CSE 252D: Advanced Computer Vision

Manmohan Chandraker

Lecture 3: Learning Correspondence

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

Enrollment logistics

- To enroll if you are on the waitlist
  - Send **"Request to enroll"** email to instructor if on waitlist
  - Include CV, courses, project experience relevant to computer vision
- While on the waitlist
  - You are welcome to attend lectures even if on waitlist
  - To limit TA workload, we can grade only enrolled students
  - Most should be able to enroll eventually
- Canvas
  - All enrolled and waitlisted students should have access
- All announcements will be posted on Piazza
  - Send email to TA (CC instructor) if cannot access Piazza

Course details

- Class webpage:
  - http://cseweb.ucsd.edu/~mkchandraker/classes/CSE252D/Spring2021/
- Instructor email: mkchandraker@eng.ucsd.edu
- TA: Yu-Ying Yeh (Email: yuyeh@eng.ucsd.edu)
- Grading
  - 10% presentation
  - 60% assignments
  - 30% final exam
  - Ungraded quizzes
- Aim is to learn together, discuss and have fun!

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!

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

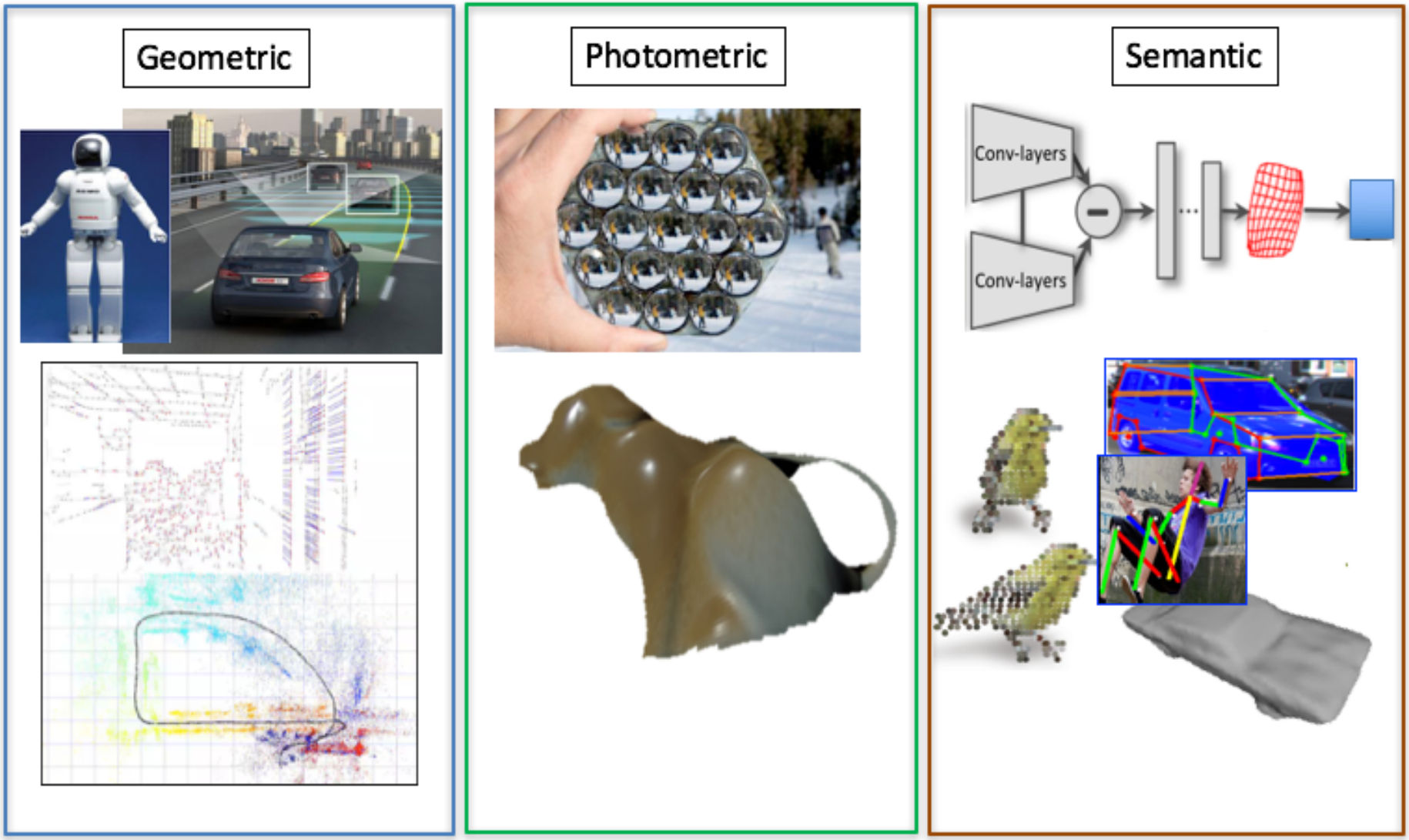
- PWC-Net: CNNs for Optical Flow Using Pyramid, Warping and Cost Volume
  - https://arxiv.org/abs/1709.02371

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

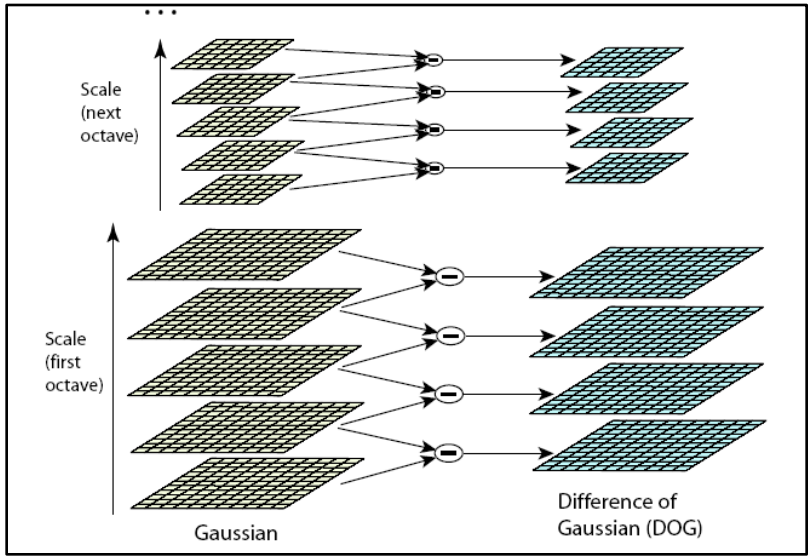
Recap

Visual correspondence aids 3D reconstruction

A key problem in 3D reconstruction: relate similar elements across a set of images

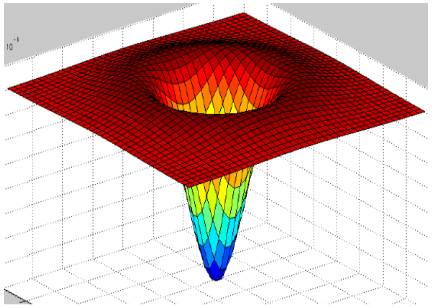


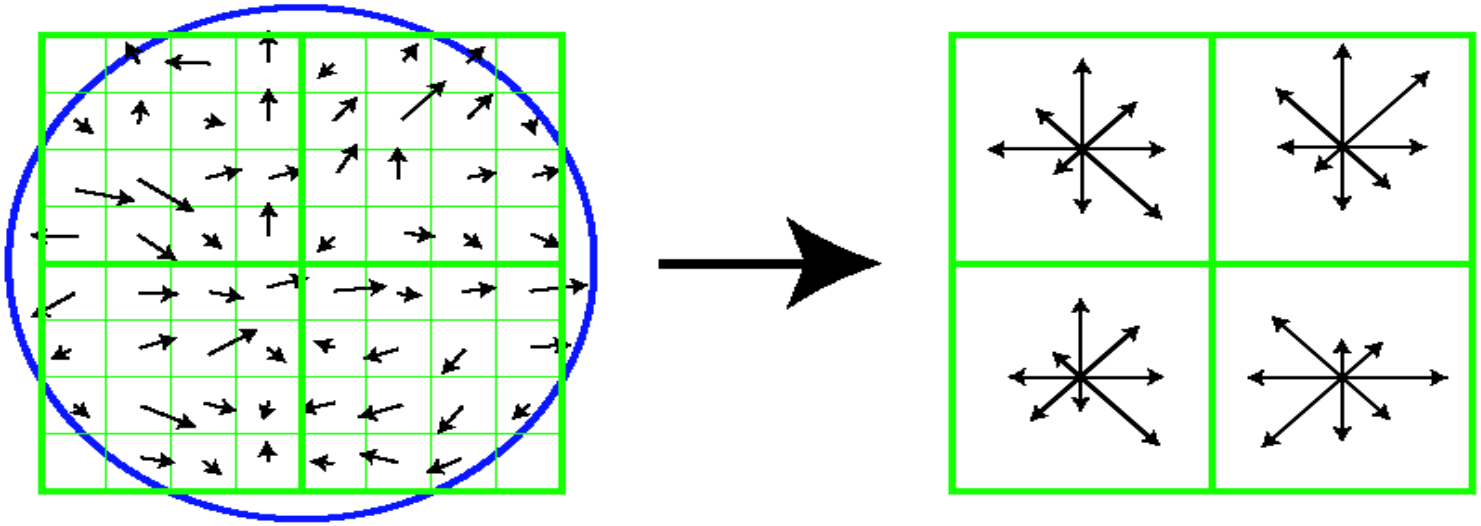
SIFT: scale space, keypoints, descriptors



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

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





Correspondence using a patch similarity CNN

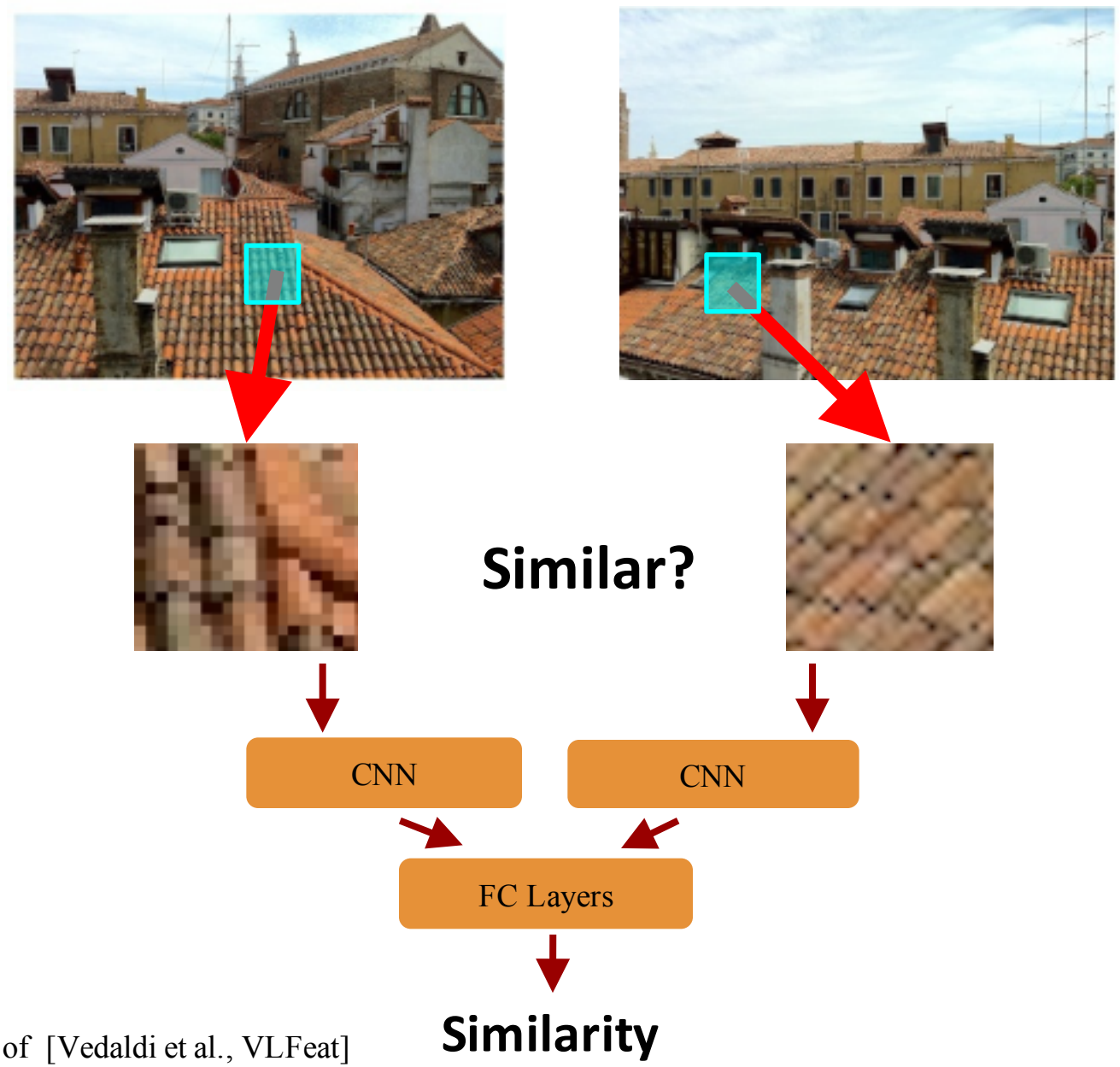


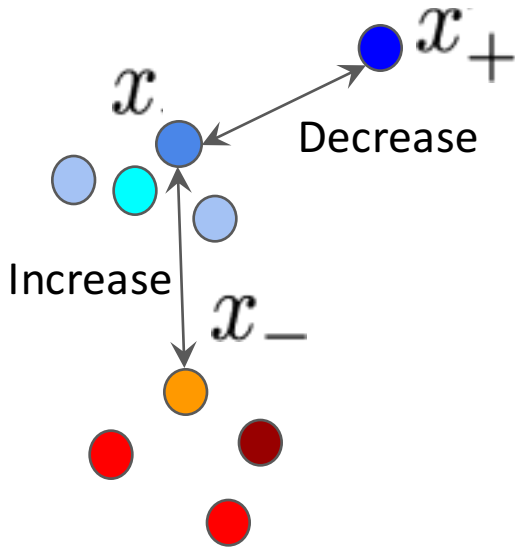
Image courtesy of [Vedaldi et al., VLFeat]

