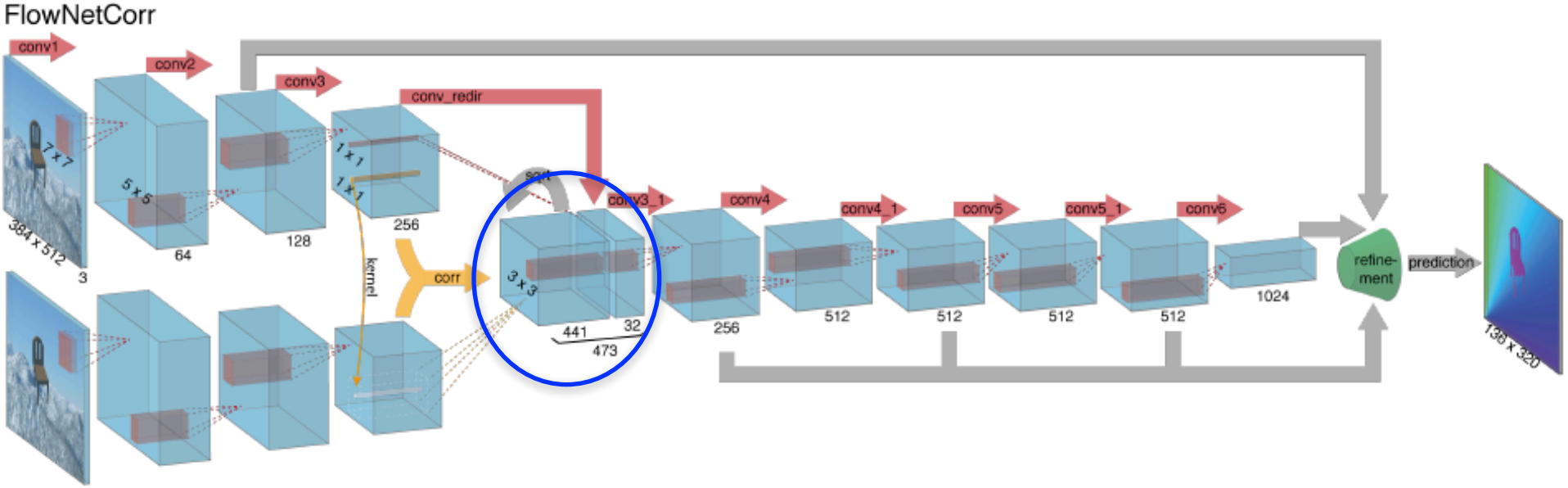


- Multi-channel feature maps $\mathbf{f}_1$ and $\mathbf{f}_2$, of size w x h and n channels
- Correlation of patches centered at $\mathbf{x}_1$ and $\mathbf{x}_2$:

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

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

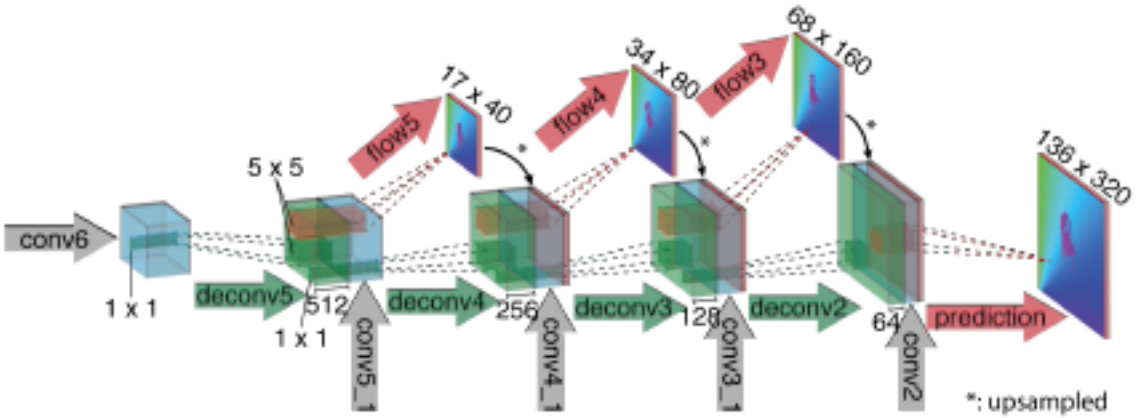
FlowNet: Correlation Layer



- 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

FlowNet: Refinement

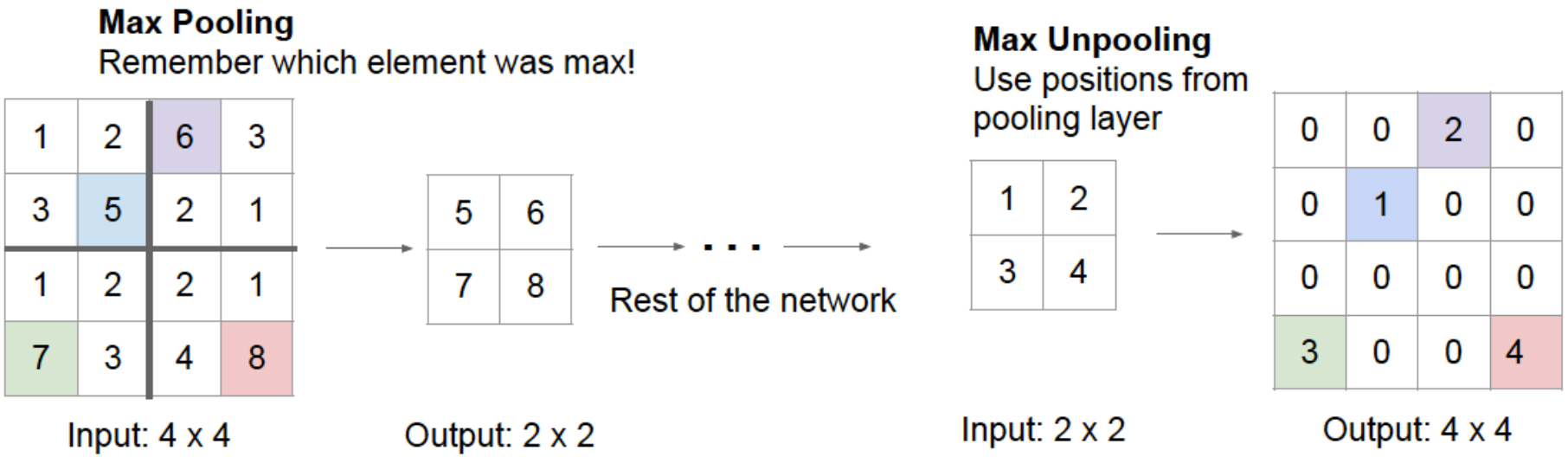


- Gradually upsample the low-dimensional feature.
- Concatenate with encoder feature of corresponding scale, to recover details.

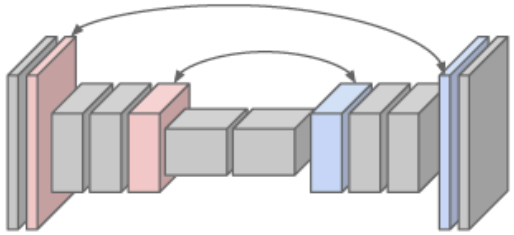
- In simplest form, upsampling can be implemented as bilinear interpolation.

- 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

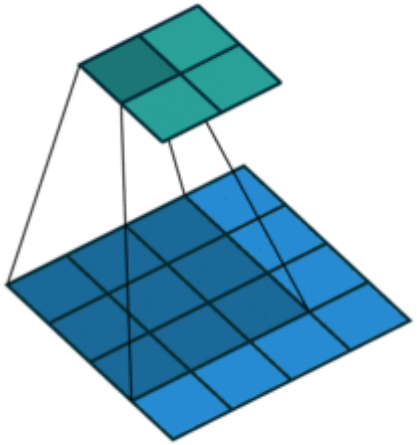
$\star$

1	2	3
4	5	6
7	8	9

$=$

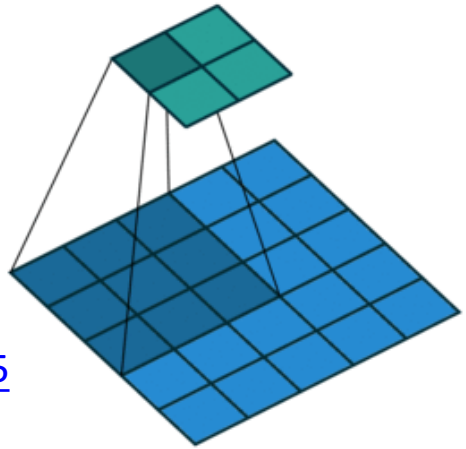
Filter, a

Input, x



https://arxiv.org/abs/1603.07285

[Dumolin and Visin, 2018]



Normal Convolutions

1	2
3	4

$\star$

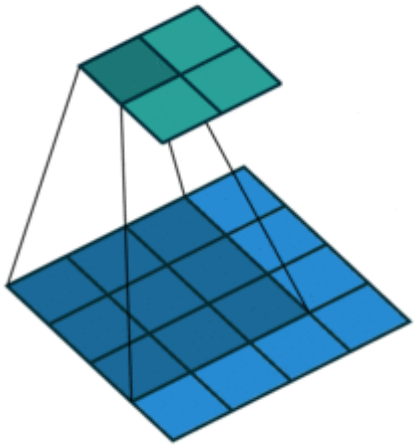
1	2	3
4	5	6
7	8	9

$=$

37	47
67	77

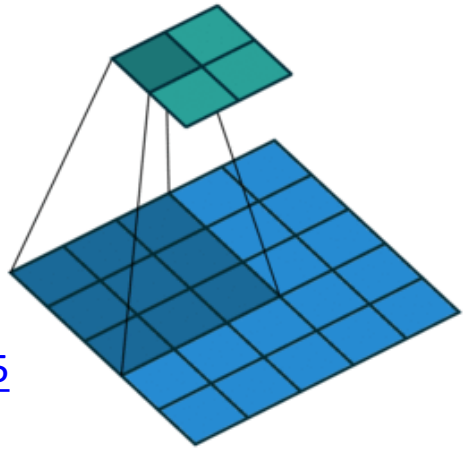
Filter, a

Input, x



https://arxiv.org/abs/1603.07285

[Dumolin and Visin, 2018]



