CSE 252D: Advanced Computer Vision

Manmohan Chandraker

Lecture 19: Review

Virtual classrooms

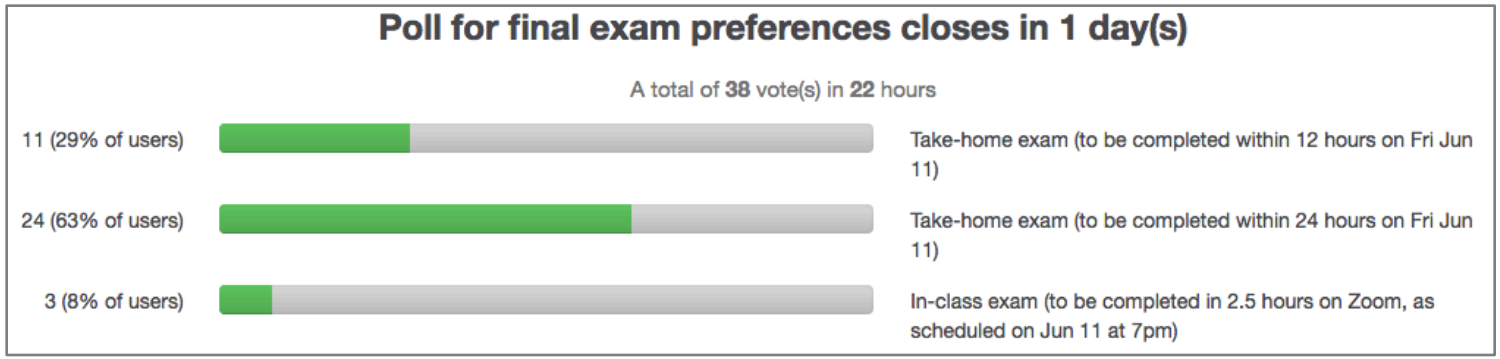
- Virtual lectures on Zoom
  - Only host shares the screen
  - Keep video off and microphone muted
  - But please do speak up (remember to unmute!)
  - Slides uploaded on webpage just before class
- Virtual interactions on Zoom
  - Ask and answer plenty of questions
  - “Raise hand” feature on Zoom when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

Announcements

- Final exam will be take-home
  - Released Jun 10 at 10pm and due on Jun 11 at 10pm

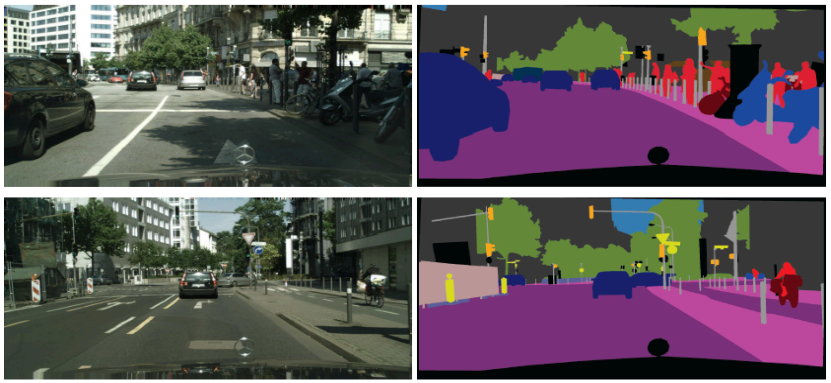


- Solutions to quizzes on course webpage
- Sample final questions on course webpage
  - Some of those questions will be in the actual final!
- Homeworks due Jun 6
  - Feedback shared with submitted ones, email your questions
- Let me know any accommodations needed

Recap

Advantage 1: Robustness of Solution

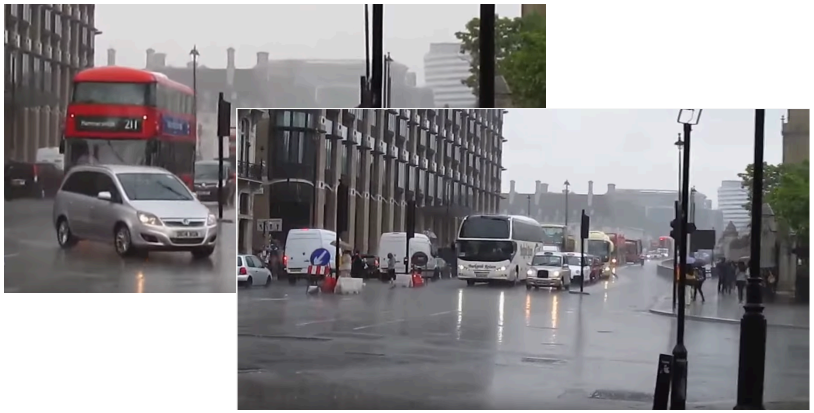
Source domain: good weather, **with** labels



Labels require **1.5 hours** per image!