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
  - Metric Space: Distance relationship = Class membership

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

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

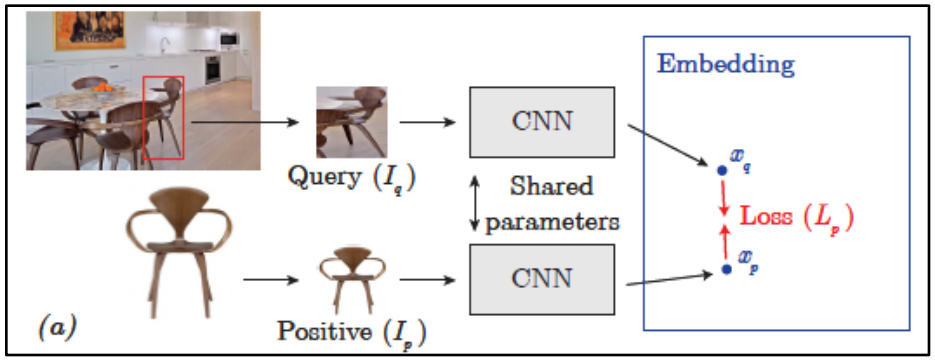


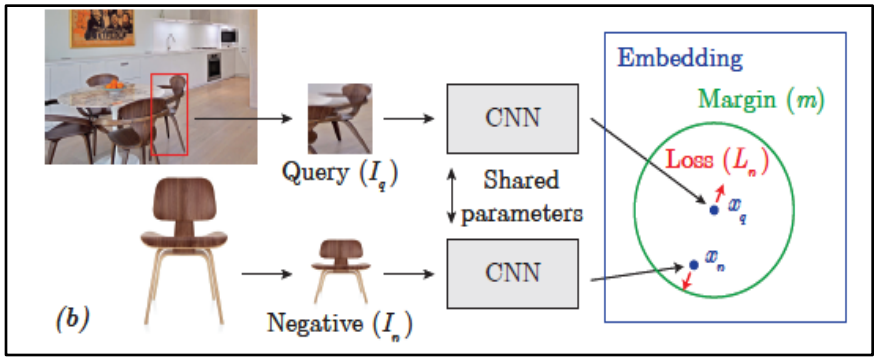
Contrastive Loss for Metric Learning

- For pair of training examples x_1 and 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

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

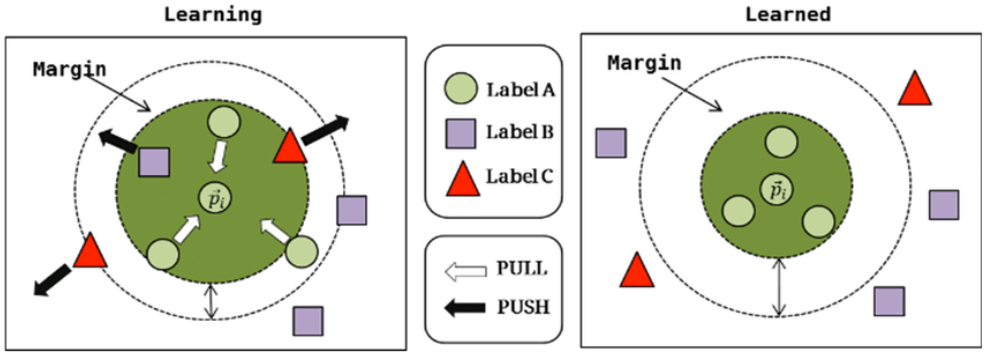
Anchor Positive Negative

- 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:

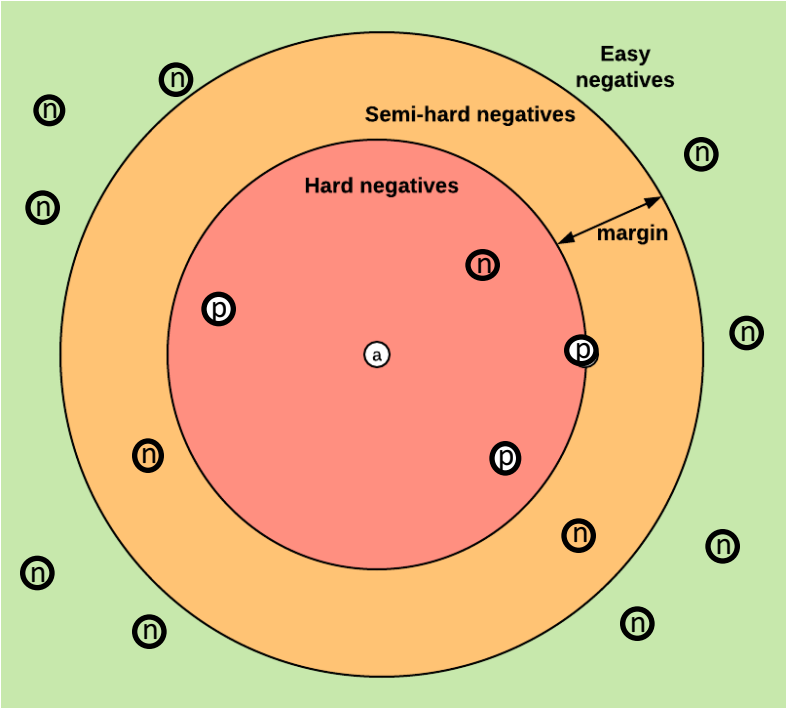
$$l_{tri}(\mathcal{T}) = \left[\|\mathbf{x}_a - \mathbf{x}_p\|^2 - \|\mathbf{x}_a - \mathbf{x}_n\|^2 + m\right]_+$$



Hard Negative Mining

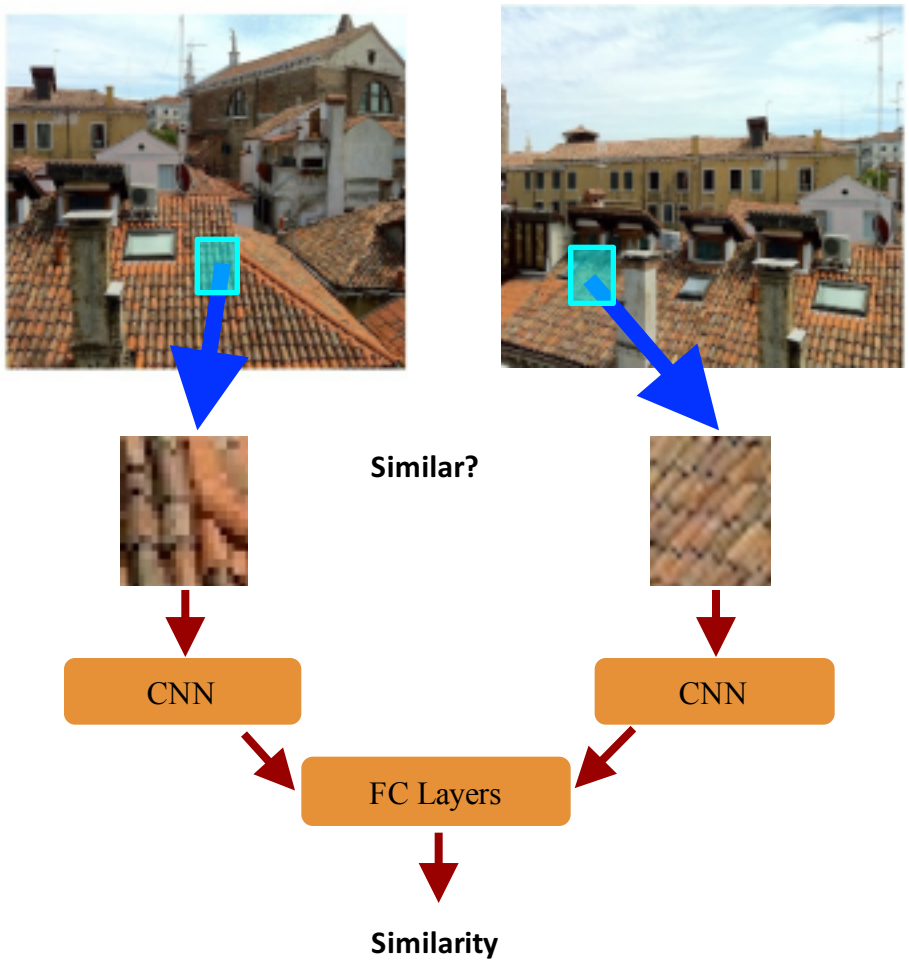
- Triplet loss: $l_{tri}(\mathcal{T}) = \left[\|\mathbf{x}_a - \mathbf{x}_p\|^2 - \|\mathbf{x}_a - \mathbf{x}_n\|^2 + m\right]_+$

Loss is 0 for easy negatives



- Metric learning is driven by hard negatives
- Most negatives are easy
- Need strategy to find hard negatives among samples

Correspondence beyond similarity CNN

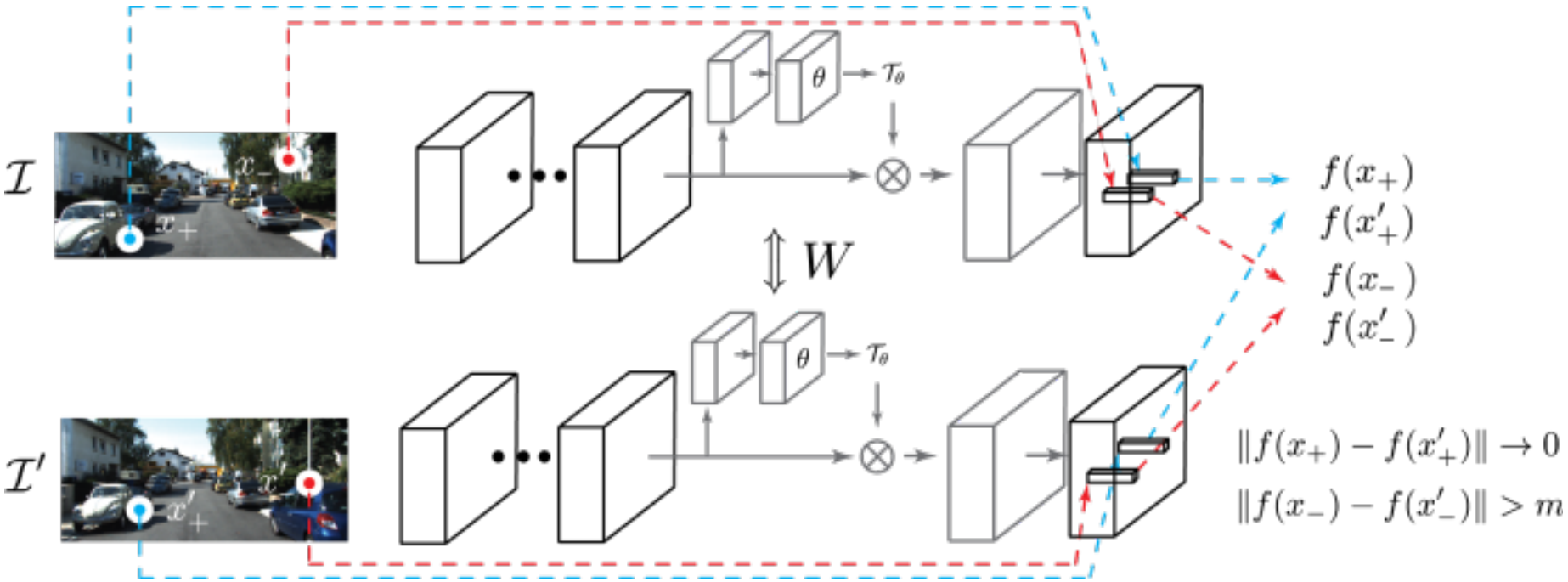


- Detection of interest points
- Normalization of patches
- Multiscale information
- Obtaining training data
- Efficient training and testing

Universal Correspondence Network

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

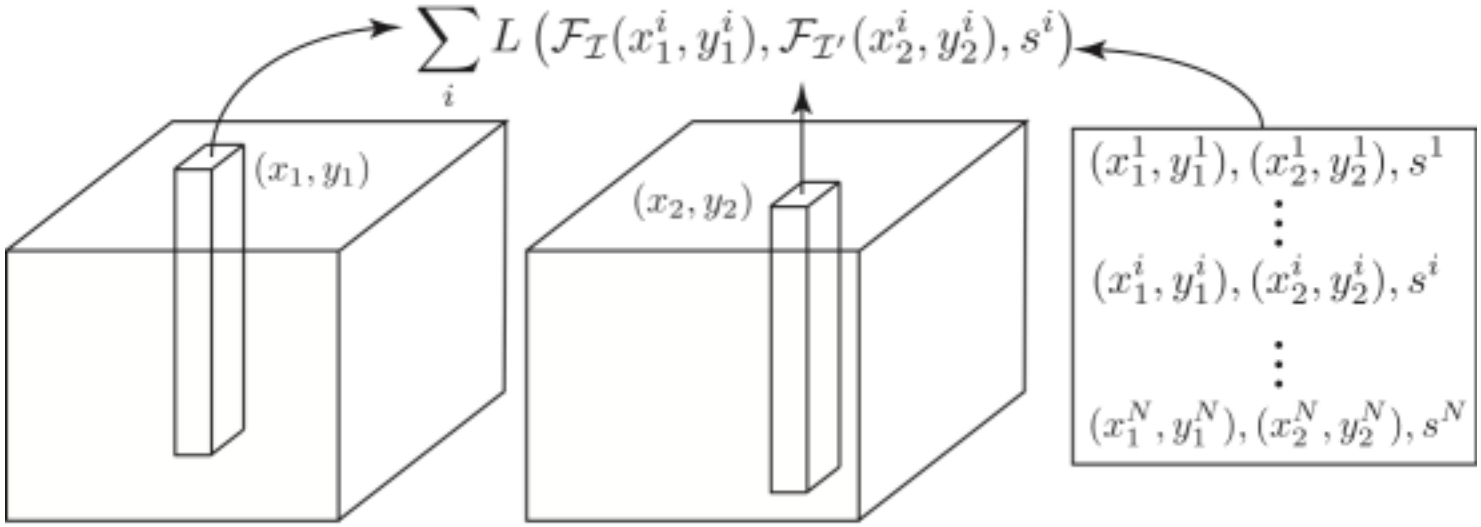


Correspondence Contrastive Loss

- Metric learning loss is defined for features
- Supervision is in the form of coordinates of correspondences