Normal Convolutions

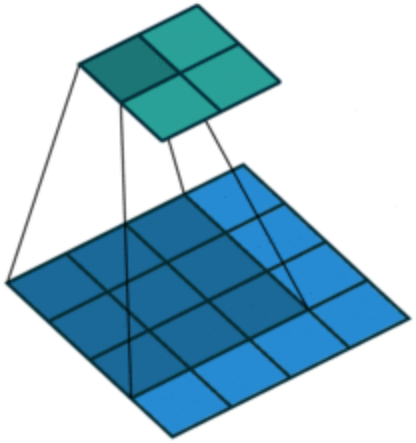
1	2
3	4

☆

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

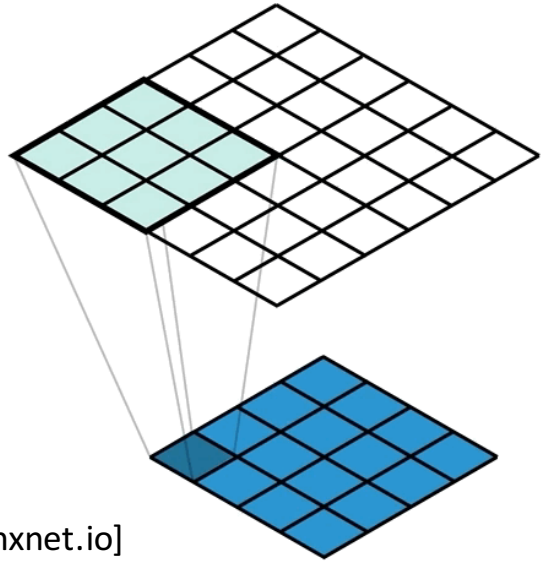
Filter, a

$\overset{T}{\star}$

1	0
2	1

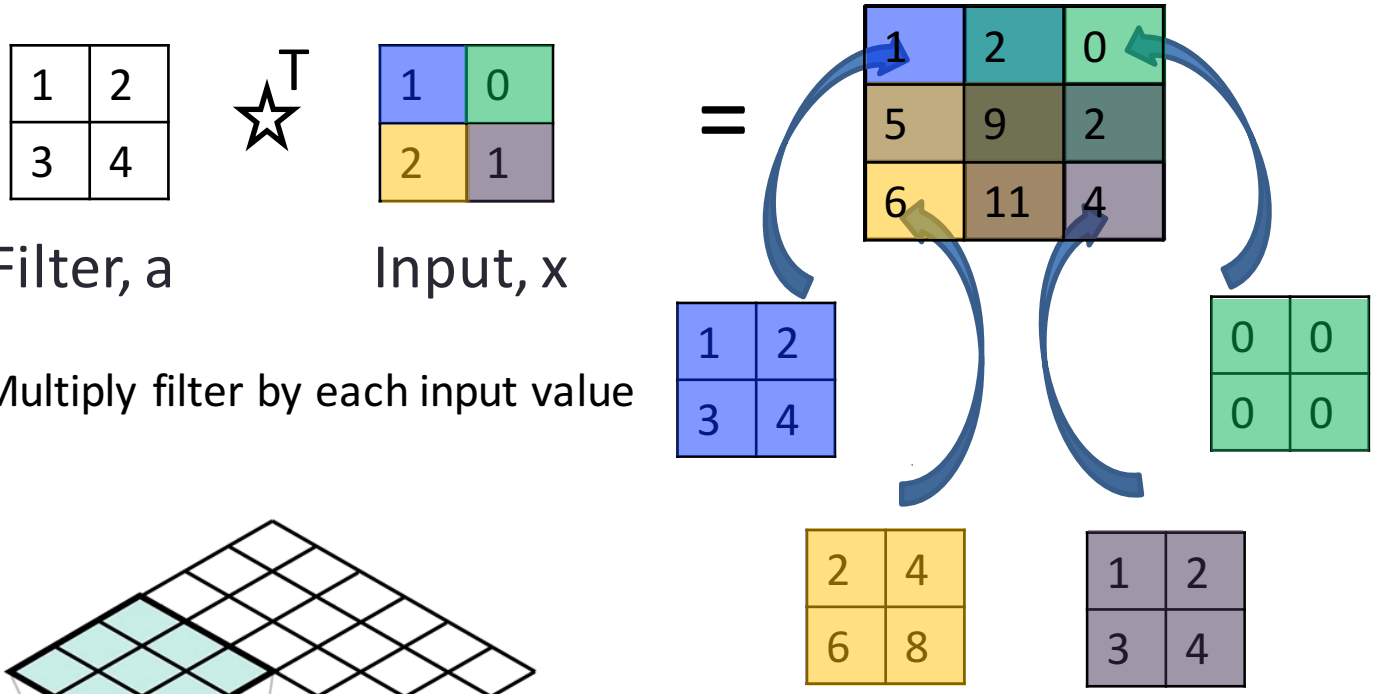
Input, x

$=$



[Thom Lane, mxnet.io]