Train on source, **apply** on target

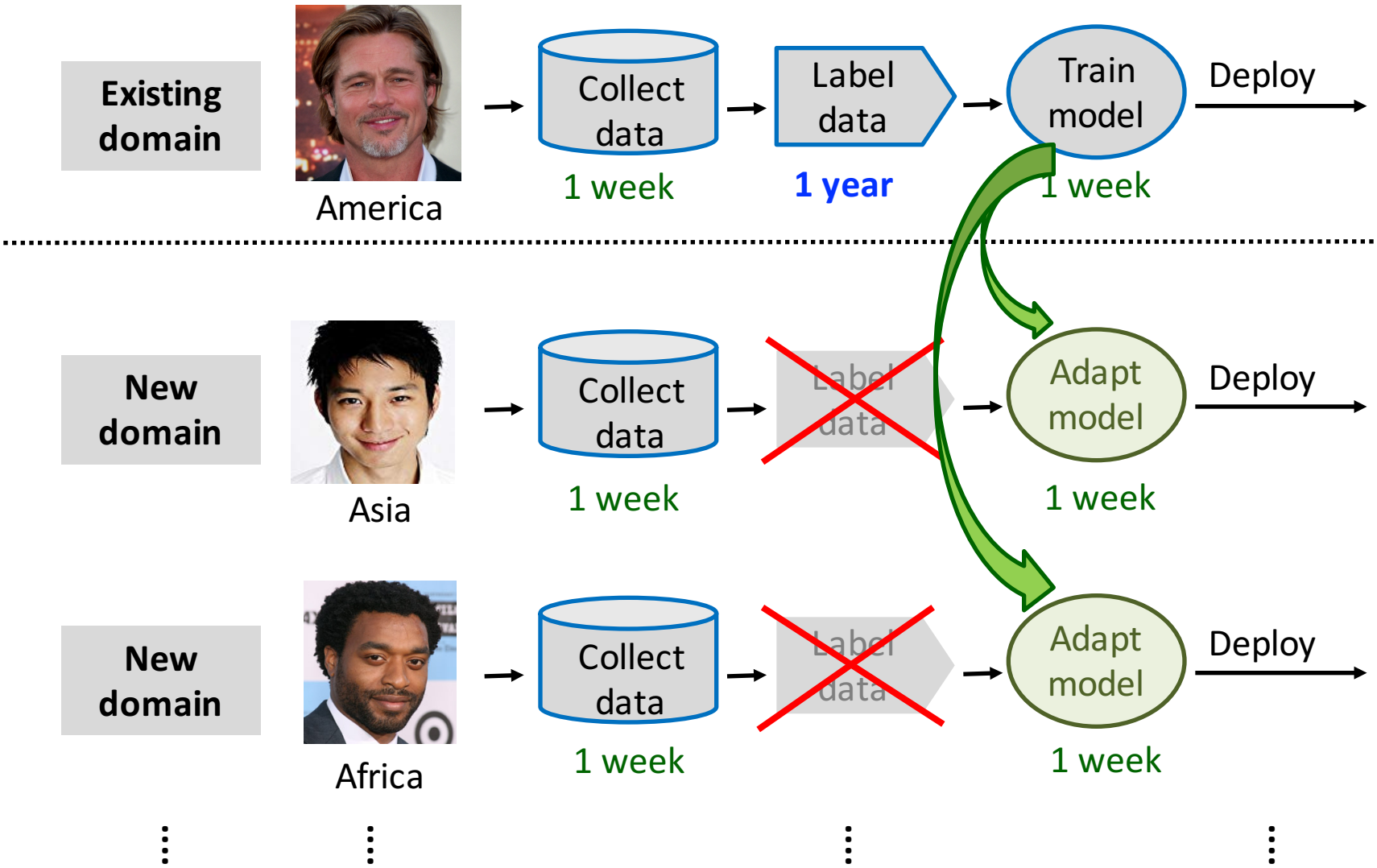
Target domain: rainy weather, **no** labels