$(x_1, y_1), (x_2, y_2), s$ → Corresponding pair: s = 1
Any other pair: s = 0

- Bilinear interpolation to extract smoothly sampled features



$$L = \frac{1}{2N} \sum_i^N s_i \|\mathcal{F}_{\mathcal{I}}(\mathbf{x}_i) - \mathcal{F}_{\mathcal{I}'}(\mathbf{x}_i')\|^2 + (1 - s_i) \max(0, m - \|\mathcal{F}_{\mathcal{I}}(\mathbf{x}) - \mathcal{F}_{\mathcal{I}'}(\mathbf{x}_i')\|)^2$$

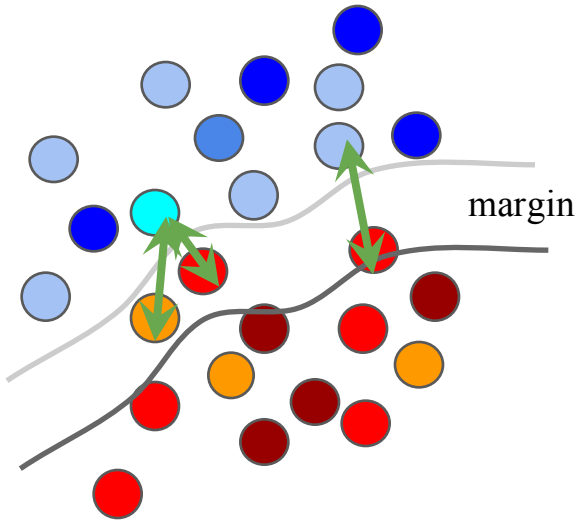
Hard Negative Mining

- Correspondence contrastive loss:

$$L = \frac{1}{2N} \sum_{i}^{N} s_i \|\mathcal{F}_{\mathcal{I}}(\mathbf{x}_i) - \mathcal{F}_{\mathcal{I}'}(\mathbf{x}_i')\|^2 + (1 - s_i) \max(0, m - \|\mathcal{F}_{\mathcal{I}}(\mathbf{x}) - \mathcal{F}_{\mathcal{I}'}(\mathbf{x}_i')\|)^2$$

Pushes negatives at least margin *m* away
Only active when negative pair within *m* of each other (in feature space)

- Metric learning driven by hard negatives.



Hard Negative Mining

- Correspondence contrastive loss:

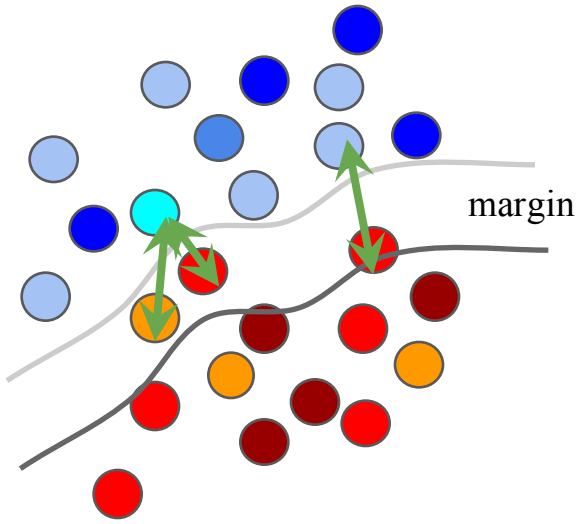
$$L = \frac{1}{2N} \sum_{i}^{N} s_i \|\mathcal{F}_{\mathcal{I}}(\mathbf{x}_i) - \mathcal{F}_{\mathcal{I}'}(\mathbf{x}_i')\|^2 + (1 - s_i) \max(0, m - \|\mathcal{F}_{\mathcal{I}}(\mathbf{x}) - \mathcal{F}_{\mathcal{I}'}(\mathbf{x}_i')\|)^2$$

Pushes negatives at least margin *m* away
Only active when negative pair within *m* of each other (in feature space)

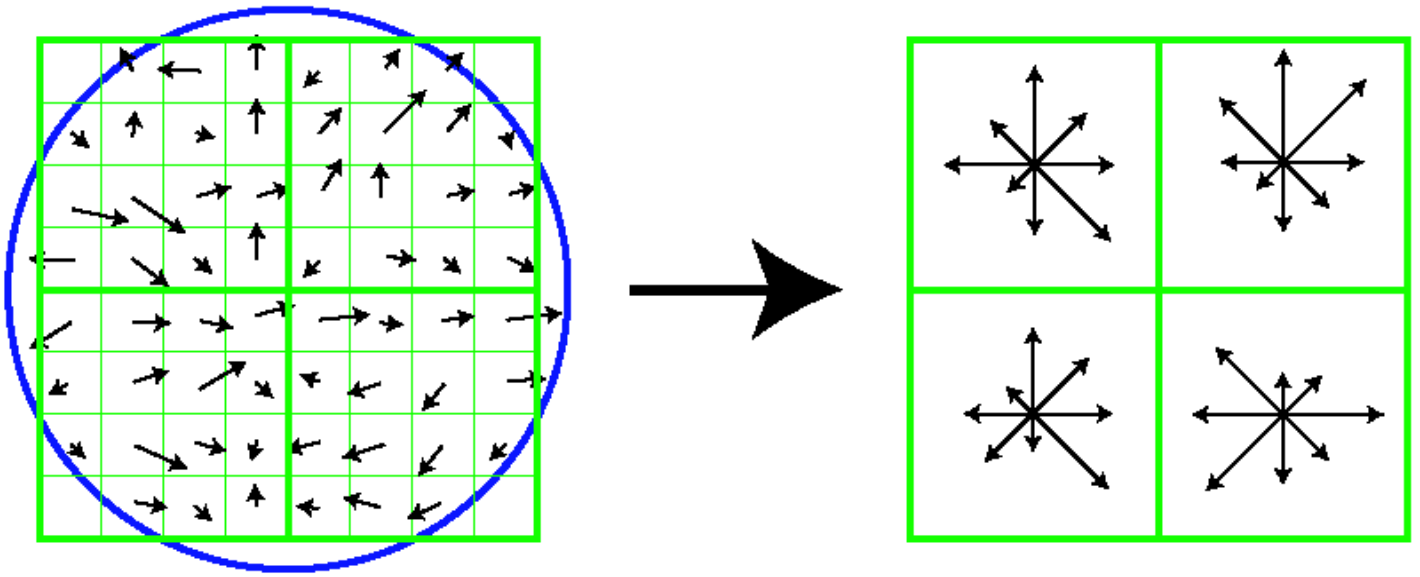
- Metric learning driven by hard negatives.

Active hard negative mining

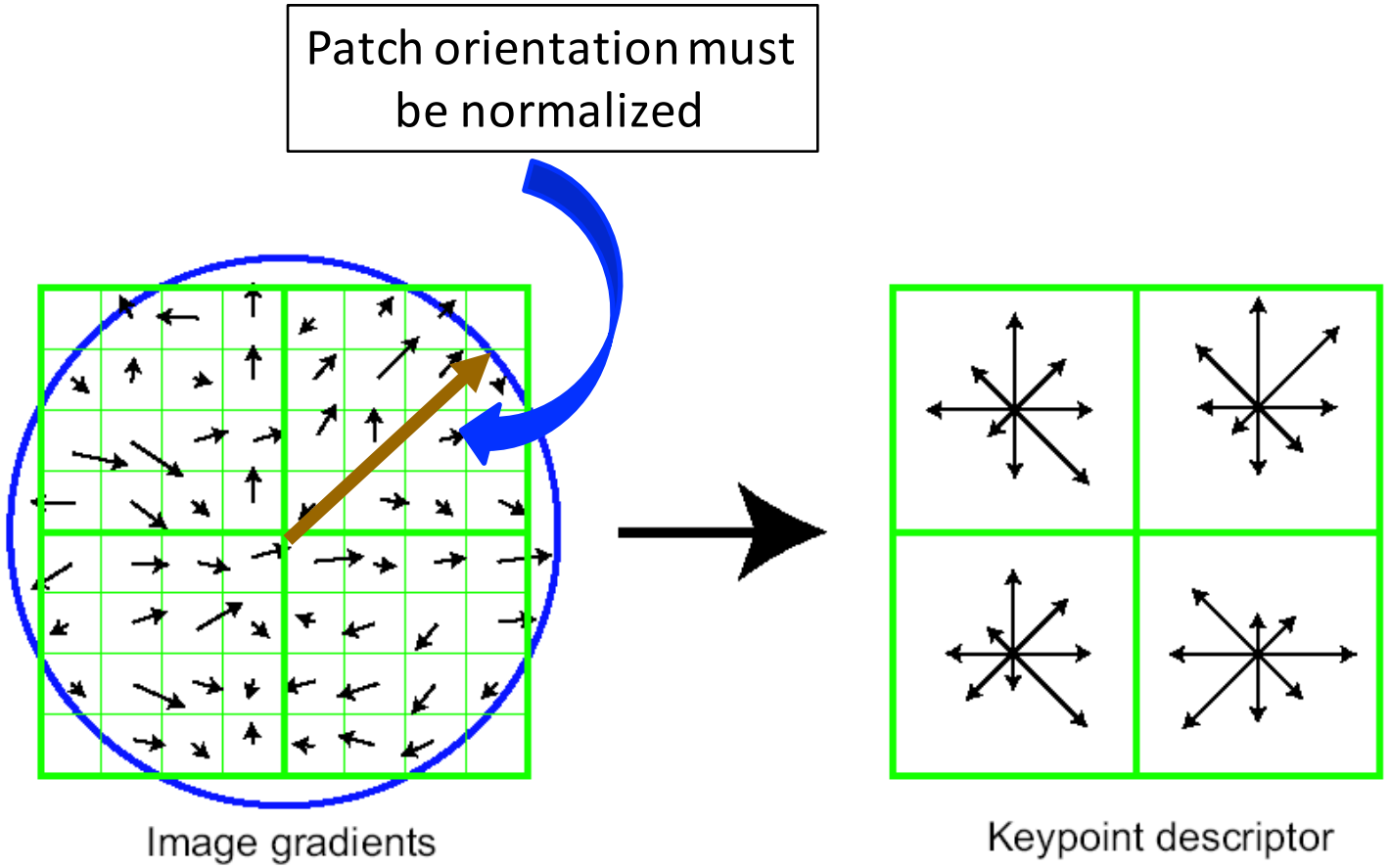
- Consider feature for a point in image 1
- Find closest point in feature space in image 2
- Hard negative if image 2 point far from ground truth correspondence
- Thousands of hard negatives per image pair
- Faster convergence to better local optimum