Transposed Convolutions

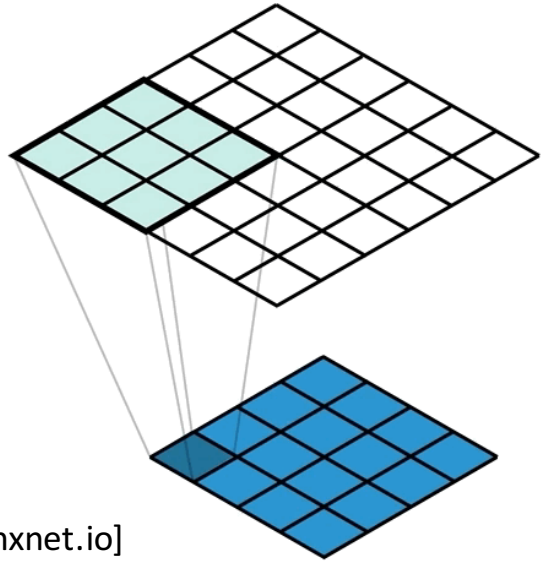


Filter, a

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

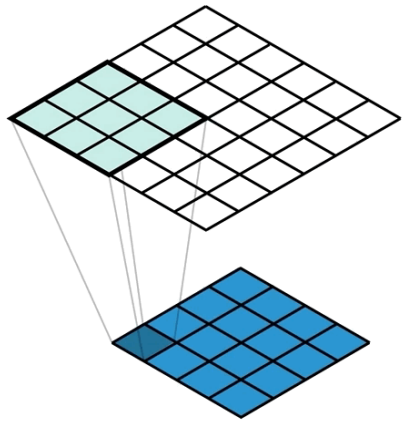
A^T

X

1
0
2
1

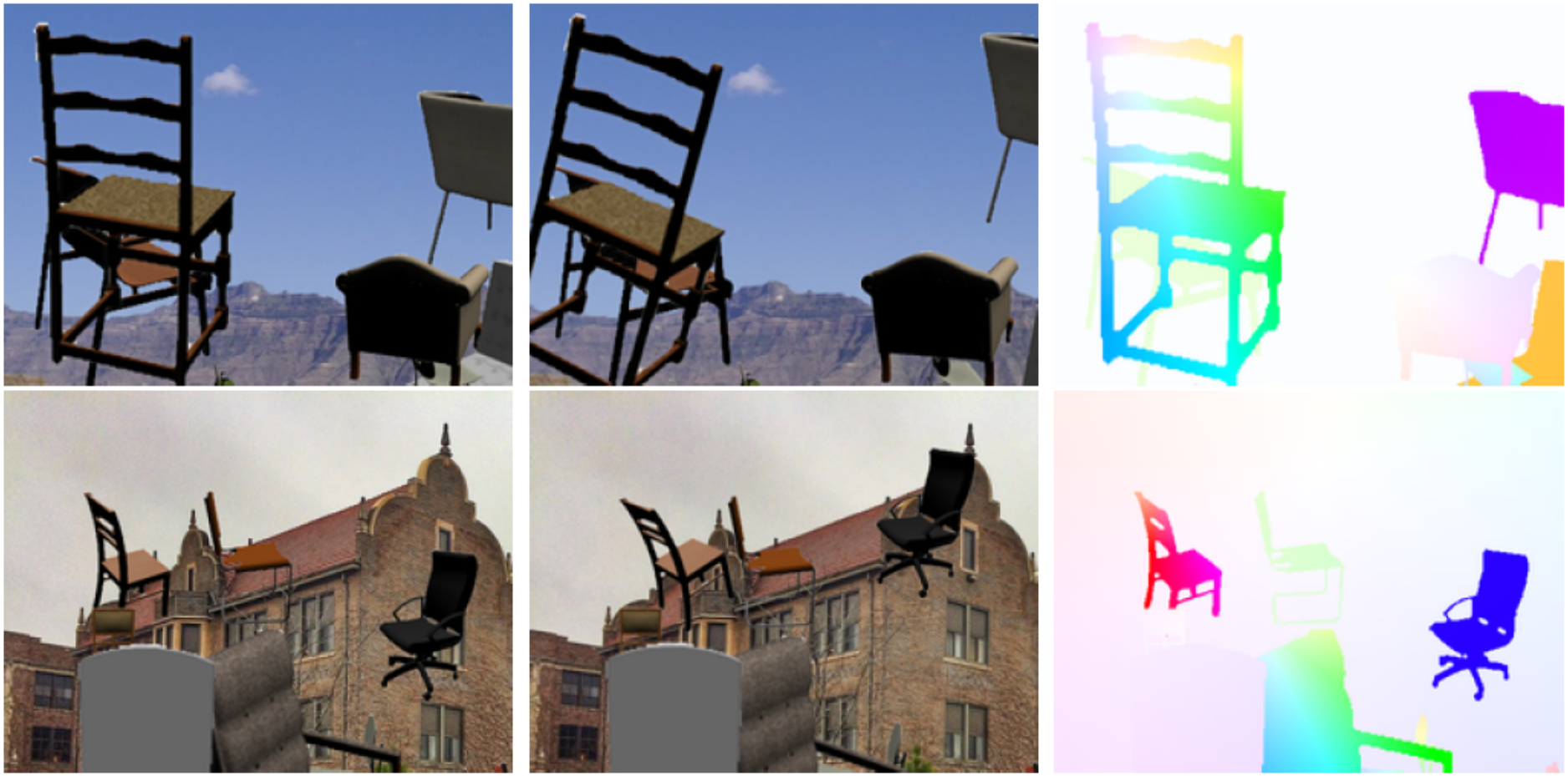
$=$

1
2
0
5
9
2
6
11
4



Synthetic Training Data

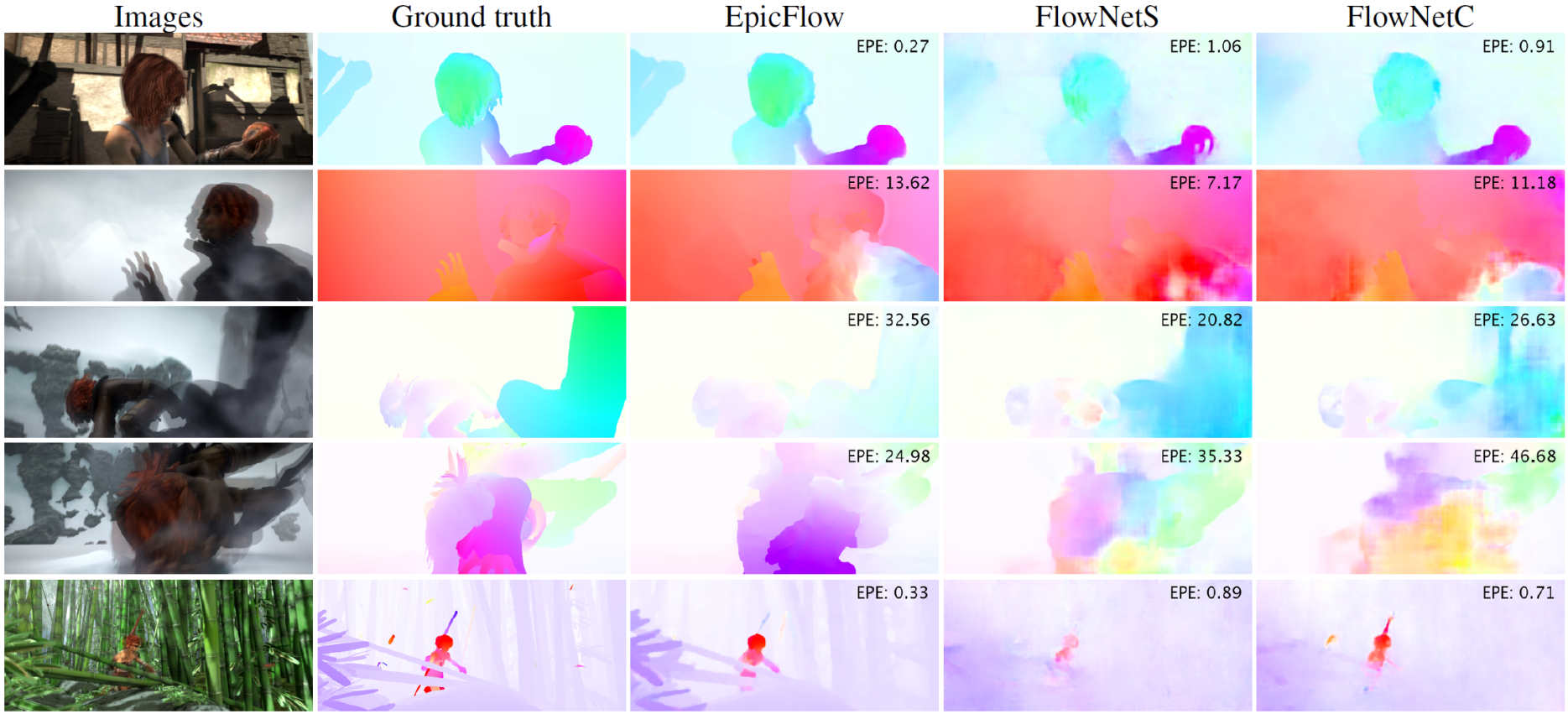
	Frame pairs	Frames with ground truth
Middlebury	72	8
KITTI	194	194
Sintel	1,041	1,041
Flying Chairs	22,872	22,872



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

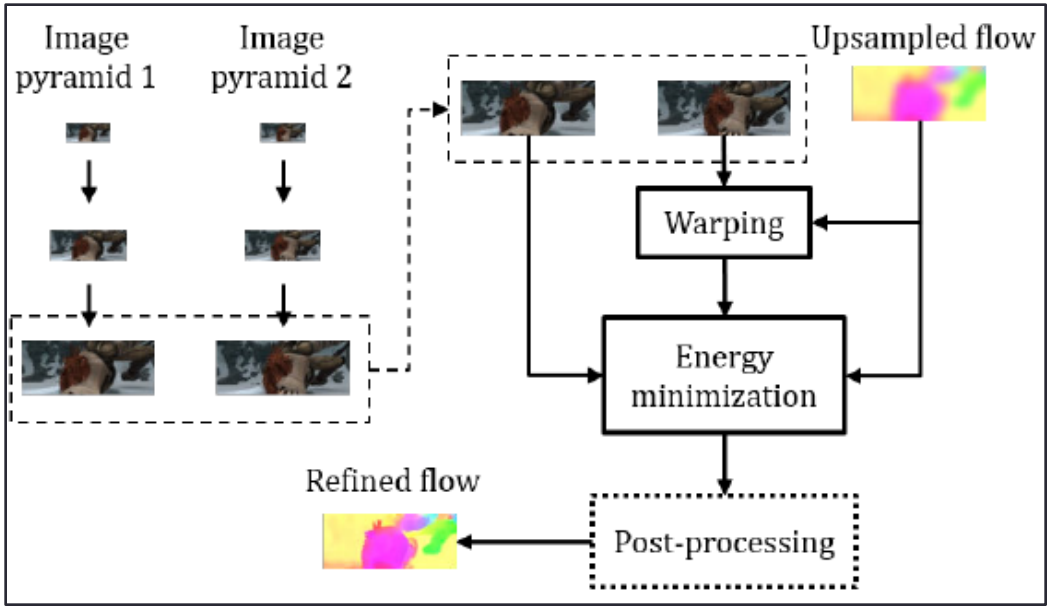


Insights from Spatial Pyramids

- 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

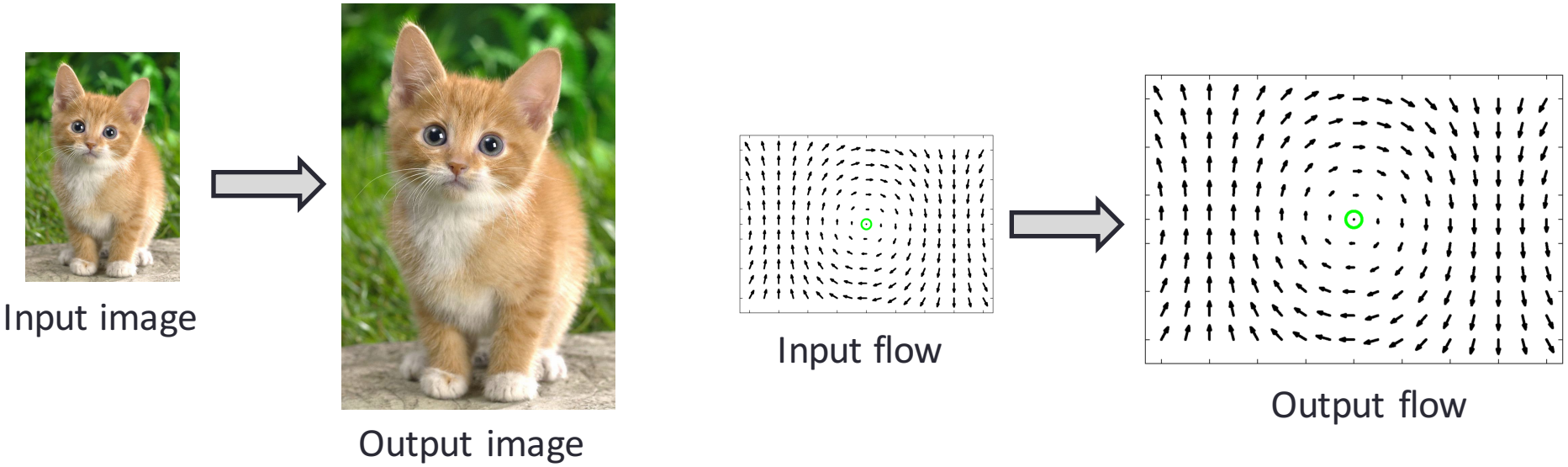
- Large motions: this is exactly what spatial pyramids are designed to handle!