Train on source, **adapt** to target

Advantage 2: Ease of Deployment

Domain adaptation eases deployment by reducing need for data labeling

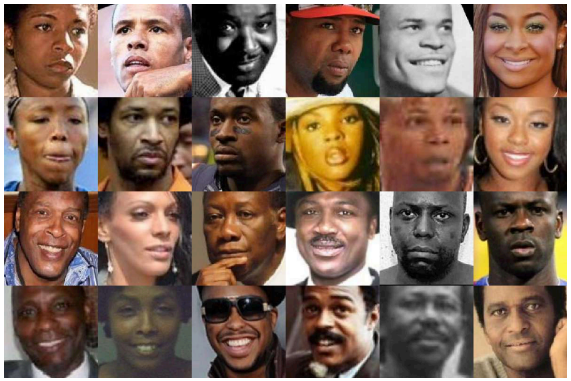


Advantage 3: Fairness to Society

Proportion in datasets: 80%

Caucasian

Proportion in datasets: 10%



African-American

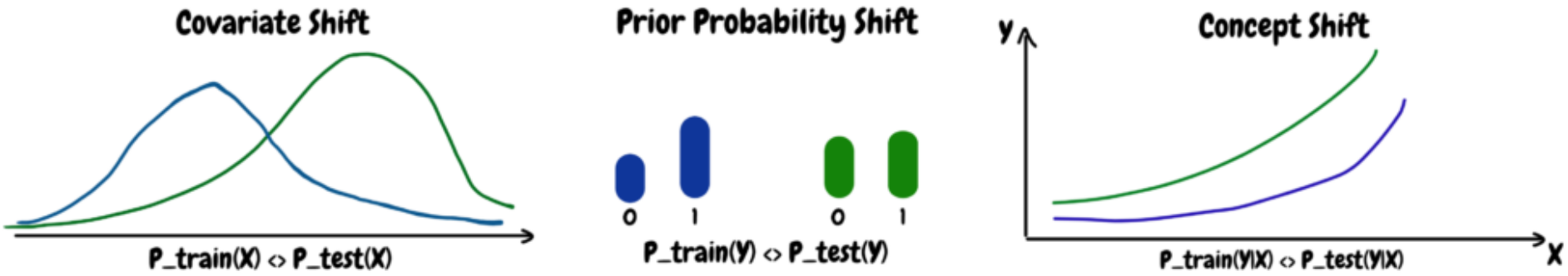
Proportion in datasets: 5%

East-Asian

Training on biased data without domain adaptation		
High accuracy	Low accuracy 😕	Low accuracy 😕

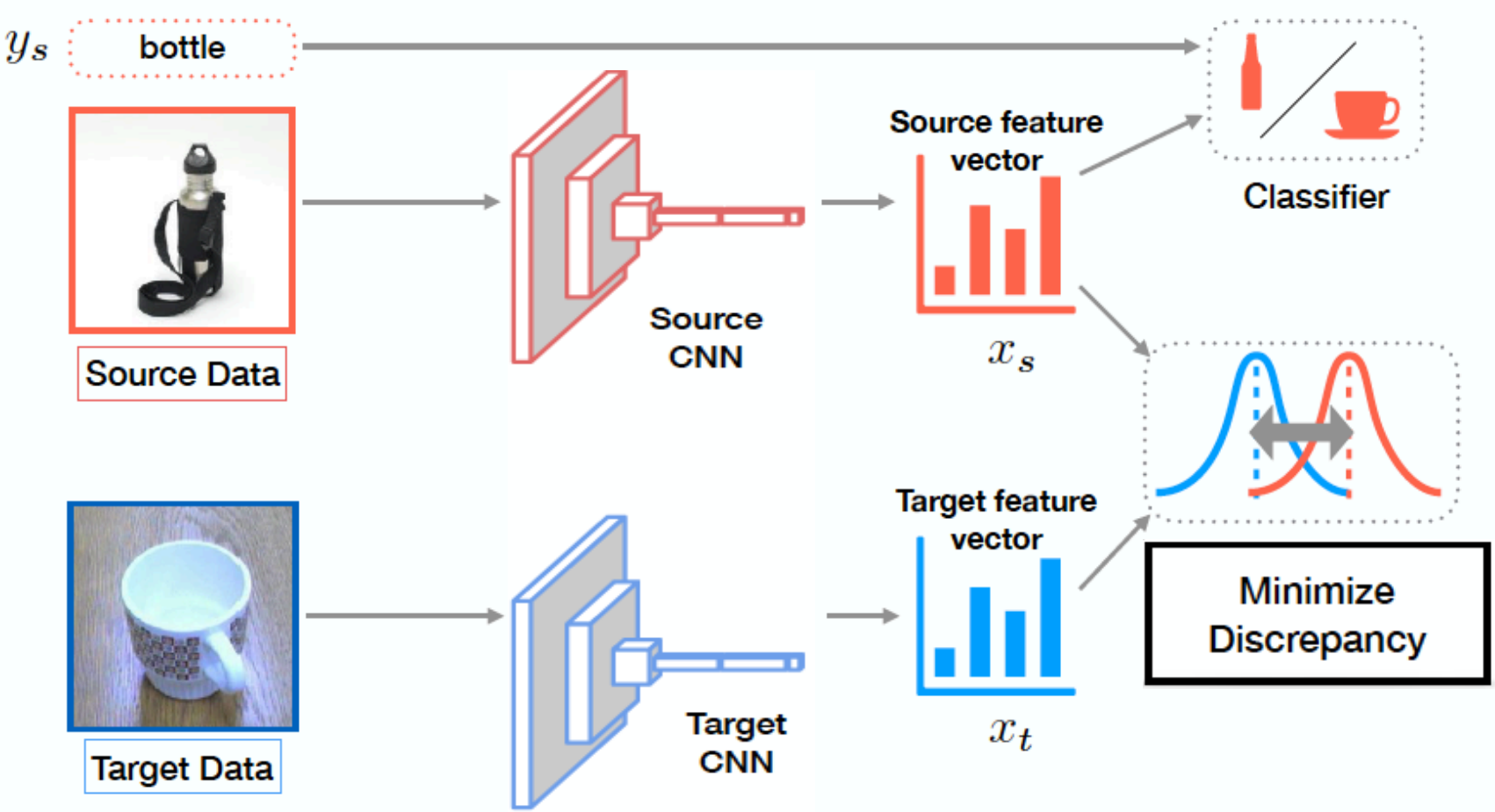
Using domain adaptation to address dataset bias		
High accuracy	High accuracy 🙂	High accuracy 🙂

Different types of data shift



- Data distributions p(x, y) cannot change in arbitrary ways
  - Example: call "cats" as "dogs" and the other way round
- Adaptation possible under reasonable assumptions of data shift