Something Missing Compared to SIFT

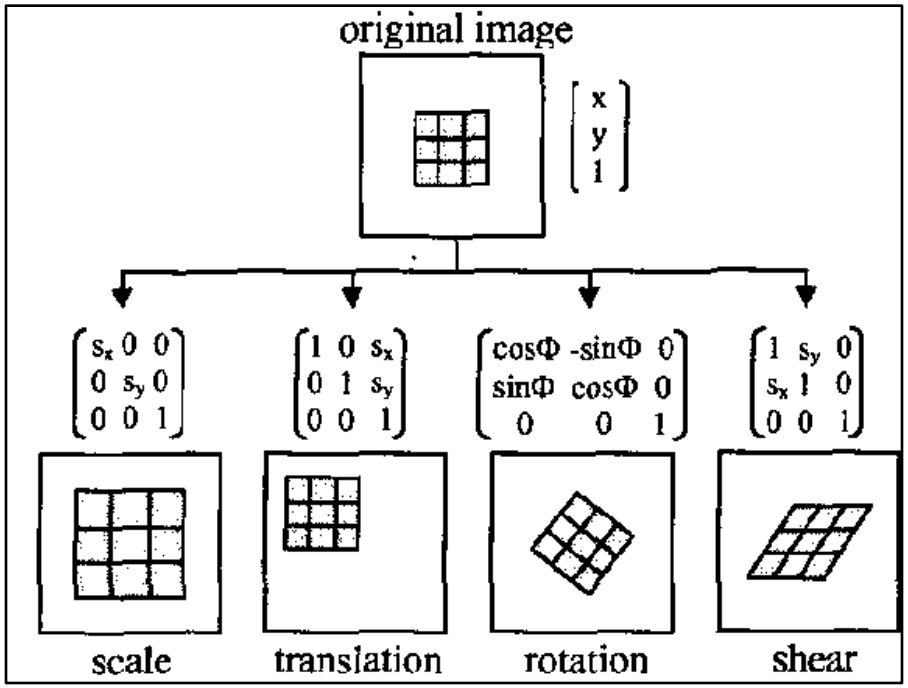


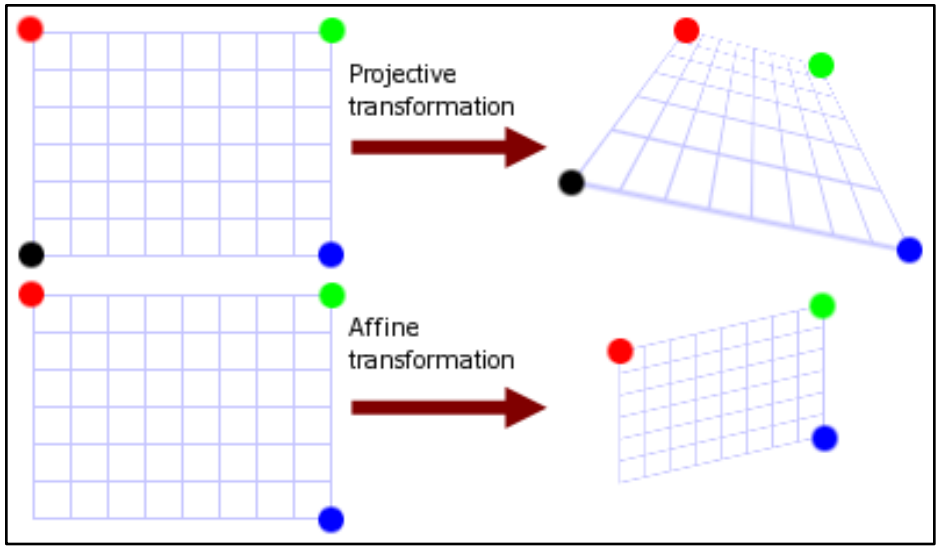
Something Missing Compared to SIFT



Spatial Transformer Networks

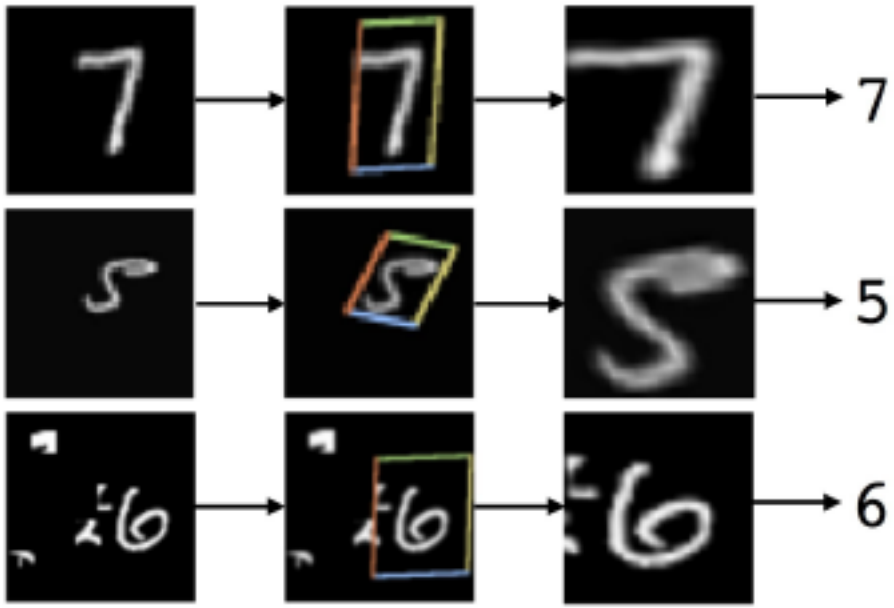
Image Coordinate Transformations





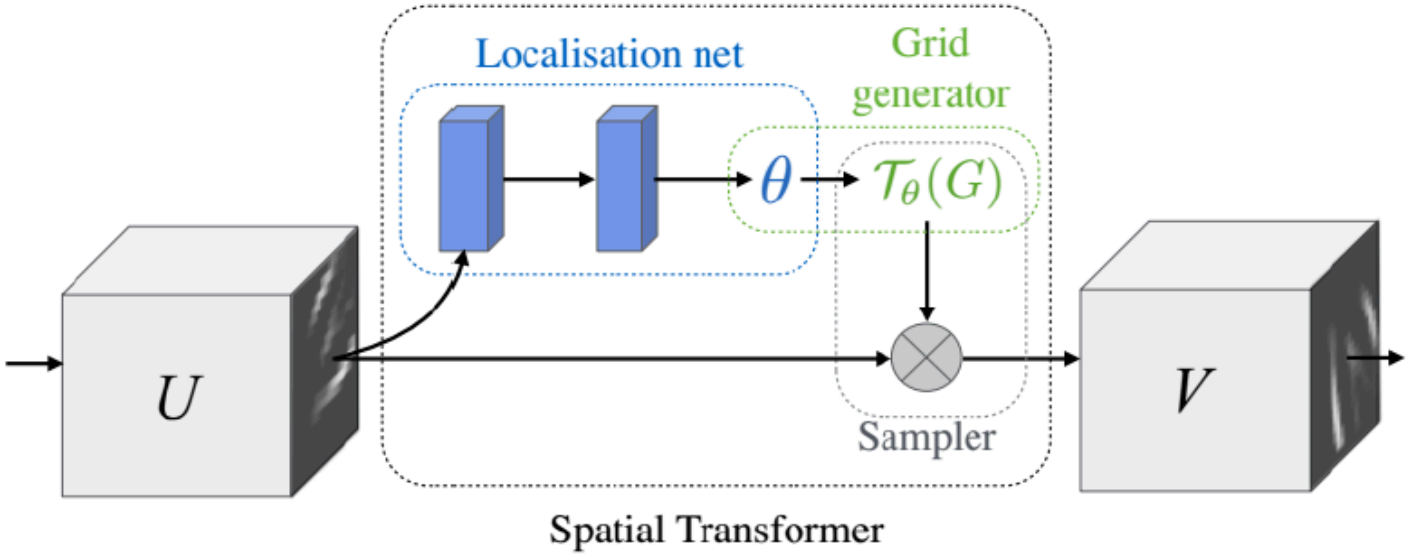
Spatial Transformers

- Transform an image or feature map by learning transformation matrix
- Translation, rotation, scale, cropping, affine, thin-plate splines, ….
- Differentiable module trainable with backpropagation



Need a network module to act on coordinates of input.

Spatial Transformers



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

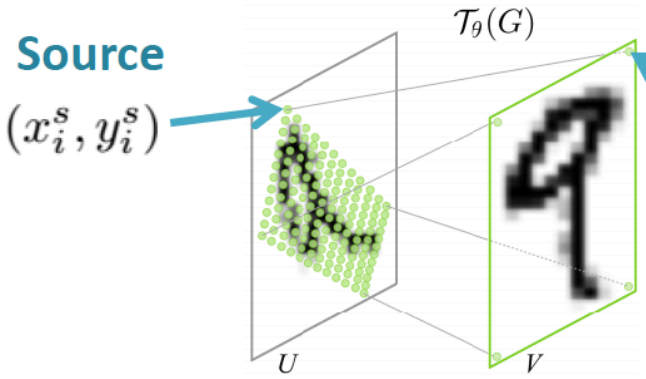
- **Grid generator** takes uniform grid in output map and deforms it by transformation
- This determines a deformed grid to sample the input feature map U

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

Spatial Transformers

Given transformation θ, we wish to populate regular grid (image or matrix) of output V. The values are to be mapped from the input U, where data is also on a regular grid.

Example: Affine transformation



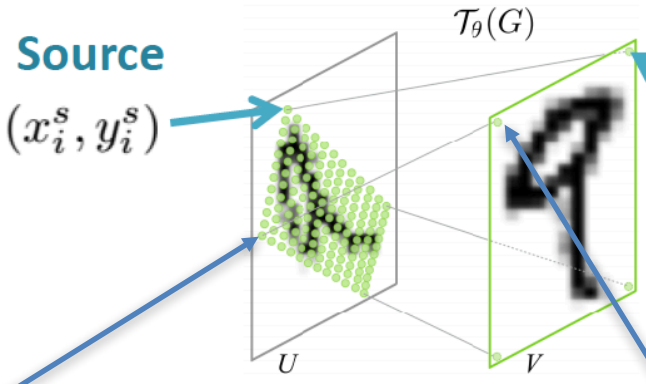
Output pixels are defined to lie on a regular grid.

$$\begin{pmatrix} x_i^s \\ y_i^s \end{pmatrix} = \mathcal{T}_\theta(G_i) = \mathtt{A}_\theta \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix} = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix} \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix}$$

Spatial Transformers

Given transformation θ, we wish to populate regular grid (image or matrix) of output V. The values are to be mapped from the input U, where data is also on a regular grid.

Example: Affine transformation



Output pixels are defined to lie on a regular grid.

Sampling Grid

These locations can be fractional, such as:
(1.1, 1.2), (0.5, 2.9), ..., (11.0, 5.7), (11.8, 5.1), ...

These are integer locations, such as:
(0,0), (0,1), ..., (11,4), (11,5), ...

$$\begin{pmatrix} x_i^s \\ y_i^s \end{pmatrix} = \mathcal{T}_\theta(G_i) = \mathtt{A}_\theta \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix} = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix} \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix}$$

Localization net predicts 6 real numbers, arranged into 2x3 matrix, for affine case

Spatial Transformers

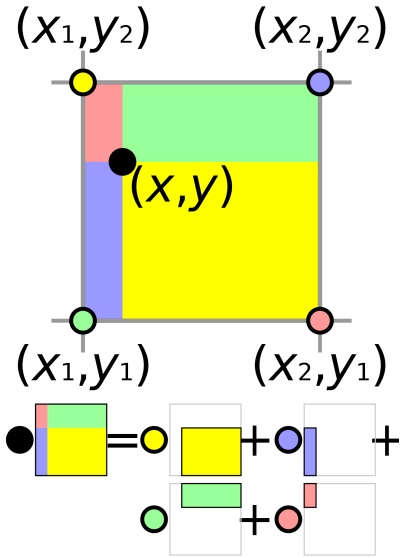
Differentiable image sampling (bilinear): determine interpolated value at U(x^s, y^s), by interpolating from neighbors of (x^s, y^s) where value of U has been recorded

$$V_i^c = \sum_n^H \sum_m^W U_{nm}^c \max(0, 1 - |x_i^s - m|) \max(0, 1 - |y_i^s - n|)$$

target value

source value

sampling grid coordinate

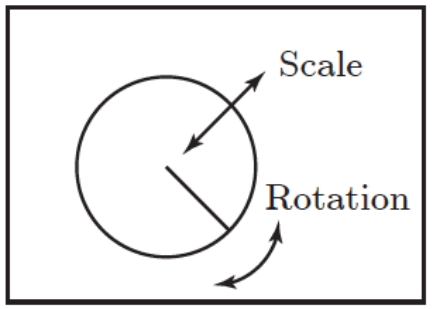


Backpropagation possible through derivatives of V with respect to U and grid:

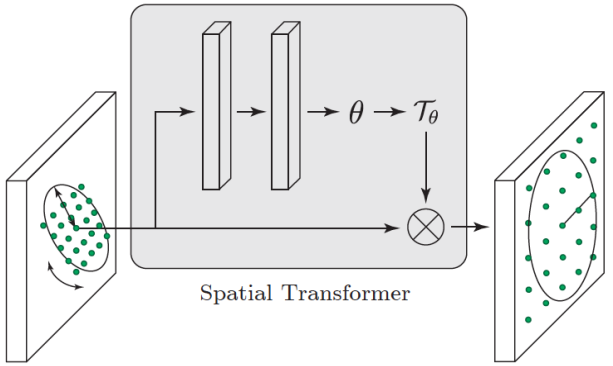
$$\frac{\partial V_i^c}{\partial U_{nm}^c} = \sum_n^H \sum_m^W \max(0, 1 - |x_i^s - m|) \max(0, 1 - |y_i^s - n|)$$

$$\frac{\partial V_i^c}{\partial x_i^s} = \sum_n^H \sum_m^W U_{nm}^c \max(0, 1 - |y_i^s - n|) \begin{cases} 0 & \text{if } |m - x_i^s| \geq 1 \\ 1 & \text{if } m \geq x_i^s \\ -1 & \text{if } m < x_i^s \end{cases}$$

Achieving Effect of Patch Normalization in UCN

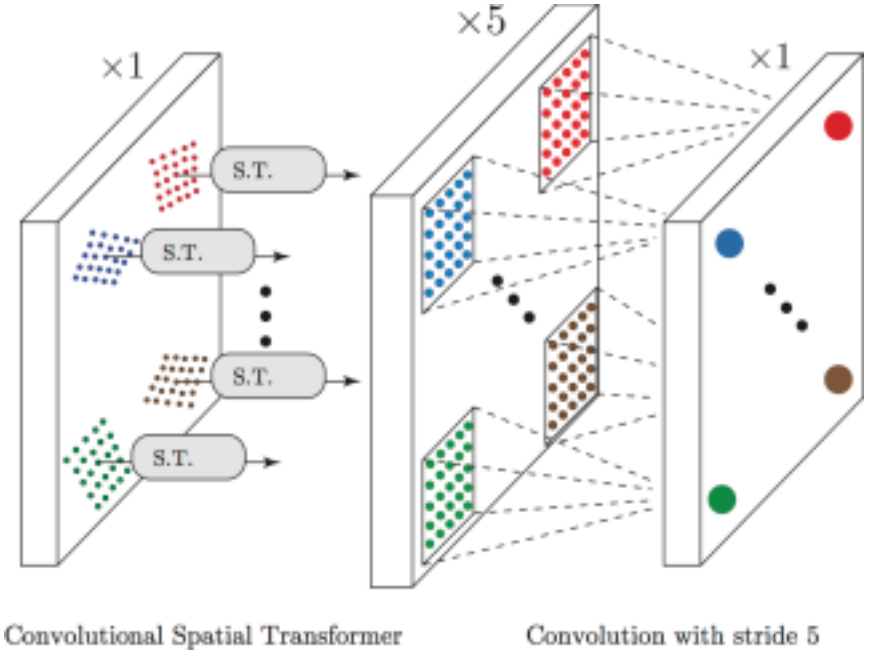


SIFT normalizes for scale and rotation



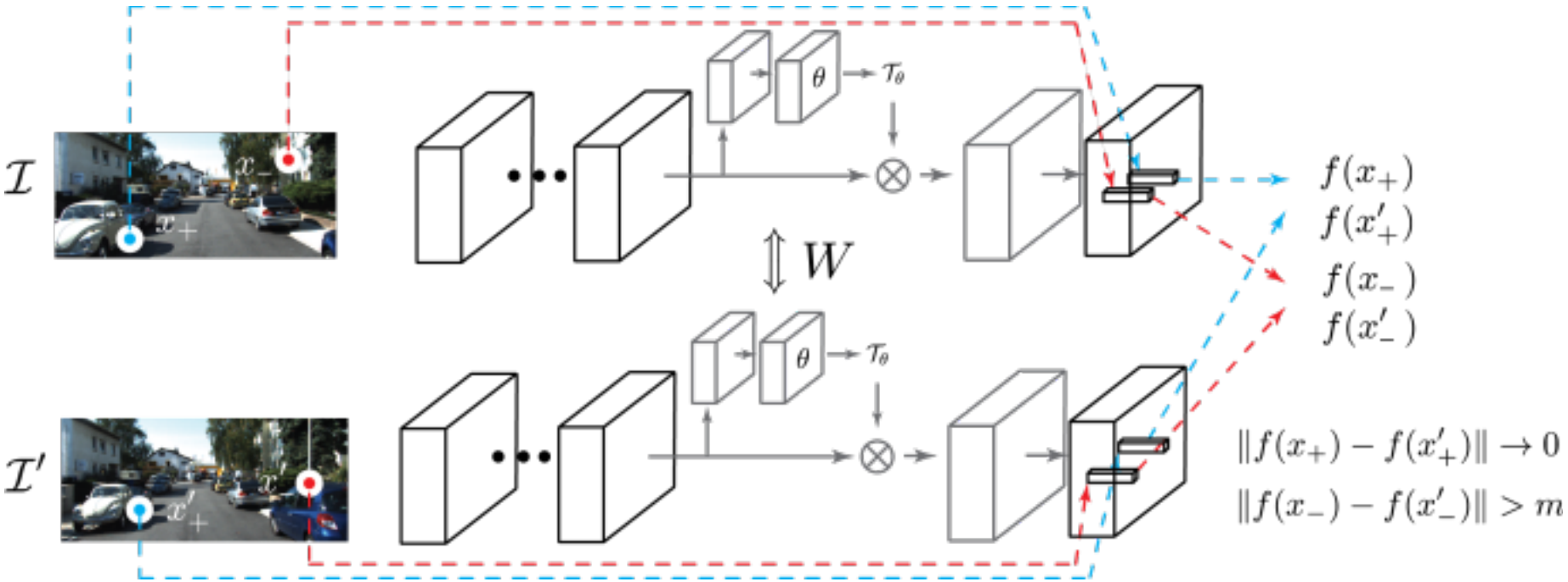
Spatial transformer for global transformation