- 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
  - Can apply to an image
  - Can apply to each component (x and y) of optical flow



Insights from Spatial Pyramids

- Upsampling function: *u*
- Warping function *w(I, V)* bilinearly interpolates image *I* using flow *V*

$$w\colon (I, V) \rightarrow I' \;:\; I'(x, y) = I(x + V_1, y + V_2)$$

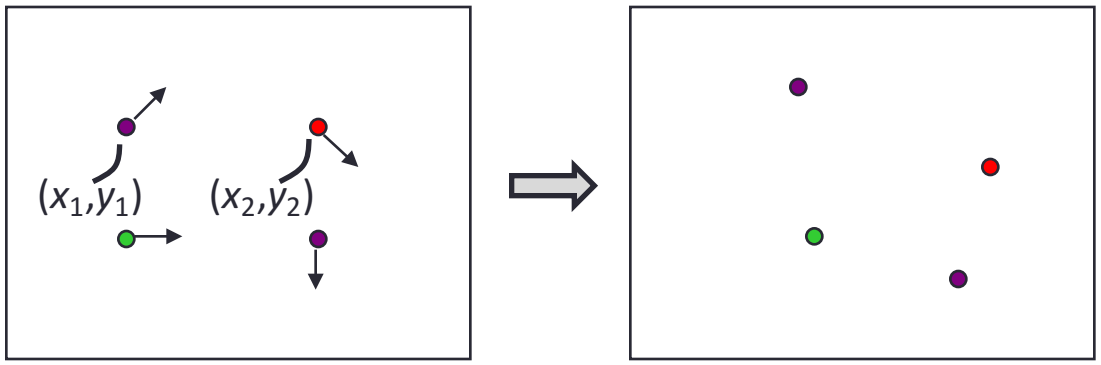
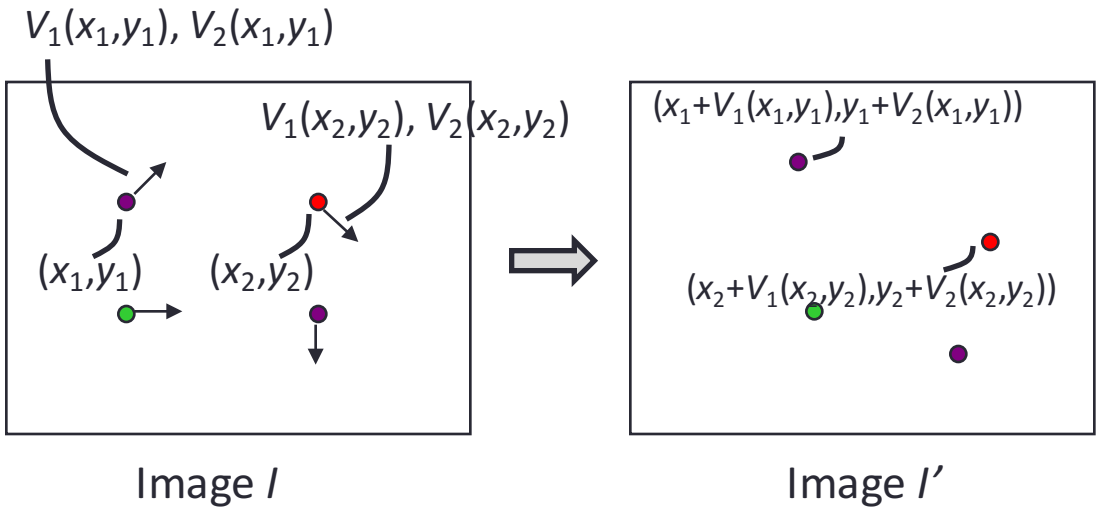


Image *I* Image *I'*

Insights from Spatial Pyramids

- Upsampling function: *u*
- Warping function *w(I, V)* bilinearly interpolates image *I* using flow *V*

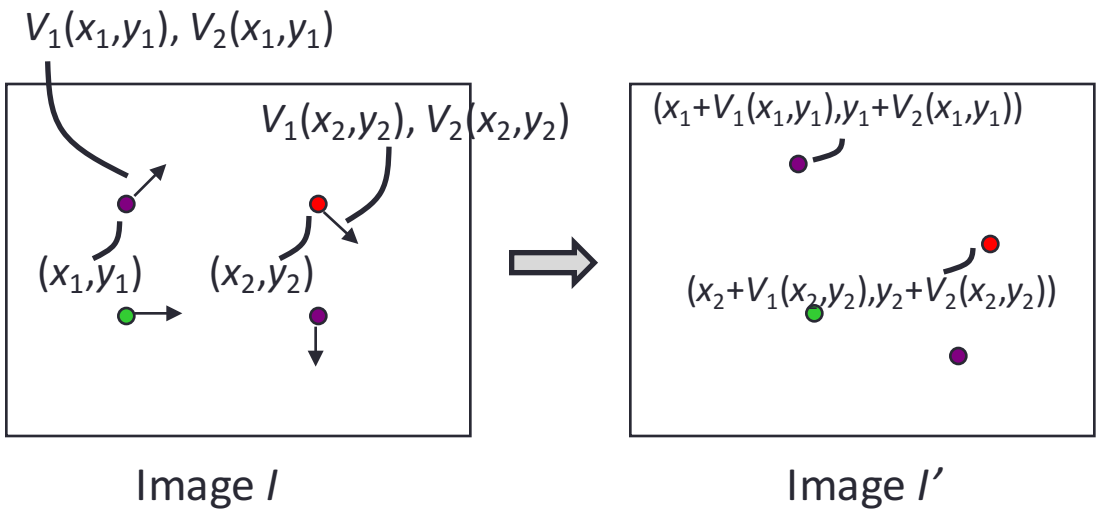
$$w\colon (I, V) \rightarrow I' \;:\; I'(x, y) = I(x + V_1, y + V_2)$$

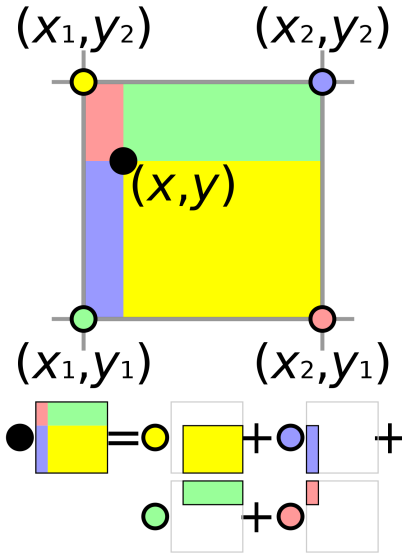


Insights from Spatial Pyramids

- Upsampling function: *u*
- Warping function *w(I, V)* bilinearly interpolates image *I* using flow *V*

$$w\colon (I, V) \rightarrow I' \;:\; I'(x, y) = I(x + V_1, y + V_2)$$





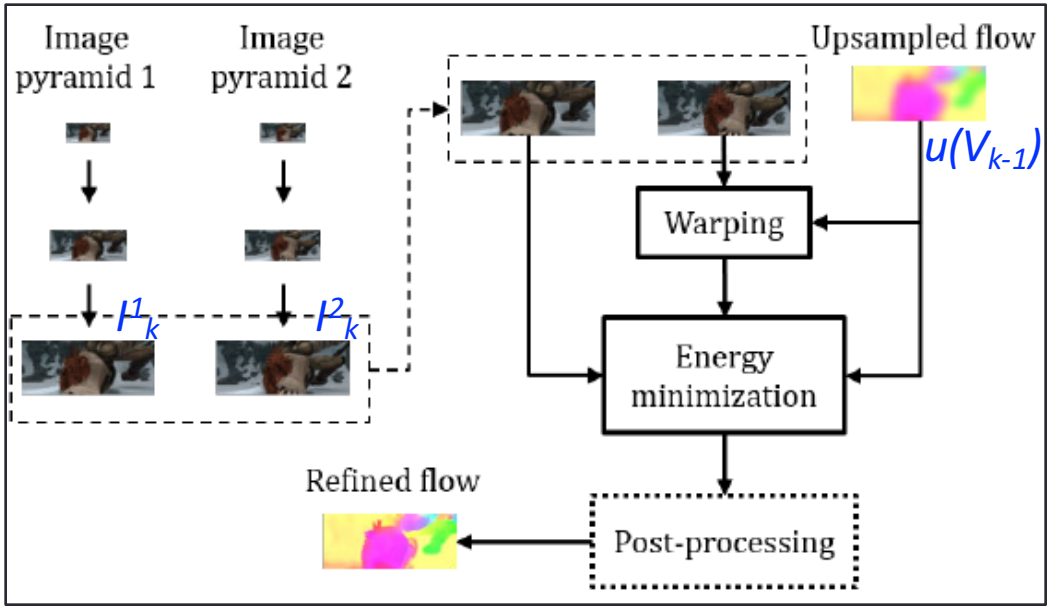
Insights from Spatial Pyramids

- Upsampling function: u
- Warping function $w(I, V)$ bilinearly interpolates image I using flow V

$$w: (I, V) \rightarrow I' \;:\; I'(x, y) = I(x + V_1, y + V_2)$$

- At pyramid level k, we have:
  - Images at appropriate resolution : I^1_k and I^2_k
  - Upsampled flow field from level $k\text{-}1$: $u(V_{k-1})$
- Residual flow field: difference between true and upsampled flows at level k

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



Insights from Spatial Pyramids

- Upsampling function: u
- Warping function $w(I, V)$ bilinearly interpolates image I using flow V