Deep Domain Adaptation



Learn a representation to minimize discrepancy

Generalized Adversarial Adaptation

- Learn classifier using standard supervised loss on source data

$$\min_{M_s, C} \mathcal{L}_{\text{cls}}(\mathbf{X}_s, Y_t) = \mathbb{E}_{(\mathbf{x}_s, y_s)\sim(\mathbf{X}_s, Y_t)} -\sum_{k=1}^{K} \mathbb{1}_{[k=y_s]} \log C(M_s(\mathbf{x}_s))$$

- Alternating minimization between two functions
  - Discriminator optimized with supervised loss (labels indicate domain)

$$\mathcal{L}_{\text{adv}_D}(\mathbf{X}_s, \mathbf{X}_t, M_s, M_t) = -\mathbb{E}_{\mathbf{x}_s\sim\mathbf{X}_s}[\log D(M_s(\mathbf{x}_s))] - \mathbb{E}_{\mathbf{x}_t\sim\mathbf{X}_t}[\log(1 - D(M_t(\mathbf{x}_t)))]$$

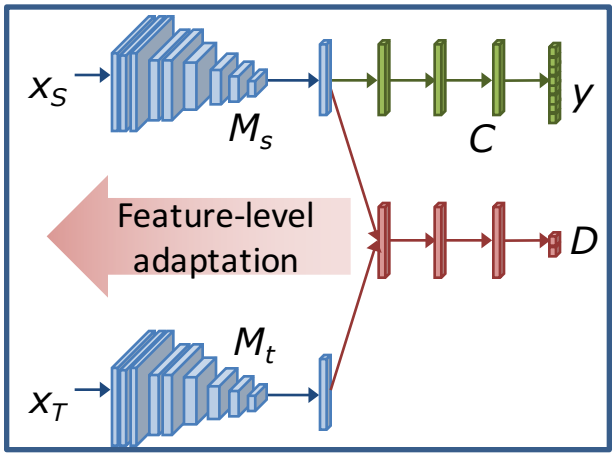
  - Source and target mappings optimized with constrained adversarial objective

$$\min_{D} \mathcal{L}_{\text{adv}_D}(\mathbf{X}_s, \mathbf{X}_t, M_s, M_t)$$

$$\min_{M_s, M_t} \mathcal{L}_{\text{adv}_M}(\mathbf{X}_s, \mathbf{X}_t, D) \qquad \boxed{\mathcal{L}_{\text{adv}_M} = -\mathcal{L}_{\text{adv}_D}}$$

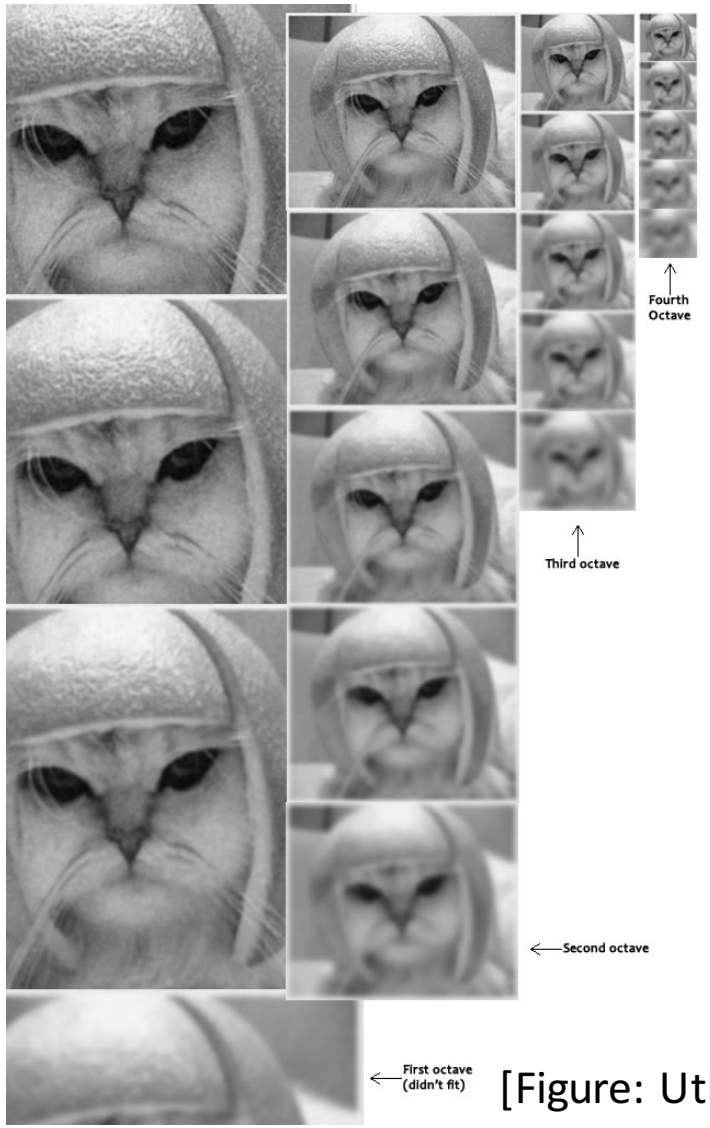
$$\text{s.t.} \quad \psi(M_s, M_t)$$

- While training D, keep encoders fixed, encourage $D(M_s)$ = 1 and $D(M_t)$ = 0
- To train encoders, keep D fixed, but now encourage $D(M_s)$ = 0 and $D(M_t)$ = 1