Convolutional spatial transformer for independent per-pixel normalization

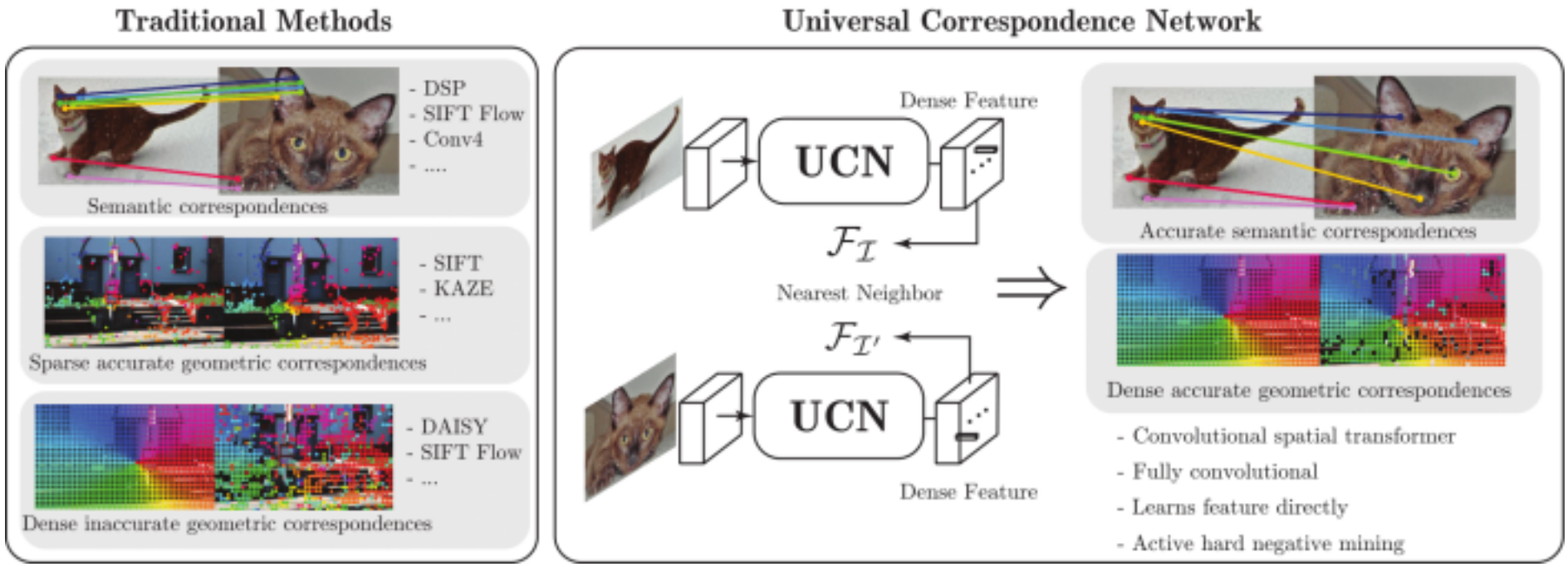


UCN Architecture

- 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



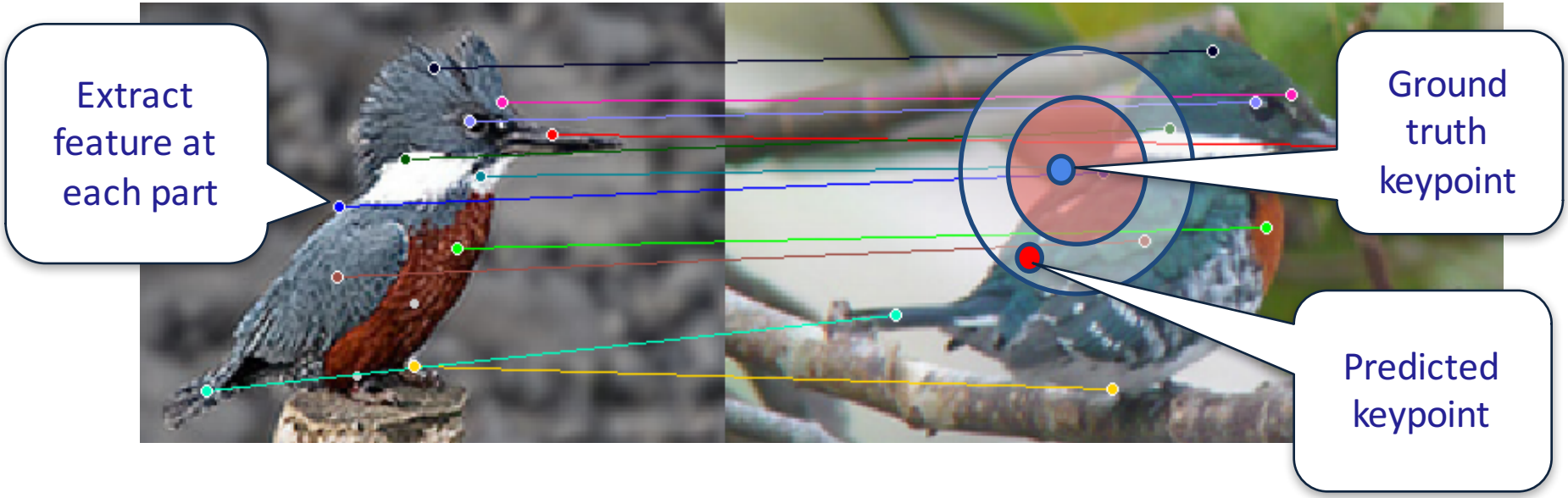
Universal Correspondence Network



Features	Dense	Geometric Corr.	Semantic Corr.	Trainable	Efficient
SIFT [22]	✗	✓	✗	✗	✓
DAISY [29]	✓	✓	✗	✗	✓
Conv4 [21]	✓	✗	✓	✓	✓
DeepMatching [25]	✓	✓	✗	✗	✗
Patch-CNN [34]	✓	✓	✗	✓	✗
Ours	✓	✓	✓	✓	✓

Quantitative metric for matching

Evaluation metric: Percentage of Correct Keypoints (PCK)



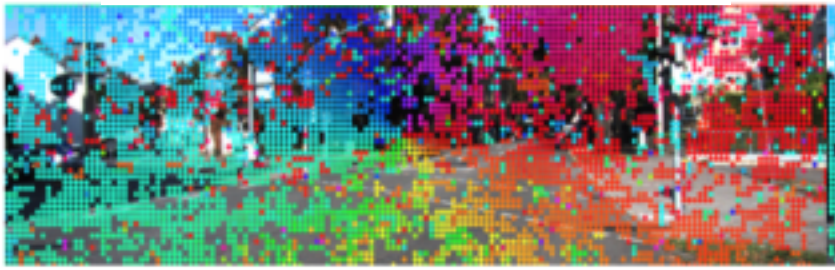
2. Find the pixel  with the most similar feature in the other image.

3. Match is **correct** if  is within *k* pixels of ground-truth  .

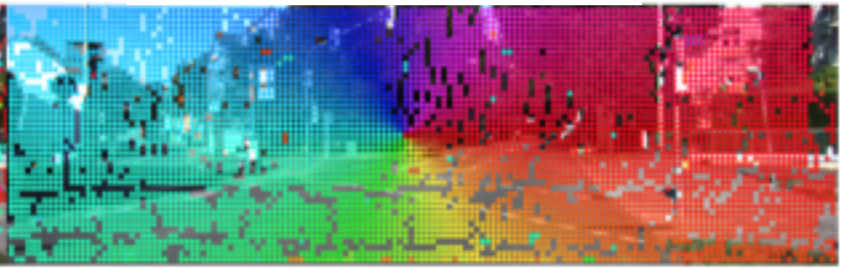
Geometric correspondence: KITTI

Original images and keypoints

SIFT matches



DAISY matches



UCN matches

	SIFT	DAISY	UCN
PCK at 10 pixels	48.9	79.6	86.5

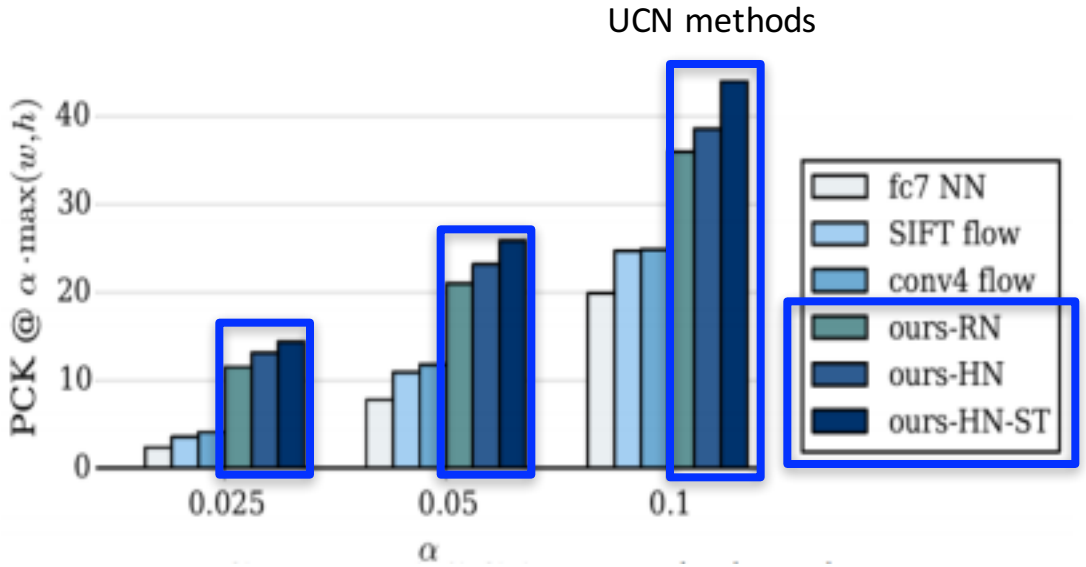
Correspondence

	SIFT	DAISY	UCN
Rotation deviation (deg)	0.307	0.309	0.317
Translation deviation (deg)	4.749	4.516	4.147