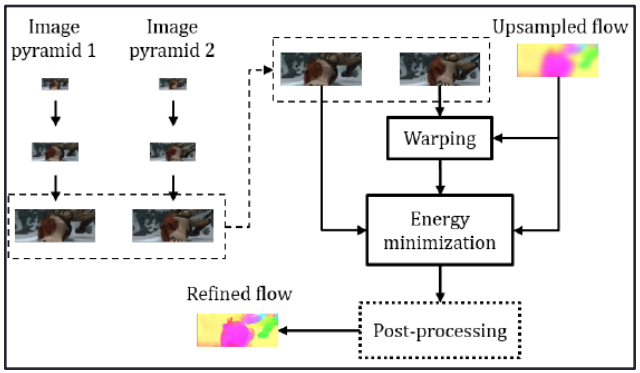
$$w: (I, V) \rightarrow I' \;:\; I'(x, y) = I(x + V_1, y + V_2)$$

- At pyramid level k, we have:
  - Images at appropriate resolution : I^1_k and I^2_k
  - Upsampled flow field from level $k\text{-}1$: $u(V_{k-1})$
- Residual flow field: difference between true and upsampled flows at level k

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

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

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

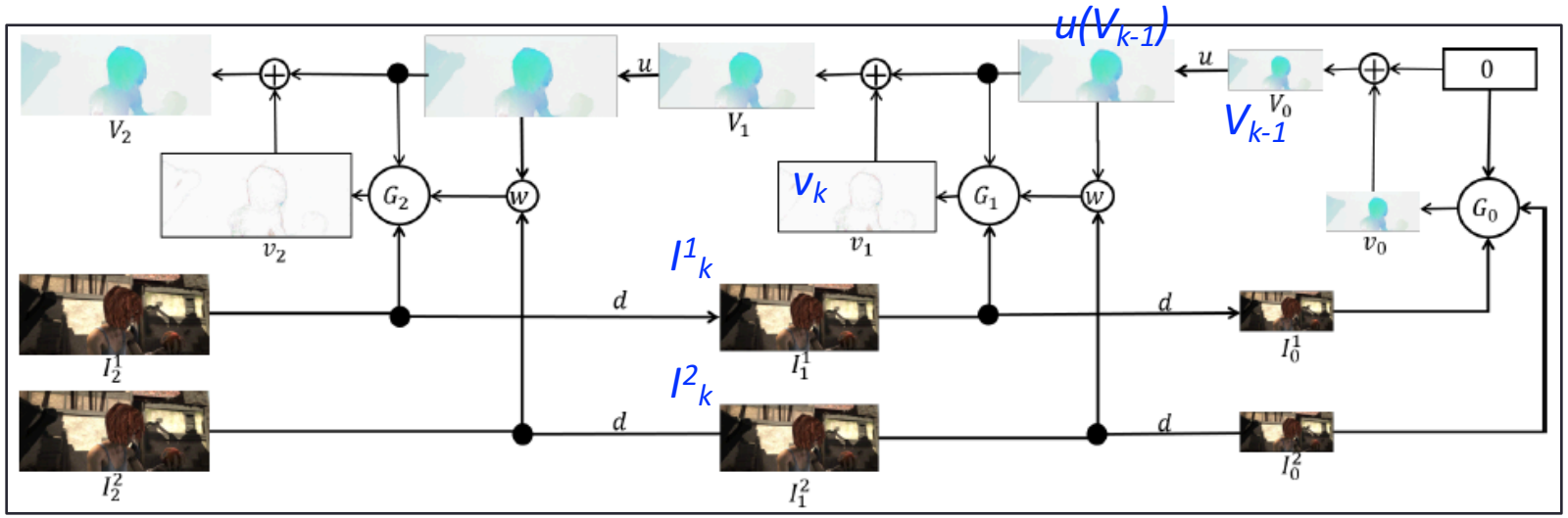


Spatial Pyramid Network

- 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

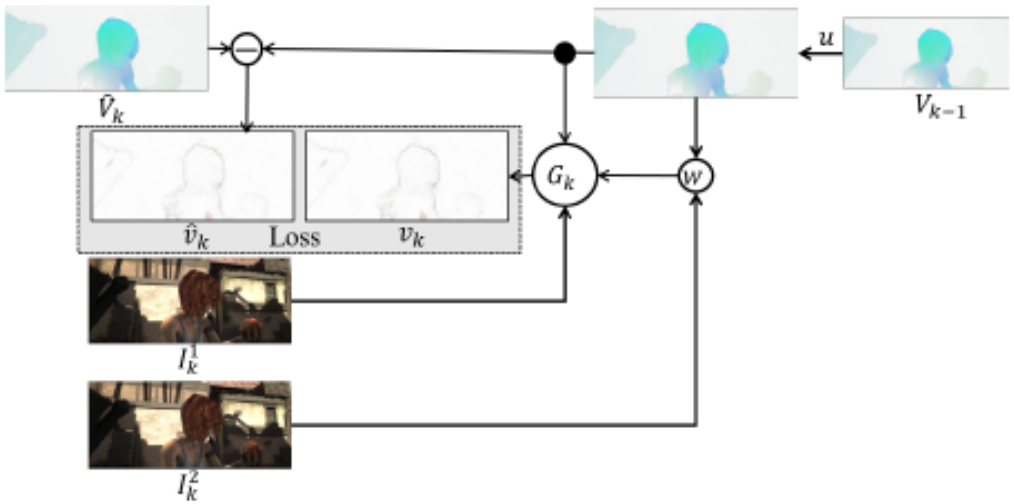
- Basic goal: learn a CNN G_k to predict residual flow at each level:

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

- 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:

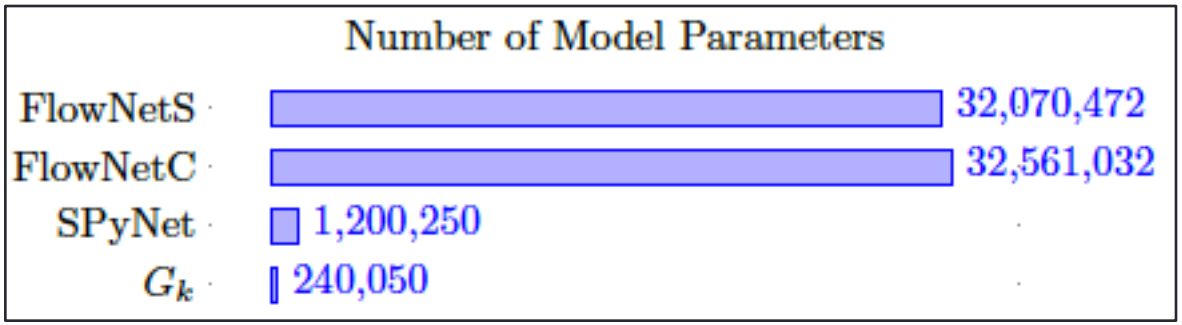
$$v_k = G_k(I_k^1, w(I_k^2, u(V_{k-1})), u(V_{k-1}))$$

- 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

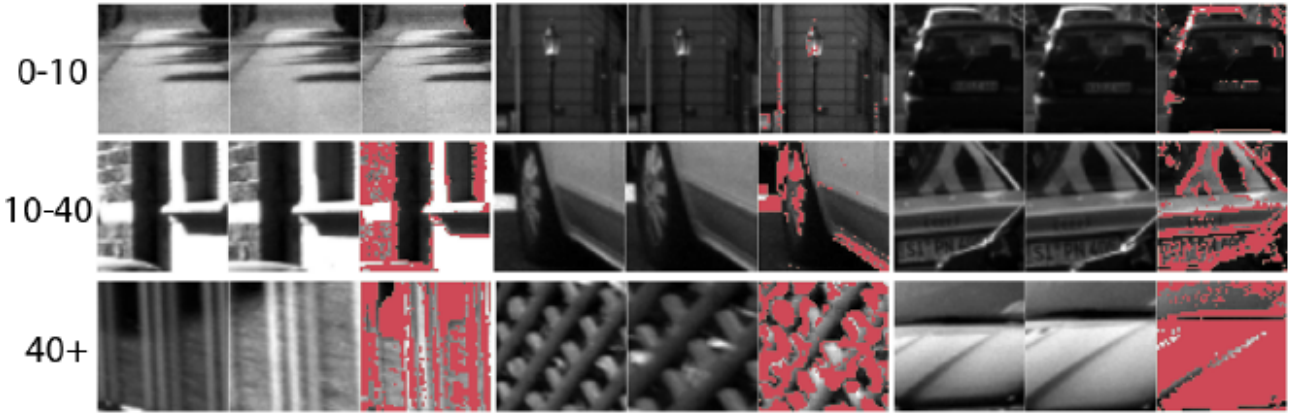


[Optical Flow Estimation Using a Spatial Pyramid Network, CVPR 2017]

Training Strategy Matters

- Large motions lead to larger appearance changes:
  - As the camera moves, background changes
  - With change in viewing direction, occlusions occur
  - Illumination changes
  - Scale changes for objects in the scene

- Need to balance training for small and large displacements
  - Small: rely more on appearance changes
  - Large: rely more on context information



For pairs of patches at various displacements, red dots show locations with intensity differences more than a threshold.