Correspondence

Scale Space for SIFT



- **Feature scale is important, but unknown**
- Facial attributes at close range
- Overall face shape at distance range

- **Gaussian filter to create scale space**
- Preserve some invariances
- Want features that are not artifacts
- Do not introduce new artifacts

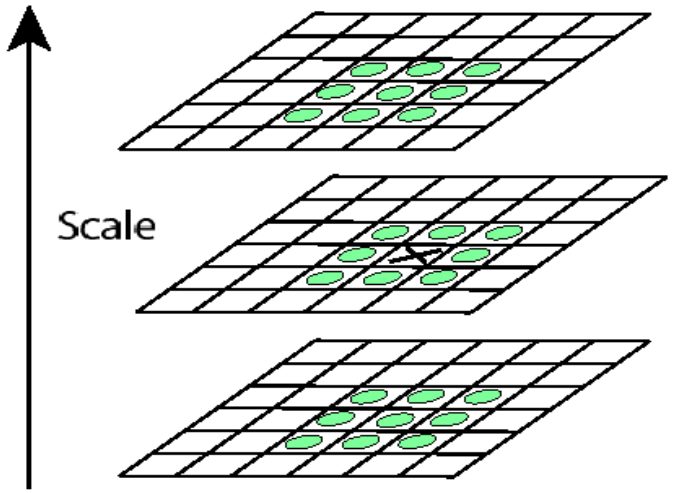
- **Octaves to more efficiently explore the scale space**
- Downsample image to lower frequency content
- Same filters to create scale space at next octave

[Figure: Utkarsh Sinha, aishack]

Key point localization

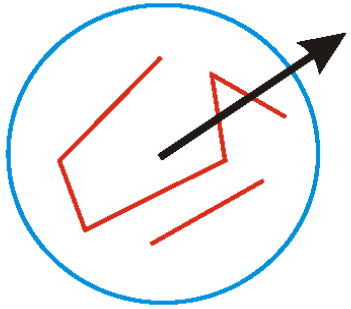
- Detect maxima and minima of difference-of-Gaussian in scale space
- Each point is compared to its 8 neighbors in the current image and 9 neighbors each in the scales above and below
- Retain only strongest keypoints, by computing eigenvalue ratio

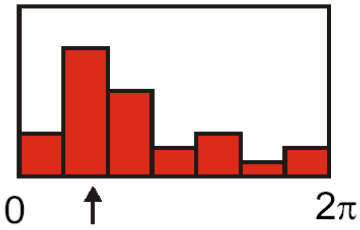
s+2 difference images.
top and bottom ignored.
s planes searched.



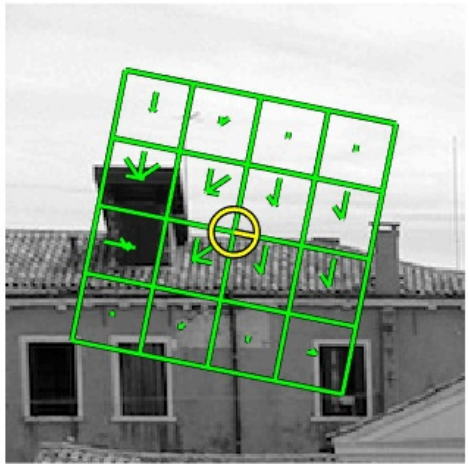
For each max or min found, output is the **location** and the **scale**.

Orientation assignment





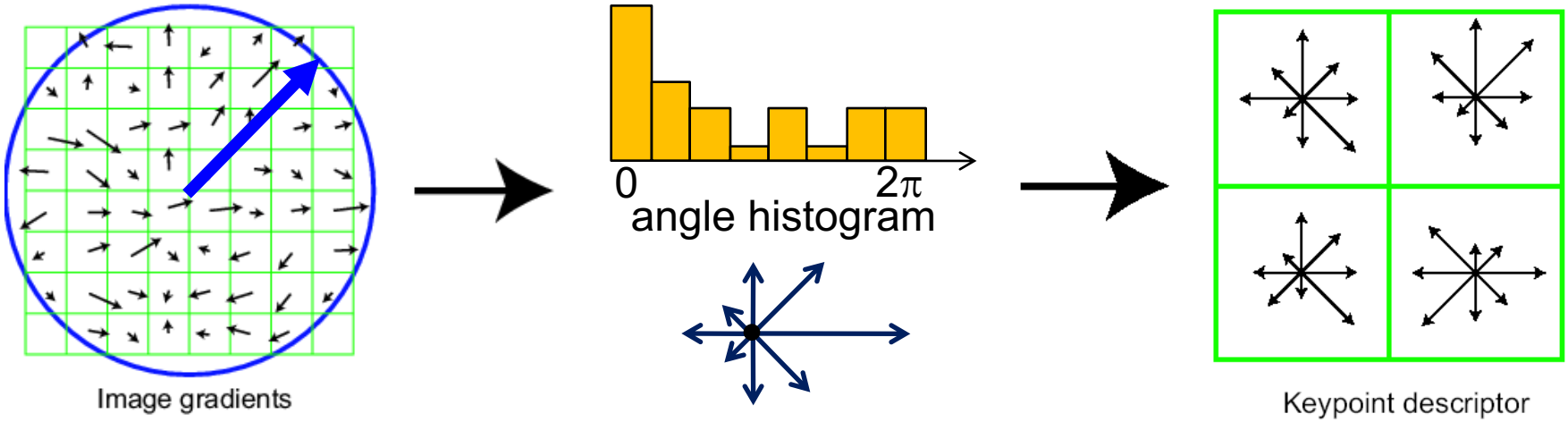
- Create histogram of local gradient directions at selected scale
- Assign canonical orientation at peak of smoothed histogram
- Consider windows around keypoints that are normalized for scale and orientation