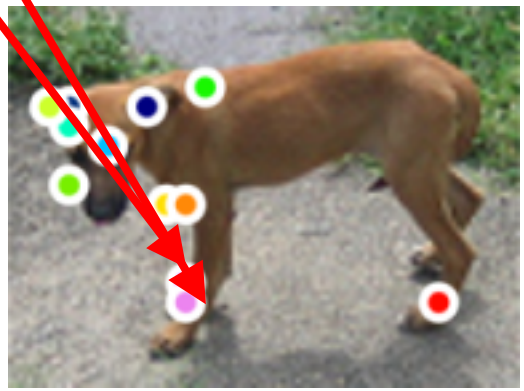
Rotation and translation estimation

Semantic correspondence: PASCAL

Query Features



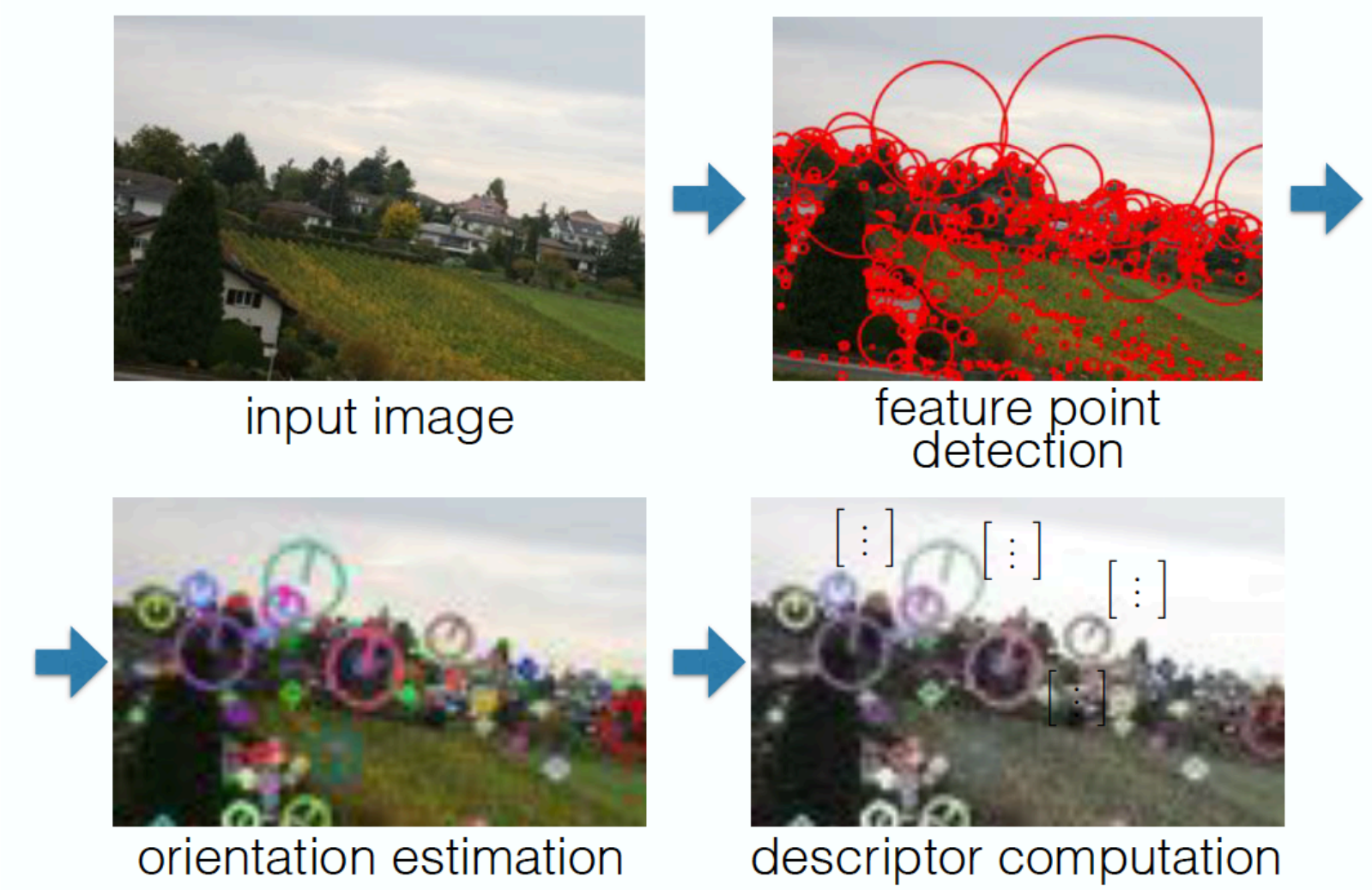
Ground truth



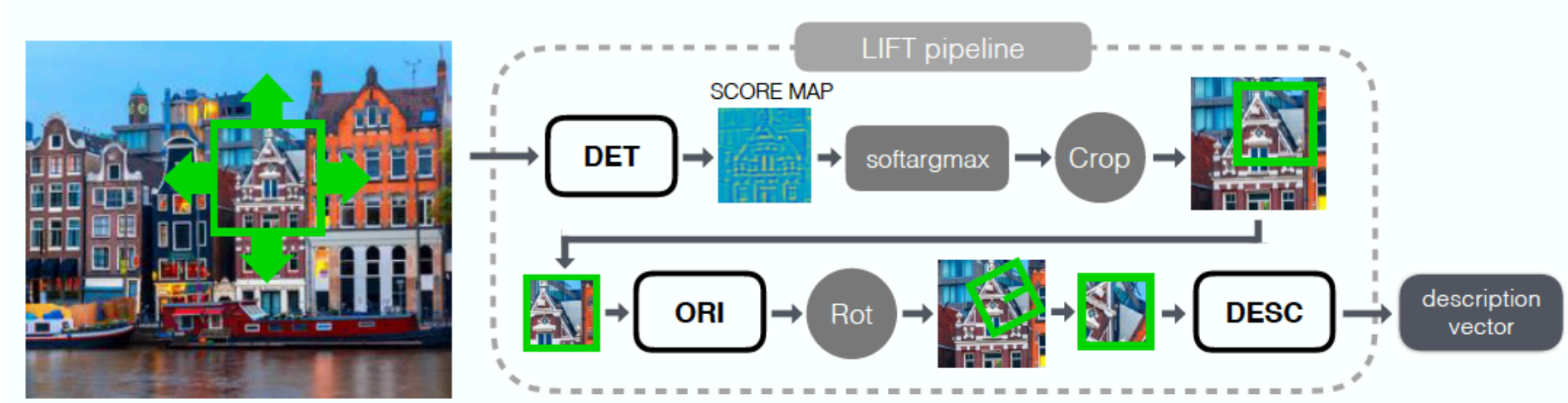
UCN

VGG Conv4

Keypoint detection and description



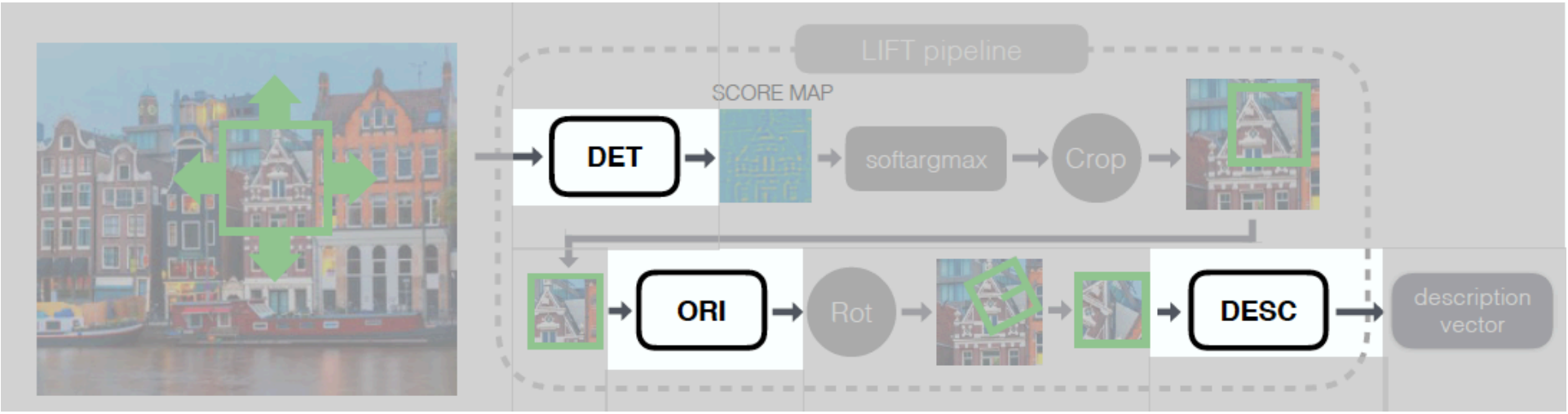
Keypoint detection and description



- 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

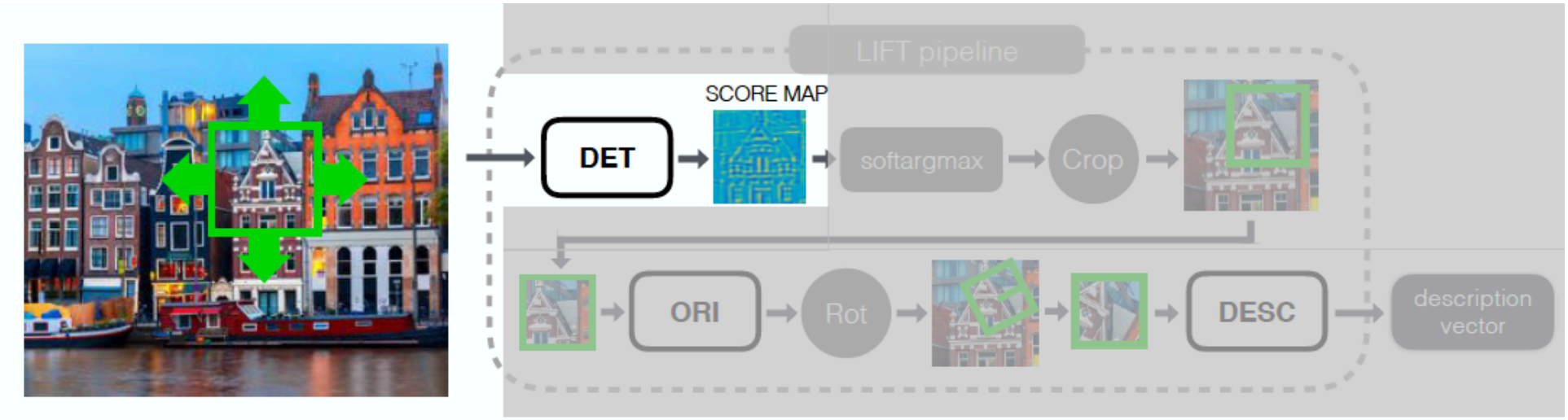
[LIFT: Learned Invariant Feature Transform, ECCV 2016]

Keypoint detection and description



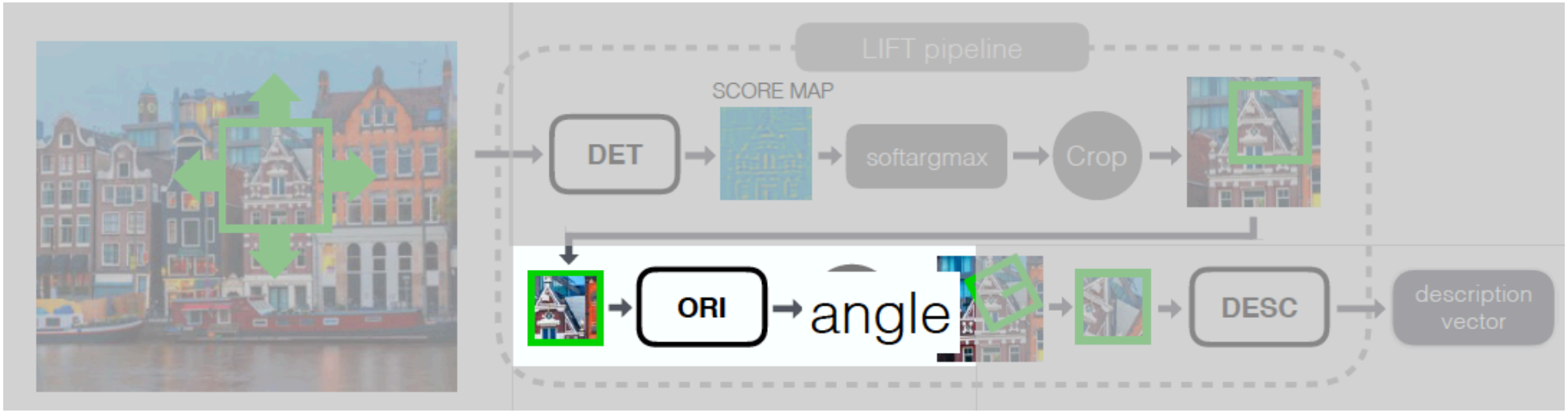
Keypoint detection, orientation estimation and feature descriptor are all learned.

Detection network



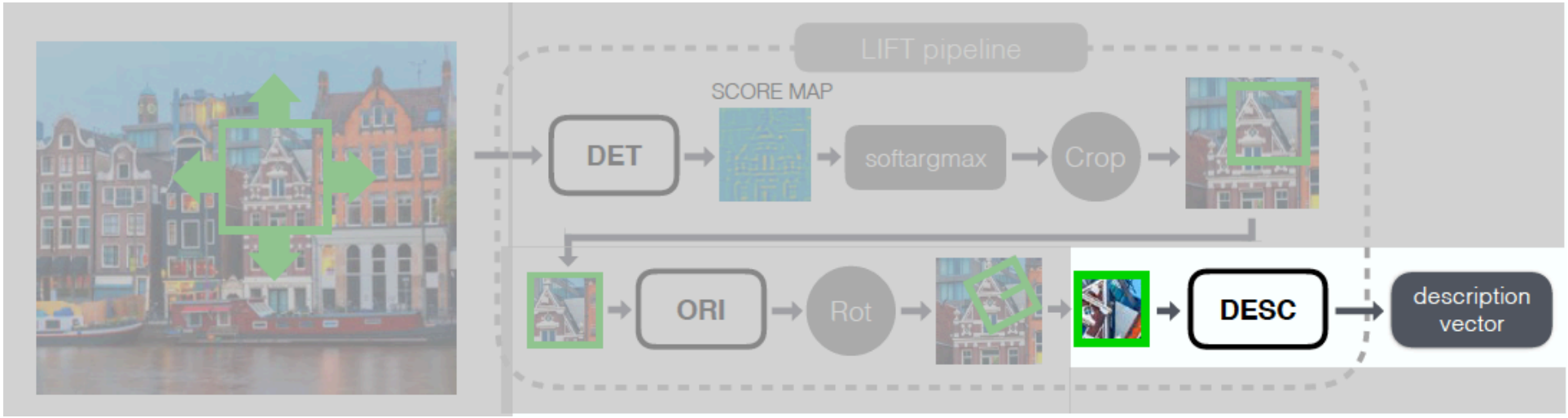
Detector produces a score map.

Orientation network



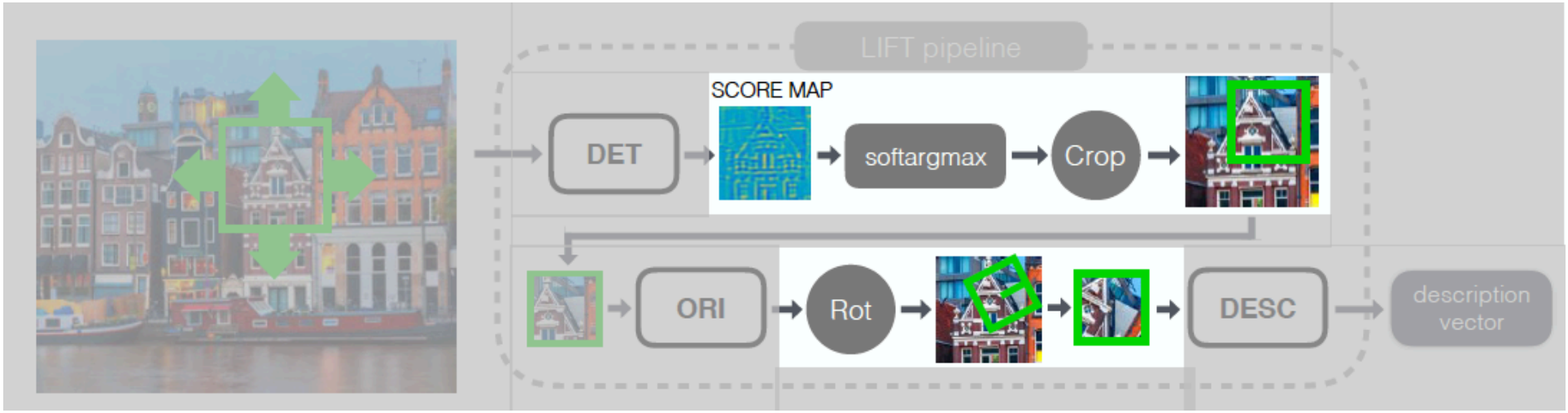
Orientation estimator produces a rotation angle.

Description network



Descriptor produces a feature vector for matching.