Training Strategy Matters

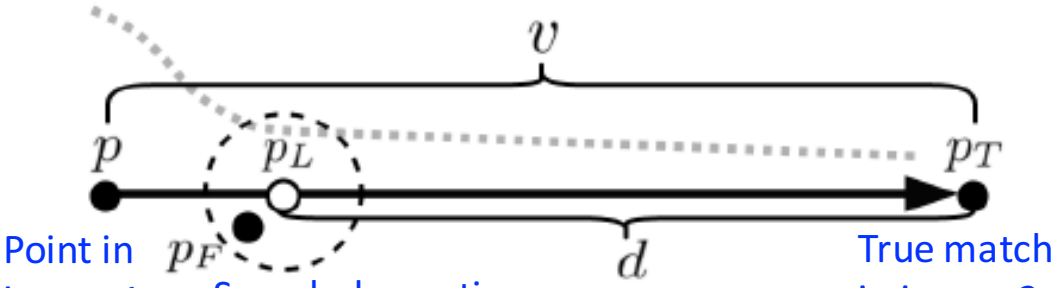
- Curriculum learning: pre-defined easy to difficult order for training samples
  - Smaller displacements can be considered easier (less appearance change)
  - Start training from samples with lower displacements
  - **Instructor-driven:** assumes student solves more difficult tasks over time

- Self-paced learning: identify easy samples based on training loss and use them
  - Learn only from the easy examples in each epoch
  - Easy examples: those with loss lower than a threshold
  - Increase the threshold over epochs to raise training difficulty
  - **Student-driven:** feedback from model to adjust difficulty of next iteration

- Issue: models can overly adapt to a particular training stage
  - For optical flow to be competitive, must excel in low displacement regime
  - Shift to higher difficulties (larger displacements) can be detrimental

Training Strategy Matters

- Interleaved learning:
  - Train for all displacement levels
  - Sample further negatives for larger displacements



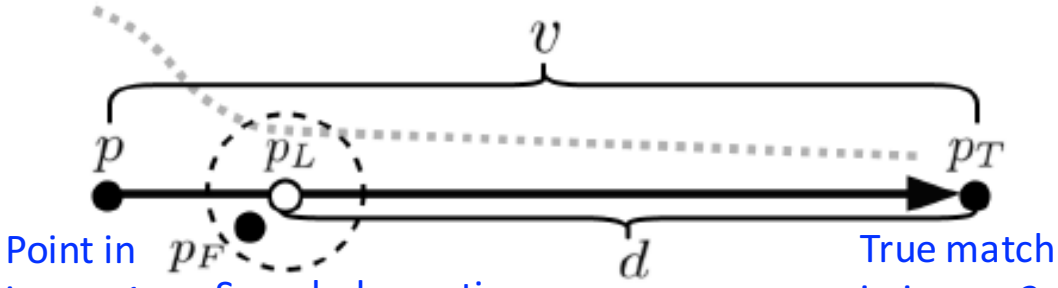
$$d = v(1 - X) \qquad X \sim \log \mathcal{N}(\mu, \sigma)$$

$$P(X = x) = \frac{1}{\sigma x \sqrt{2\pi}} e^{(-\frac{(ln(x)-\mu)^2}{2\sigma^2})}$$

[Optical Flow Requires Multiple Strategies, ICCV 2017]

Training Strategy Matters

- Interleaved learning:
  - Train for all displacement levels
  - Sample further negatives for larger displacements



$$d = v(1 - X) \qquad X \sim \log \mathcal{N}(\mu, \sigma)$$

$$P(X = x) = \frac{1}{\sigma x \sqrt{2\pi}} e^{(-\frac{(ln(x)-\mu)^2}{2\sigma^2})}$$

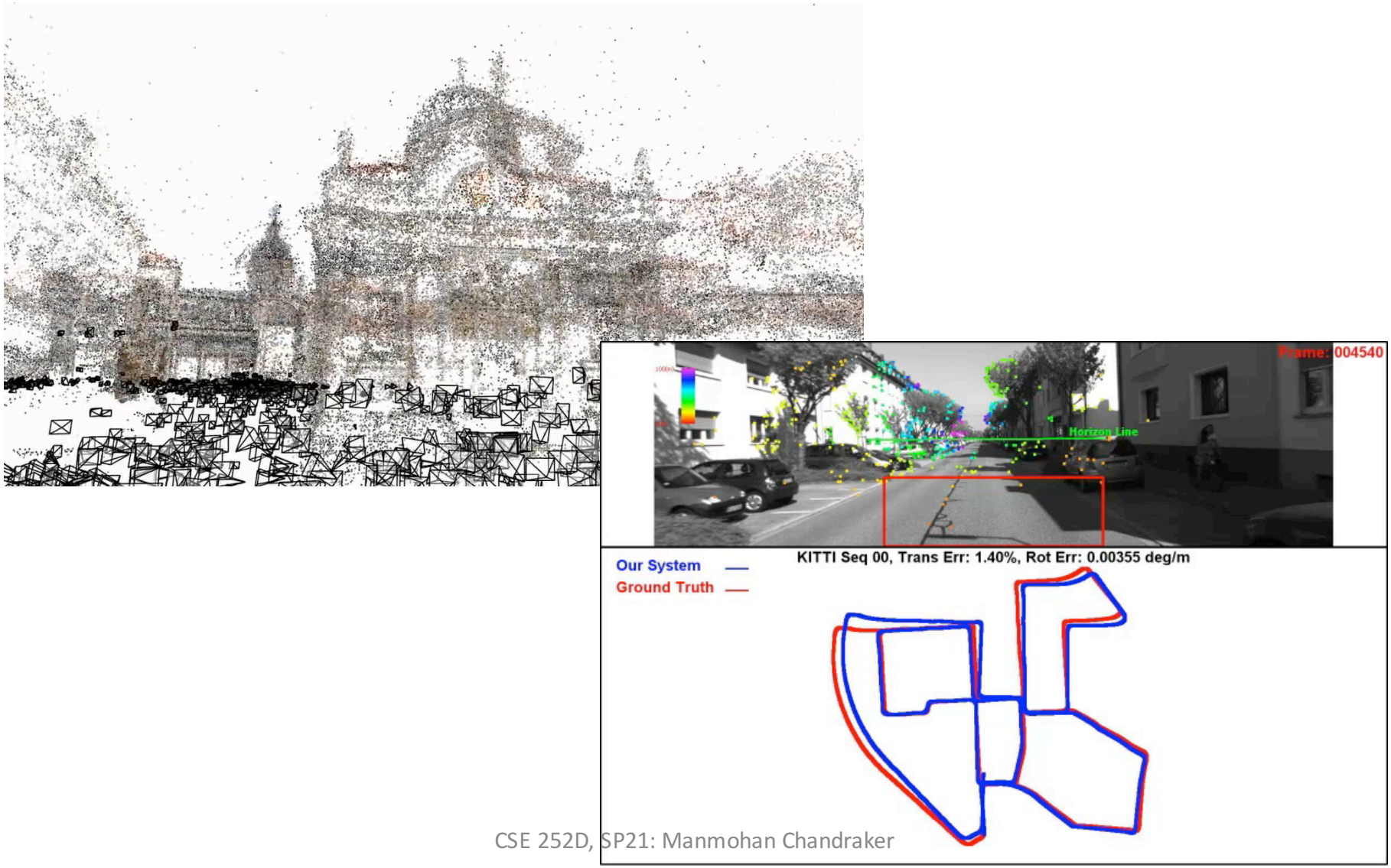
- Add a self-paced curriculum to enhance difficulty of negatives in each epoch
  - Reduce sampling distance from true positive
  - For epoch i, with total m epochs and loss given by l:

$$d_i = v(1 - X - R_i) \text{ , where } \quad R_i = \underbrace{\frac{i}{m}}_{\text{curriculum}} \underbrace{\cdot \max(0, 1 - \frac{l_{i-1}}{l_{init}})}_{\text{self-paced}}$$

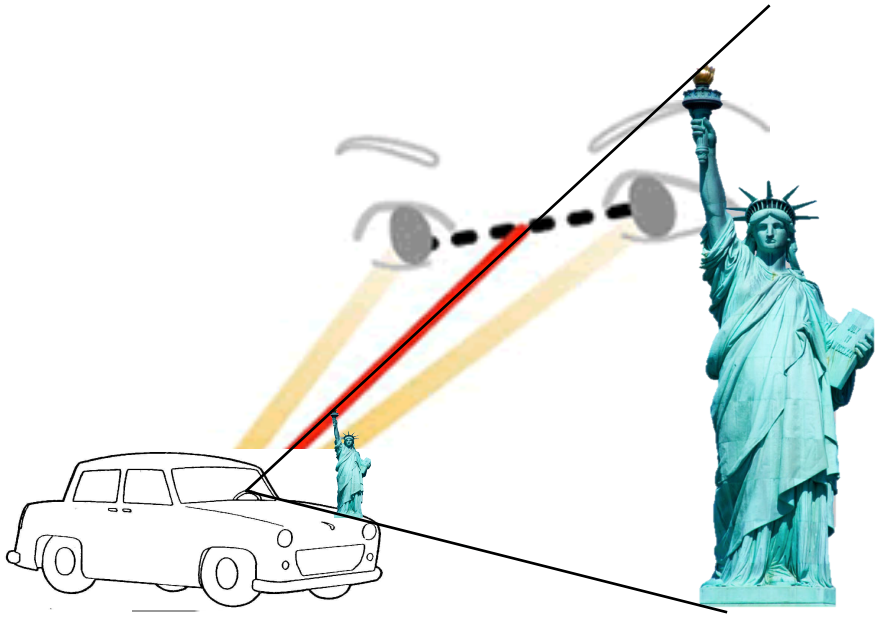
[Optical Flow Requires Multiple Strategies, ICCV 2017]