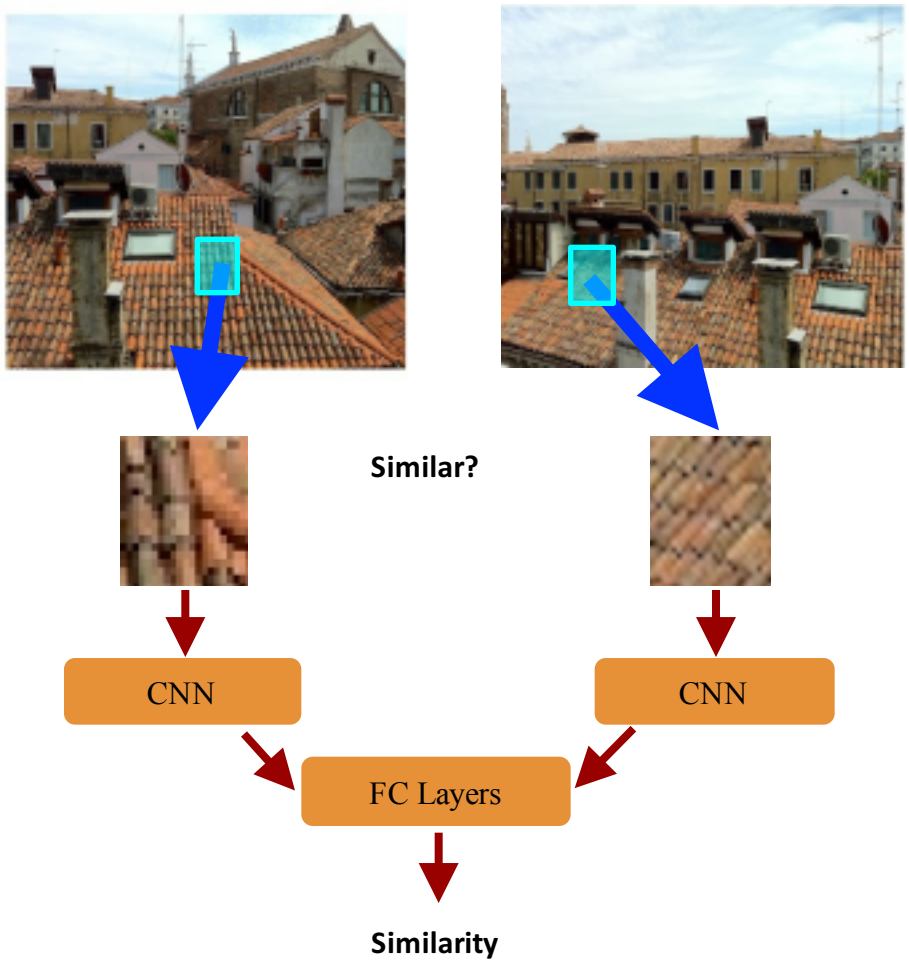
Descriptor computation

- Consider 16 x 16 local window
- For each 4 x 4 sub-window
  - Compute edge orientation (angle of the gradient - 90°) for each pixel
  - Throw out weak edges (threshold gradient magnitude)
  - Create 8-bin histogram of surviving edge orientations
- Concatenate histograms from each sub-window

Illustration for a 2 x 2 sub-window



Correspondence beyond similarity CNN



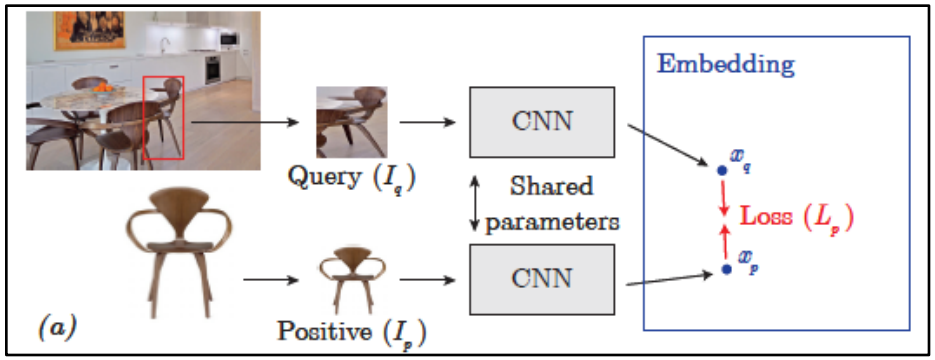
- Detection of interest points
- Normalization of patches
- Multiscale information
- Obtaining training data
- Efficient training and testing