“Glue” to Connect Networks

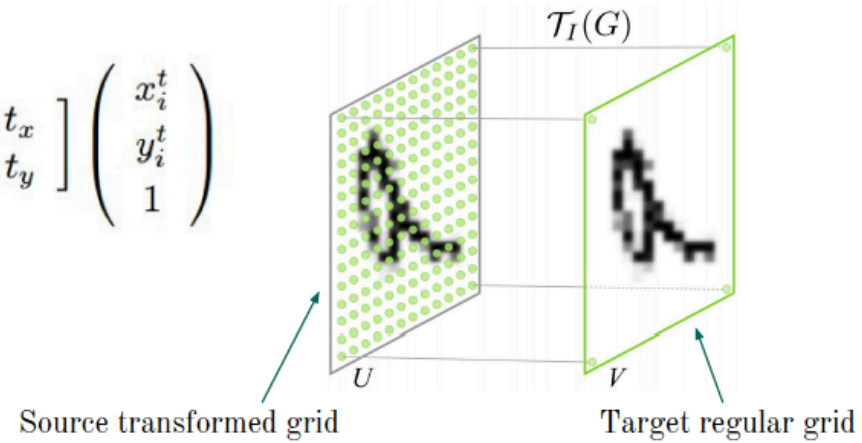


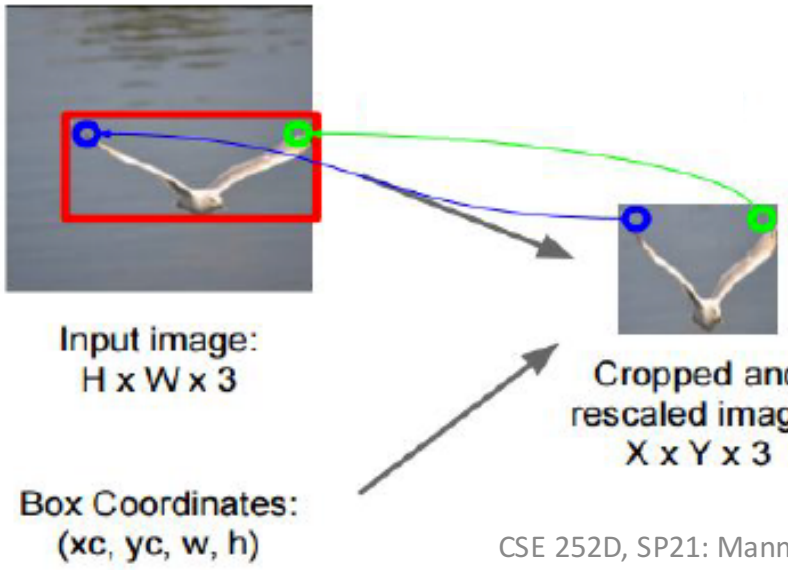
Spatial transformers to perform cropping and rotation to connect different tasks.

Spatial Transformers

Example: Cropping

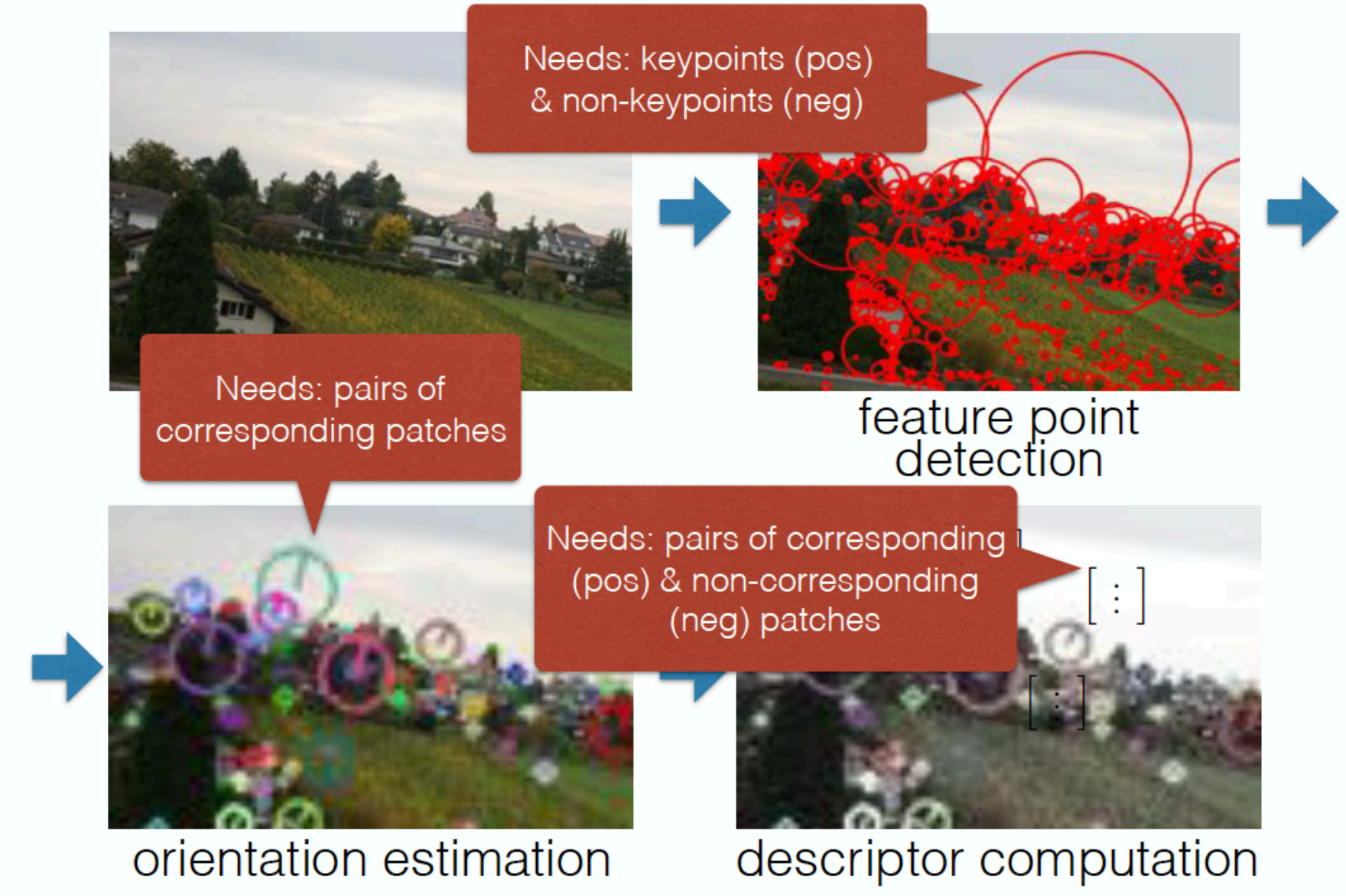
$$\begin{pmatrix} x_i^s \\ y_i^s \end{pmatrix} = \mathcal{T}_\theta(G_i) = \mathtt{A}_\theta \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix} = \begin{bmatrix} s & 0 & t_x \\ 0 & s & t_y \end{bmatrix} \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix}$$



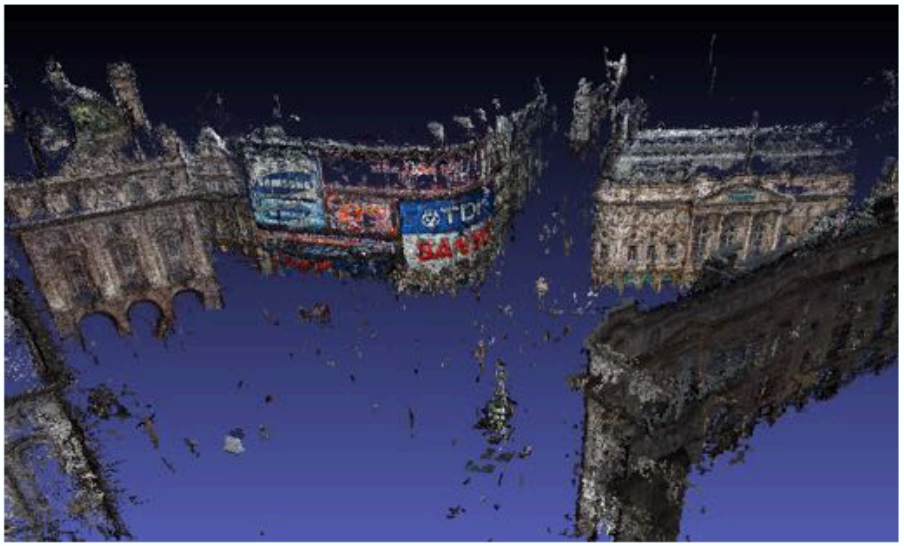


With s < 1, a crop is obtained.

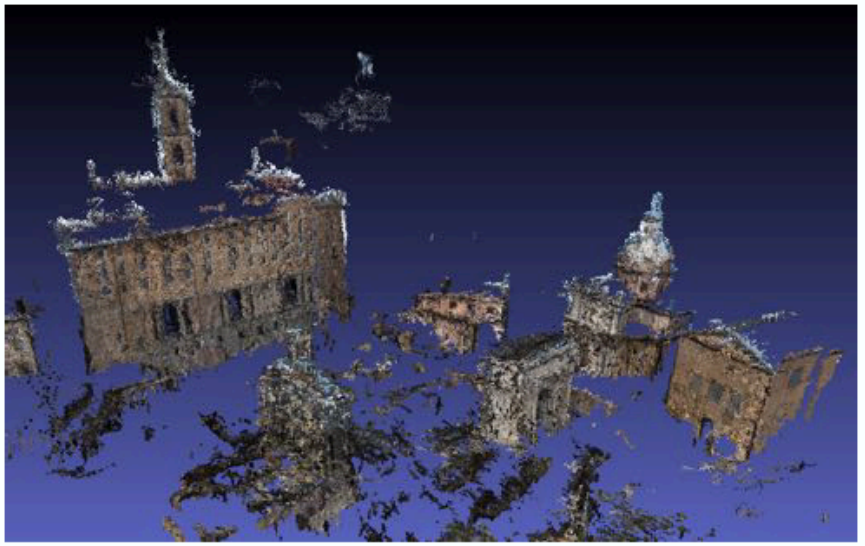
Training Requirements



Training Requirements



Piccadilly (pic)

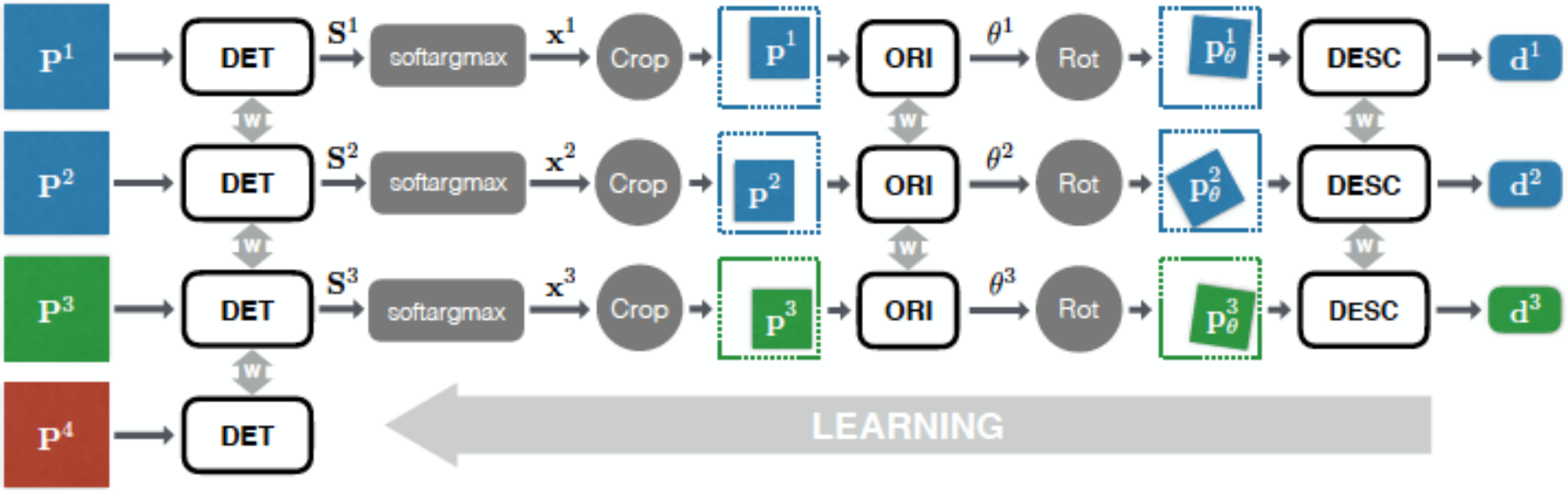


Roman Forum (rf)

- Train with SFM keypoints
- Extract SIFT features and do feature matching
- Points that are successfully matched by SFM are positives

- Descriptor training : SIFT keypoint patch and orientation
- Orientation training : SIFT keypoint patch
- Detector training : SIFT keypoint loosely cropped patch

Learning pipeline



Keypoint detection, orientation estimation and feature descriptor are all learned.

P^1, **P**^{2}: from same 3D keypoint in two views
P^3: from another keypoint
P^4: non-feature point.