Structure from Motion

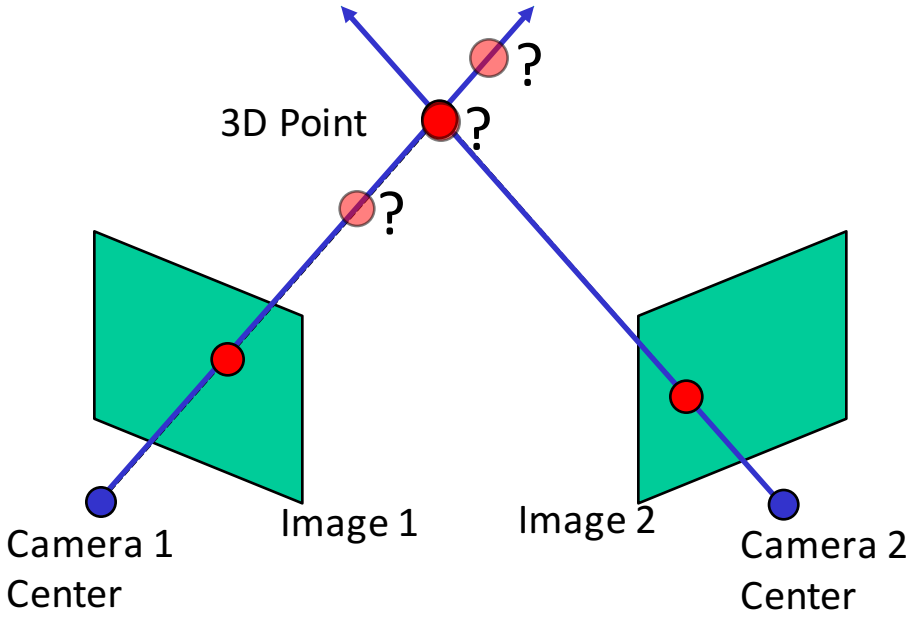
Unordered or Ordered Images



Correspondence is a vital 3D cue



Scale Ambiguity

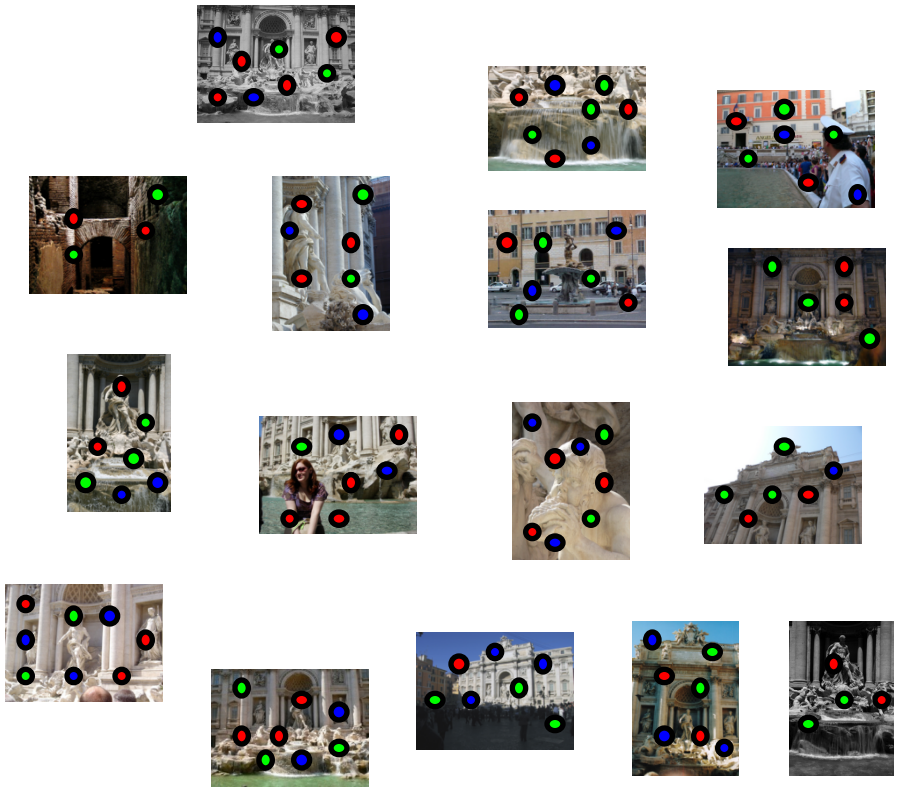


Feature detection

Several images observe a scene from different viewpoints

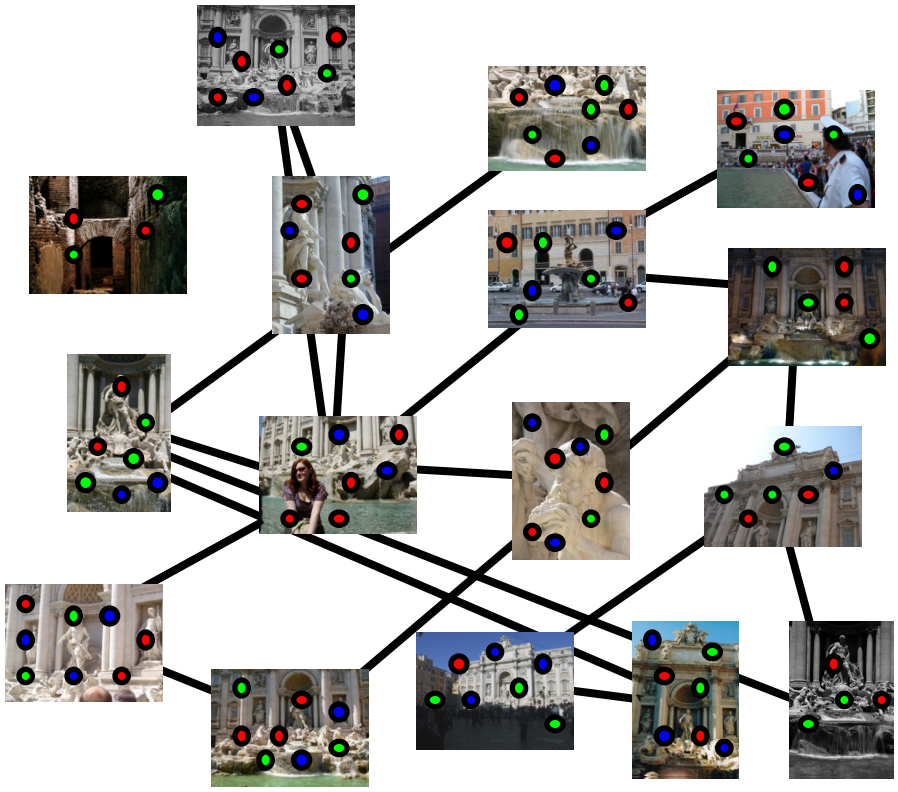
Feature detection

Detect features using, for example, SIFT [Lowe, IJCV 2004]

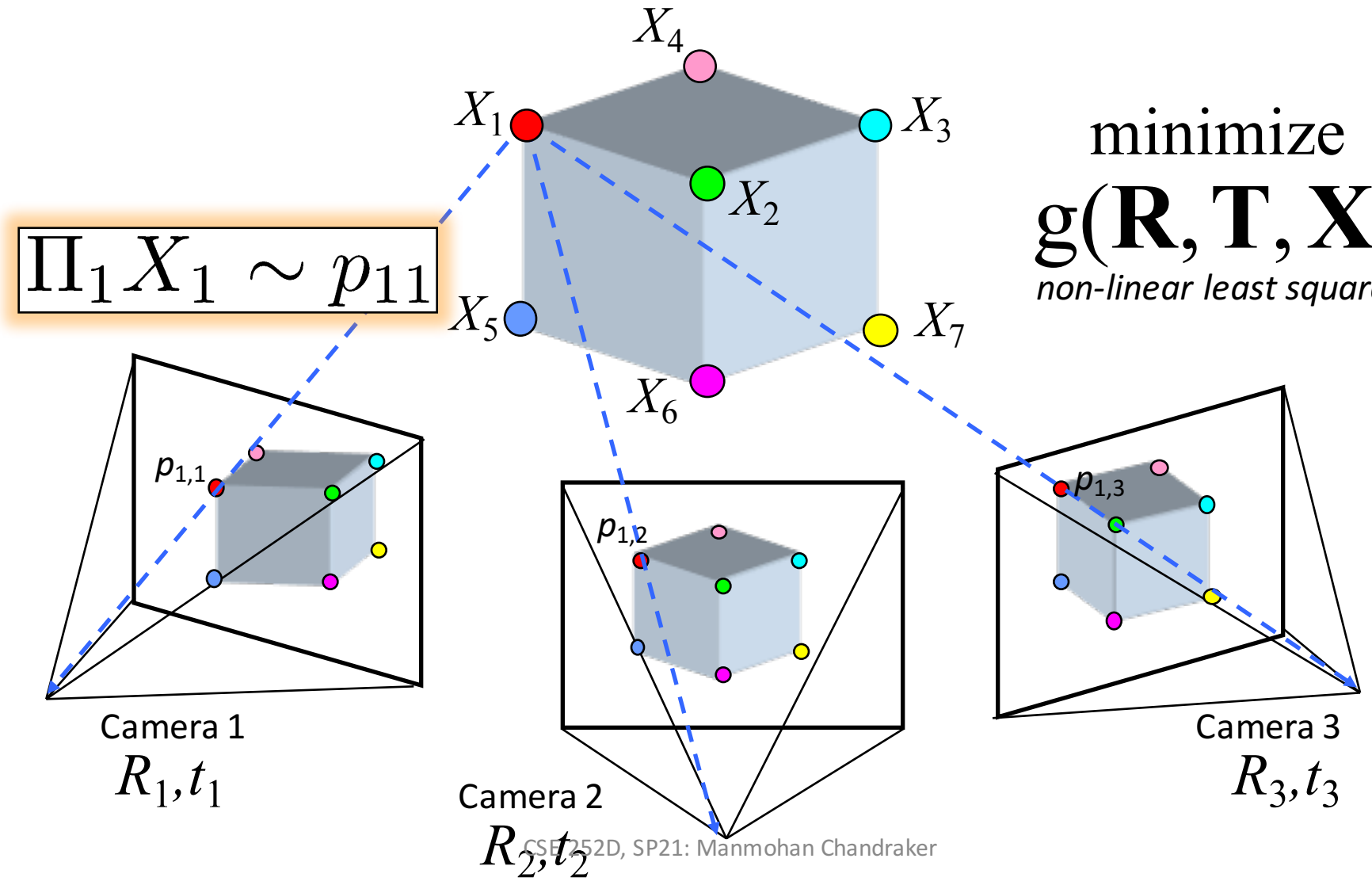


Feature matching

Match features between each pair of images



Structure from motion



Bundle adjustment

- Minimize sum of squared reprojection errors:

$$g(\mathbf{X}, \mathbf{R}, \mathbf{T}) = \sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij} \cdot \left\| \mathbf{P}(\mathbf{x}_i, \mathbf{R}_j, \mathbf{t}_j) - \begin{bmatrix} u_{i,j} \\ v_{i,j} \end{bmatrix} \right\|^2$$

indicator variable: whether point *i* visible in image *j*

predicted image location

observed image location

  - Optimized with non-linear least squares
  - Levenberg-Marquardt is a popular choice
- Practical challenges?