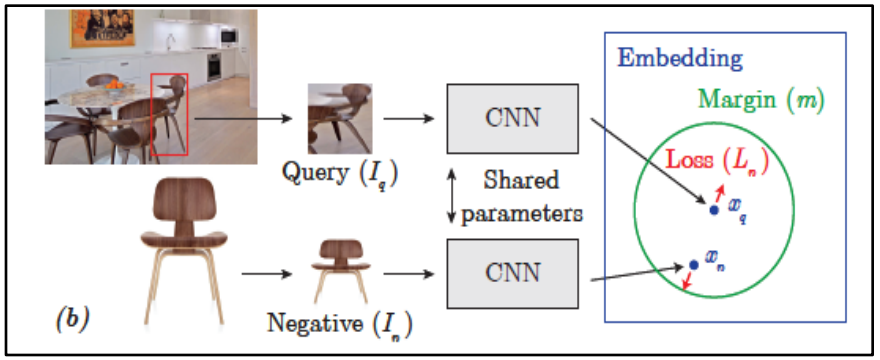
Contrastive Loss for Metric Learning

- 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

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

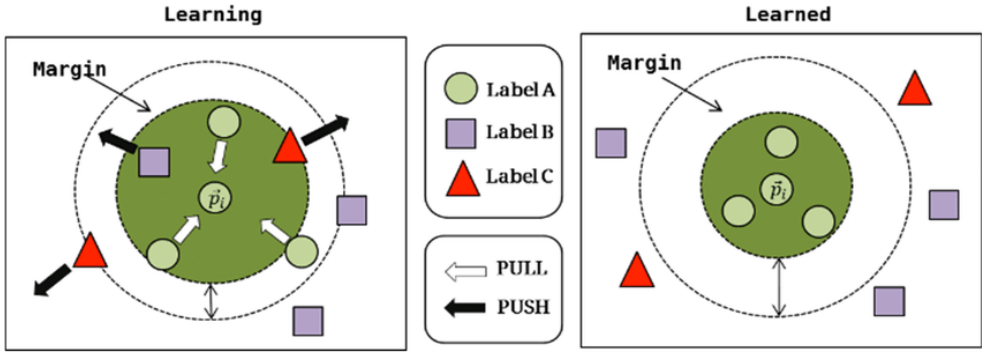
Anchor Positive Negative

- 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:

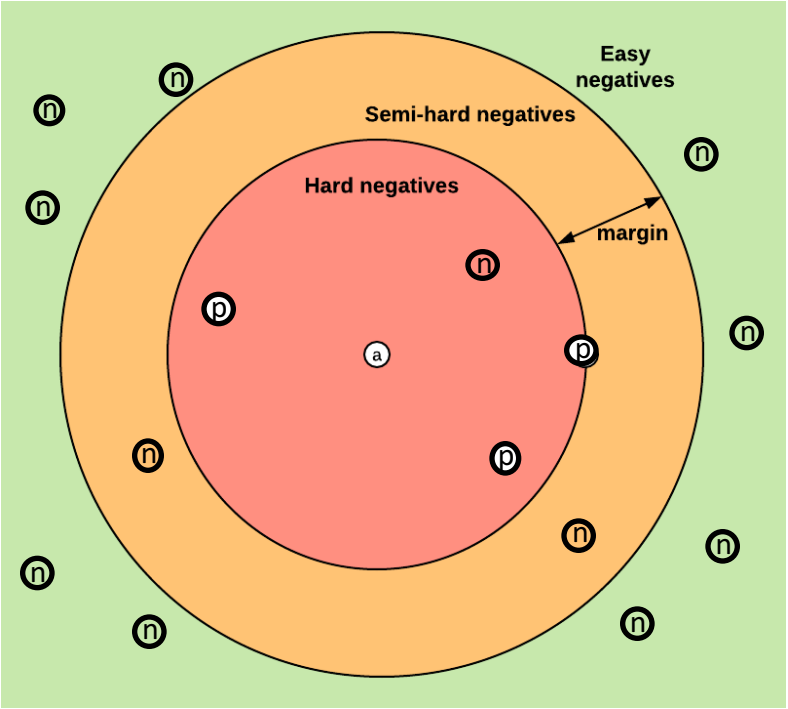
$$l_{tri}(\mathcal{T}) = \left[\|\mathbf{x}_a - \mathbf{x}_p\|^2 - \|\mathbf{x}_a - \mathbf{x}_n\|^2 + m\right]_+$$