Descriptor learning

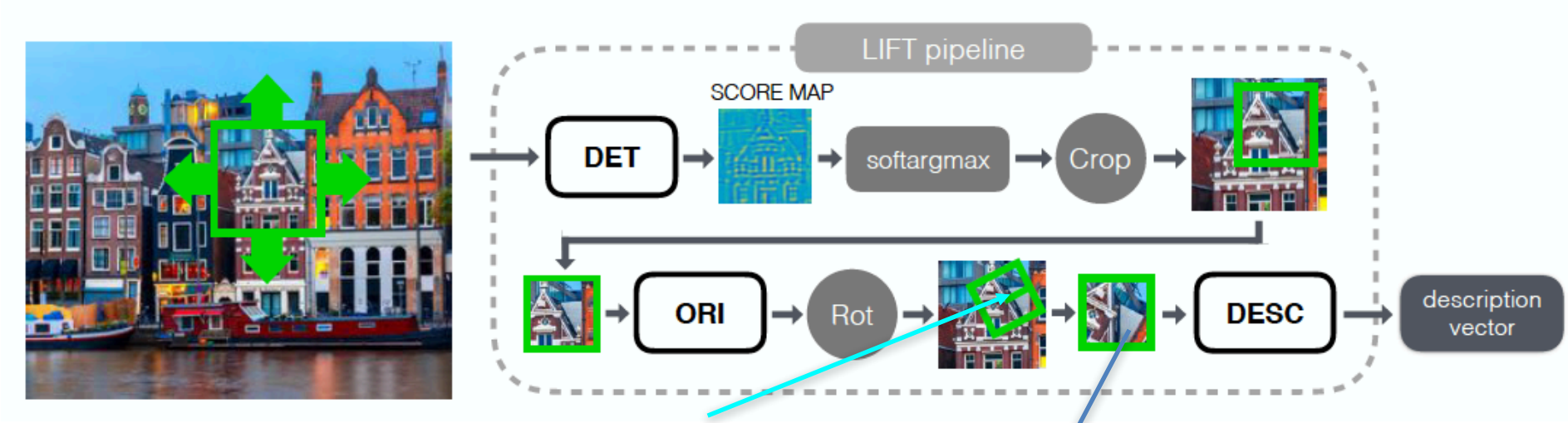
- Minimize the distance for corresponding matches.
- Maximize it for non-corresponding patches.

$$\mathcal{L}(\text{patch}_1, \text{patch}_2) = \|\text{Descriptor}(\text{patch}_1) - \text{Descriptor}(\text{patch}_2)\|^2$$

$$\mathcal{L}(\text{patch}_1, \text{patch}_3) = \max(0, C - \|\text{Descriptor}(\text{patch}_1) - \text{Descriptor}(\text{patch}_3)\|^2)$$

patch_1 patch_2 patch_3

Descriptor learning



Use locations and orientations of SIFT keypoints for training descriptor:

$$\mathbf{d} = h_\rho(\mathbf{p}_\theta)$$

Rotated patch

Use metric learning for training:

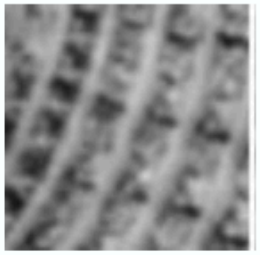
$$\mathcal{L}_{\text{desc}}(\mathbf{p}_\theta^k, \mathbf{p}_\theta^l) = \begin{cases} \left\| h_\rho(\mathbf{p}_\theta^k) - h_\rho(\mathbf{p}_\theta^l) \right\|_2 & \text{for positive pairs,} \\ \max\left(0, C - \left\| h_\rho(\mathbf{p}_\theta^k) - h_\rho(\mathbf{p}_\theta^l) \right\|_2\right) & \text{for negative pairs} \end{cases}$$

Hard example mining:

Forward pass a few pairs and evaluate losses

Only use a fraction of pairs with highest losses for backpropagation

Orientation estimator learning

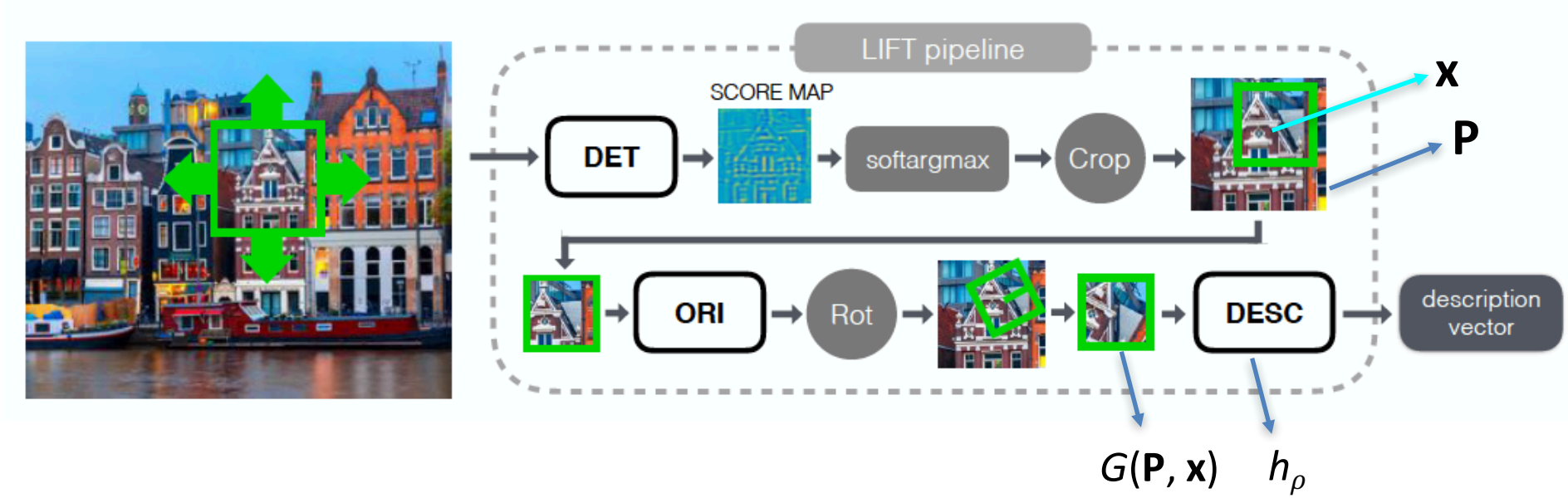


patch_1

patch_2

$$\mathcal{L}(\text{patch}_1, \text{patch}_2) = \left\| \begin{array}{l} \text{Descriptor}(\text{patch}_1, \text{Orientation}(\text{patch}_1)) - \\ \text{Descriptor}(\text{patch}_2, \text{Orientation}(\text{patch}_2)) \end{array} \right\|^2$$

Orientation estimator learning



Let $G(\mathbf{P}, \mathbf{x})$ be patch centered at x after orientation normalization of patch **P**.

Loss for training orientation estimator:

$$\mathcal{L}_{\text{orientation}}(\mathbf{P}^1, \mathbf{x}^1, \mathbf{P}^2, \mathbf{x}^2) = \left\| h_\rho(G(\mathbf{P}^1, \mathbf{x}^1)) - h_\rho(G(\mathbf{P}^2, \mathbf{x}^2)) \right\|_2$$

Want the descriptors to align after orientation correction.
Detector is not trained yet, so still use SFM points for **x**.

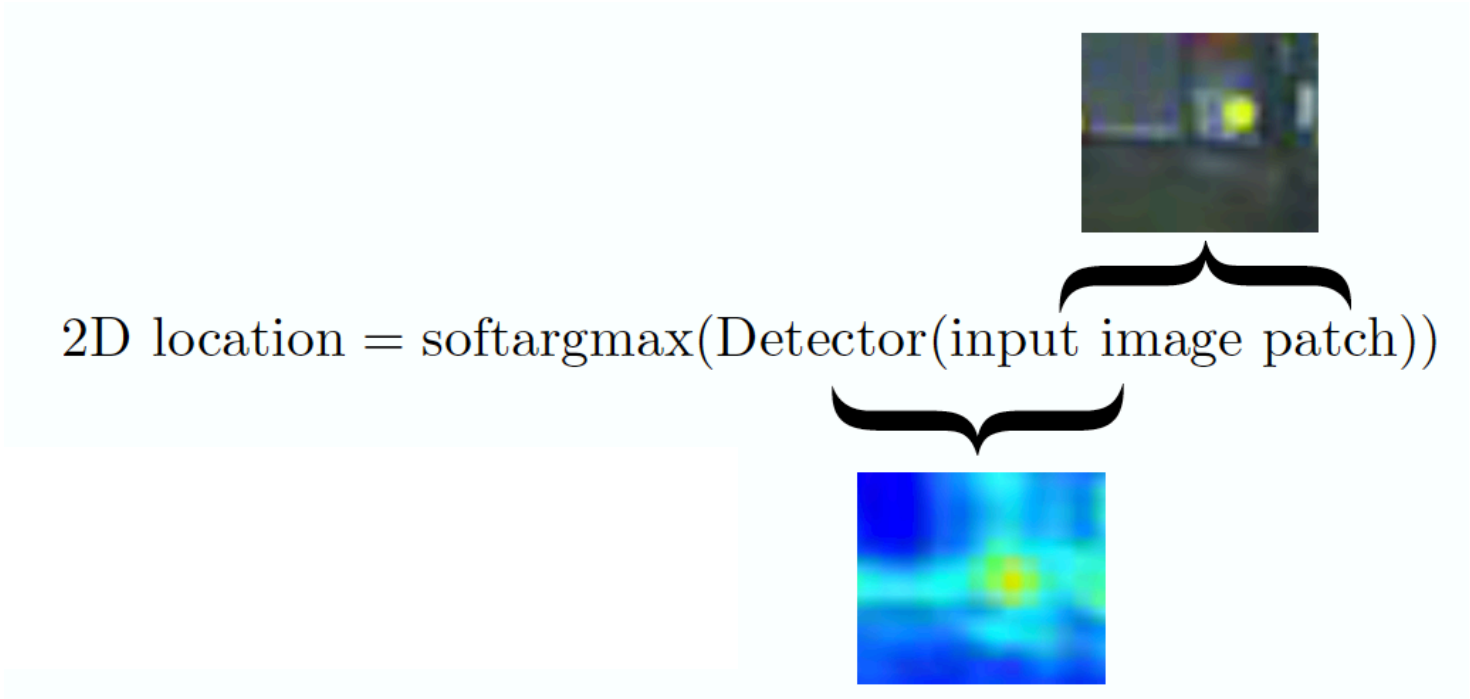
Detector training

Obtain score map for input match (piecewise linear activations on convolutional output):

$$\mathbf{S} = f_\mu(\mathbf{P}) = \sum_n^N \delta_n \max_m^M (\mathbf{W}_{mn} * \mathbf{P} + \mathbf{b}_{mn})$$

Use softmax for differentiable variant of non-maximal suppression:

$$\mathbf{x} = \text{softargmax}(\mathbf{S}) = \frac{\sum_{\mathbf{y}} \exp(\beta \mathbf{S}(\mathbf{y}))\mathbf{y}}{\sum_{\mathbf{y}} \exp(\beta \mathbf{S}(\mathbf{y}))}$$



Overall Training

$\mathbf{P}^1$, $\mathbf{P}^2$: from same 3D keypoint in two views, $\mathbf{P}^3$: from another keypoint, $\mathbf{P}^4$: non-feature point.

Pair loss: projections of same 3D point should have similar descriptors

Classification loss: push score map to high values for positive classes

$$\min_{\{f_\mu, g_\phi, h_\rho\}} \sum_{\{(\mathbf{P}_1,\mathbf{P}_2,\mathbf{P}_3,\mathbf{P}_4)\}} \gamma \mathcal{L}_{class}(\mathbf{P}^1,\mathbf{P}^2,\mathbf{P}^3,\mathbf{P}^4) + \mathcal{L}_{pair}(\mathbf{P}^1,\mathbf{P}^2)$$

detector
orientation
descriptor

$$\mathcal{L}_{\text{class}}(\mathbf{P}^1,\mathbf{P}^2,\mathbf{P}^3,\mathbf{P}^4) = \sum_{i=1}^{4} \alpha_i \max\left(0, \left(1 - \text{softmax}\left(f_\mu\left(\mathbf{P}^i\right)\right) y_i\right)\right)^2$$

detector

$$\mathcal{L}_{\text{pair}}(\mathbf{P}^1,\mathbf{P}^2) = \| \; h_\rho(G(\mathbf{P}^1, \text{softargmax}(f_\mu(\mathbf{P}^1)))) - h_\rho(G(\mathbf{P}^2, \text{softargmax}(f_\mu(\mathbf{P}^2)))) \; \|_2$$

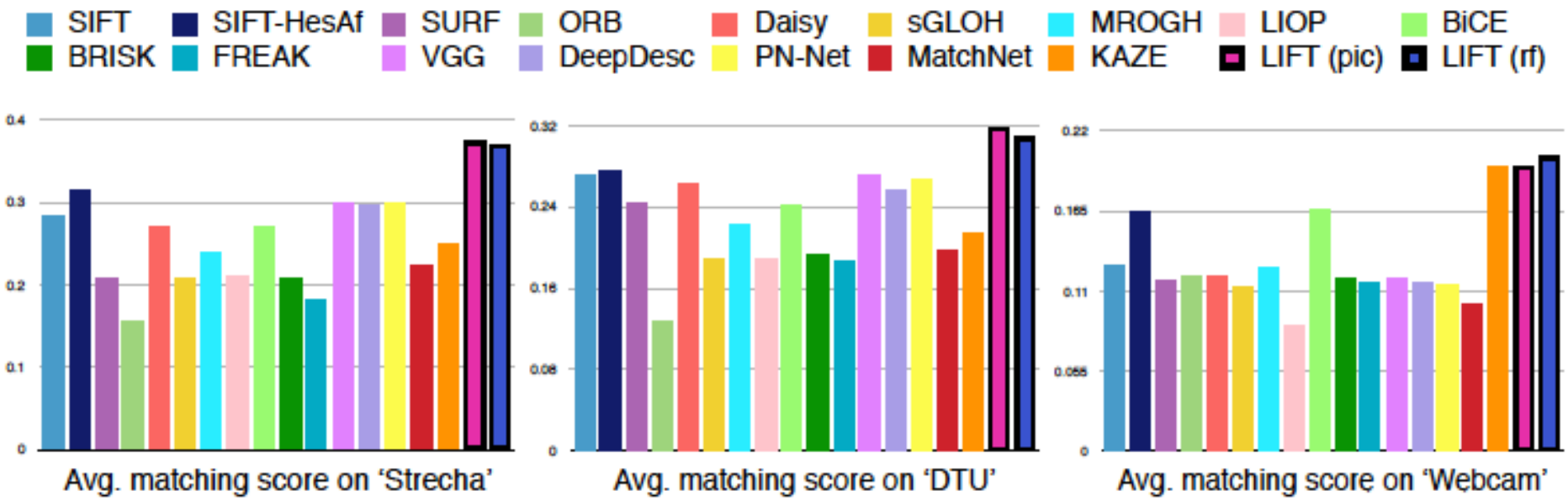
descriptor

$$G(\mathbf{P},\mathbf{x}) = \text{Rot}(\mathbf{P},\mathbf{x}, g_\phi(\text{Crop}(\mathbf{P},\mathbf{x})))$$

orientation

Metrics

- **Repeatability**: fraction of keypoints common between images 1 and 2
- **Matching score**: ratio of correspondences and keypoints



Qualitative results

Strecha

Webcam

DTU

DTU