Bundle adjustment

- Minimize sum of squared reprojection errors:

$$g(\mathbf{X}, \mathbf{R}, \mathbf{T}) = \sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij} \cdot \left\| \mathbf{P}(\mathbf{x}_i, \mathbf{R}_j, \mathbf{t}_j) - \begin{bmatrix} u_{i,j} \\ v_{i,j} \end{bmatrix} \right\|^2$$

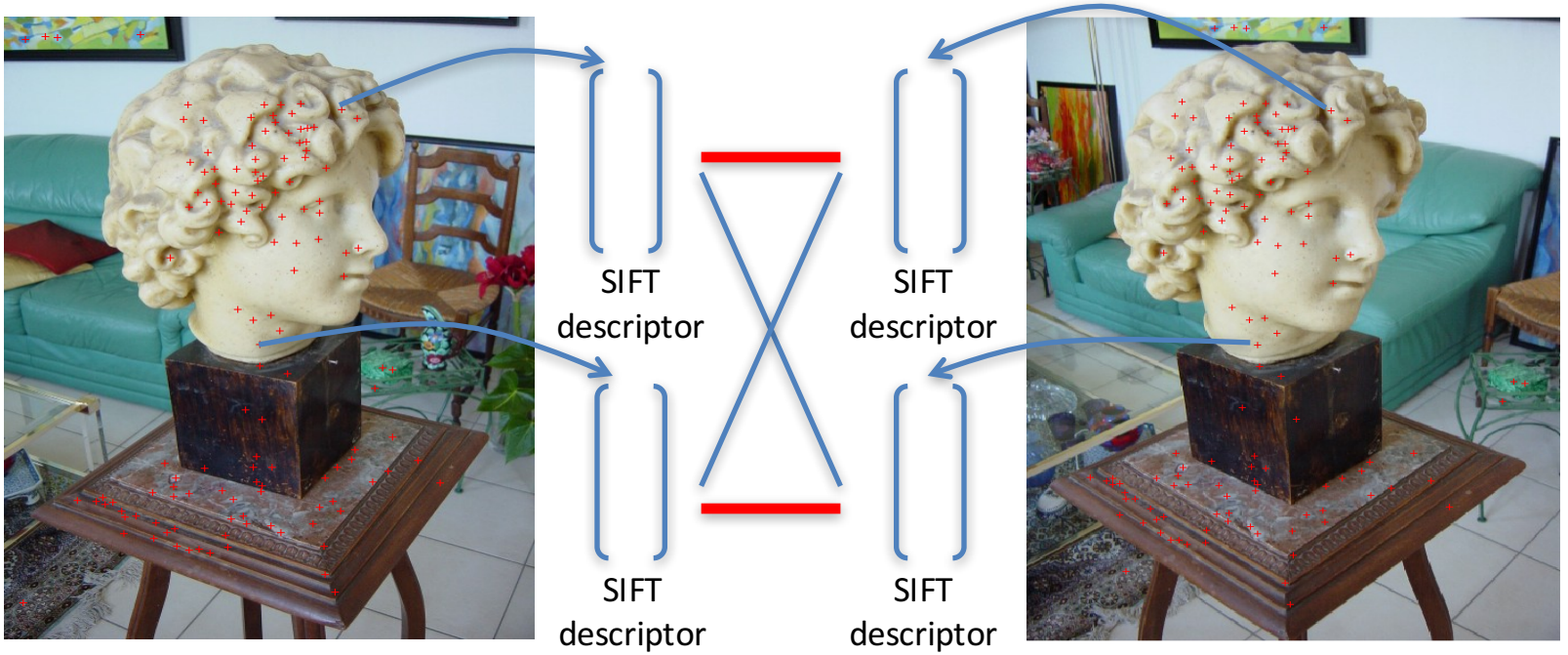
indicator variable: whether point *i* visible in image *j*

predicted image location

observed image location

  - Optimized with non-linear least squares
  - Levenberg-Marquardt is a popular choice
- Practical challenges?
  - Initialization
  - Outliers

Matching and Optimization



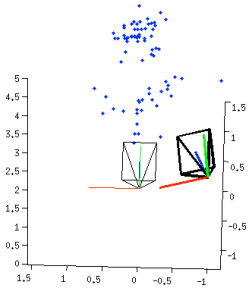
Master Plan! Repeat for all pairs?


Keypoints

Keypoints

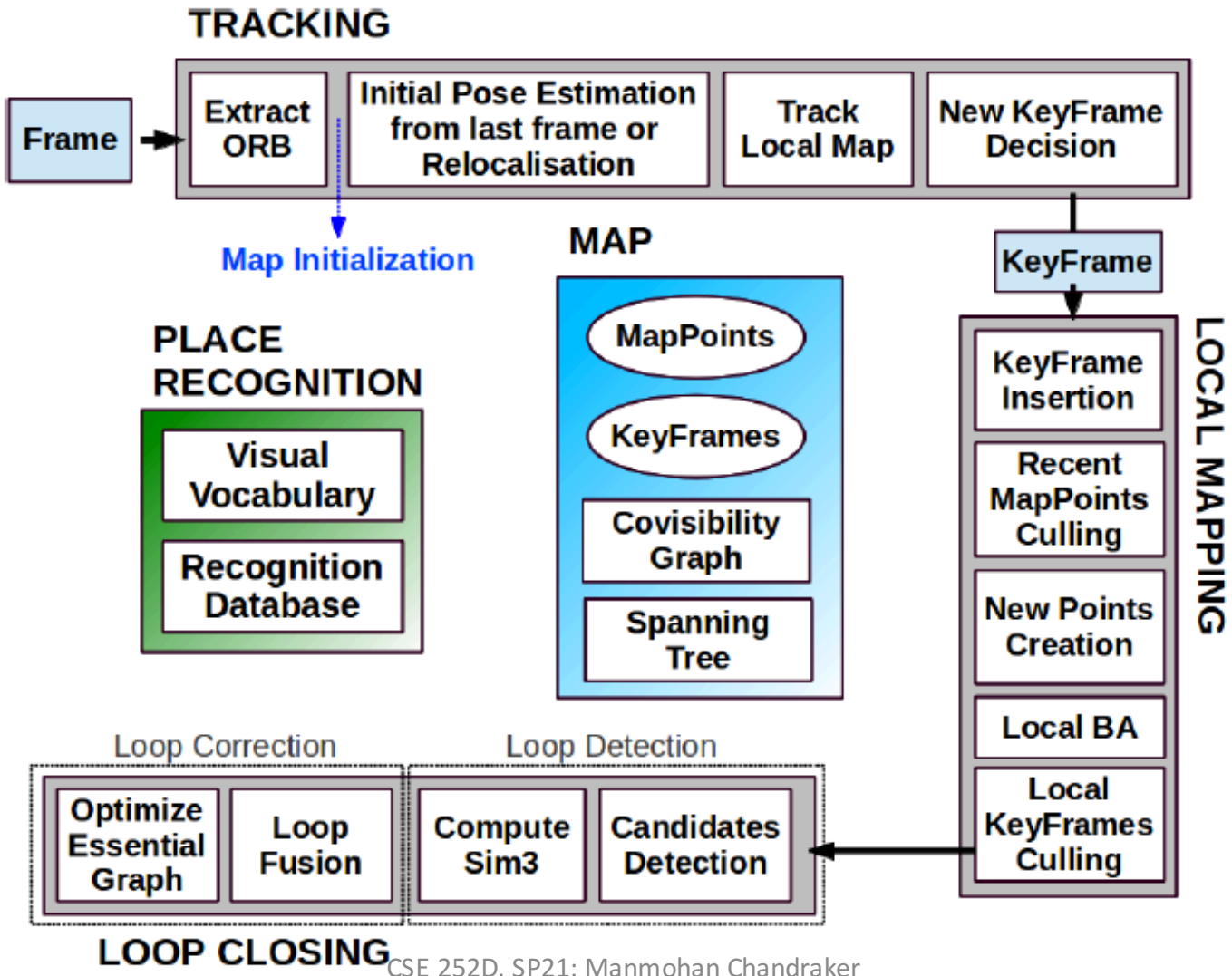
Match

Optimization for camera poses and 3D points



An Actual SFM System

- Use fundamental building blocks, but need to do a lot for robustness



[Mur-Artal et al., ORB-SLAM]

Some Recipes for SFM to Work

- Do everything you can to remove outliers

- Solve minimal problems to estimate geometric entities
  - Keeps RANSAC tractable
  - Typically, expect to spend 0.01ms

- Strategically consider what variables to optimize
  - Keyframe-based designs are successful
  - Try to robustly build long feature tracks
  - Do bundle adjustment whenever possible

- Drift is inevitable, so have a plan to address it
  - Local scale correction and global pose correction when possible

Toolkit for Practical SFM