Hard Negative Mining

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

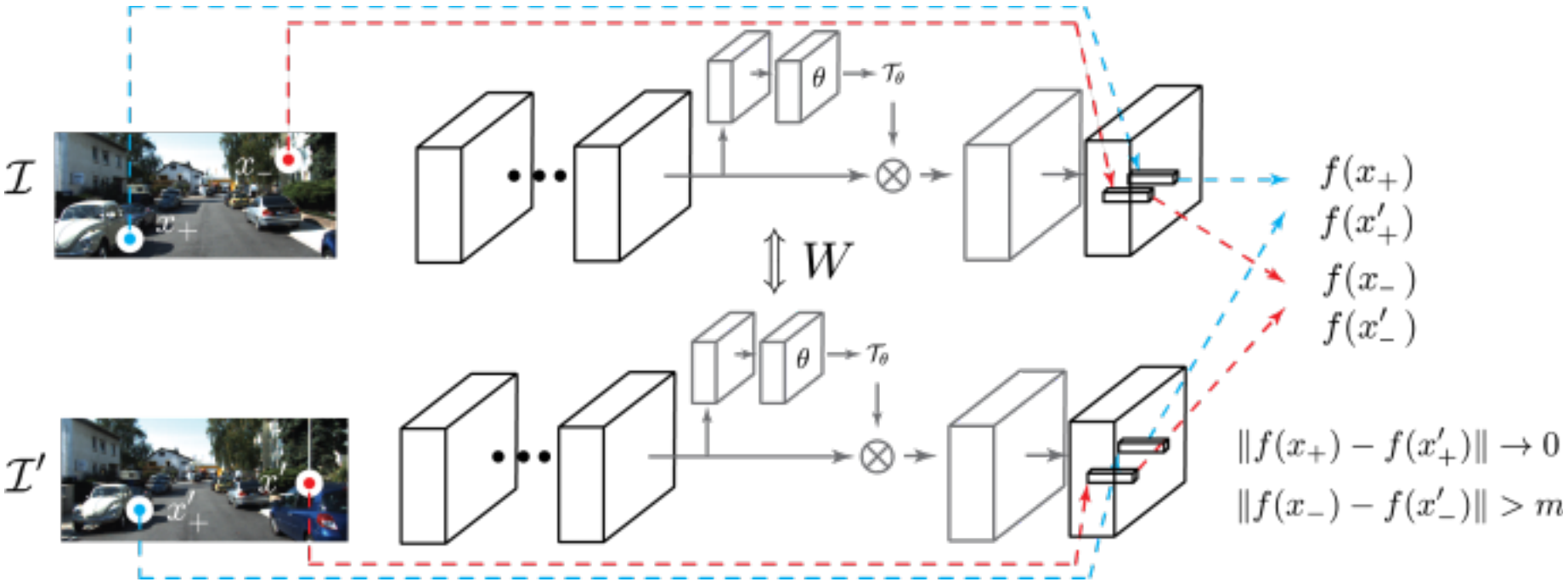
Loss is 0 for easy negatives



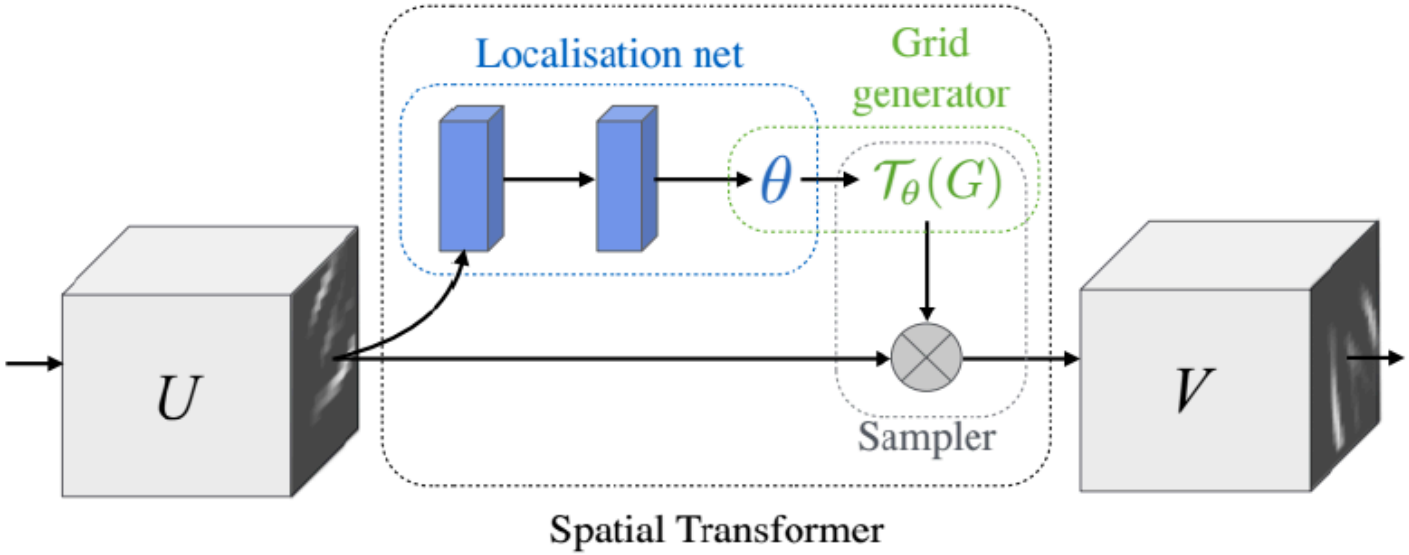
- Metric learning is driven by hard negatives
- Most negatives are easy
- Need strategy to find hard negatives among samples

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

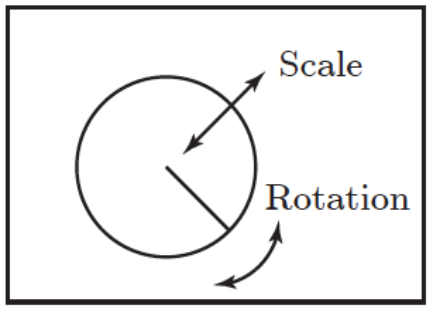


- **Localization net** takes input feature map U and outputs transformation parameters

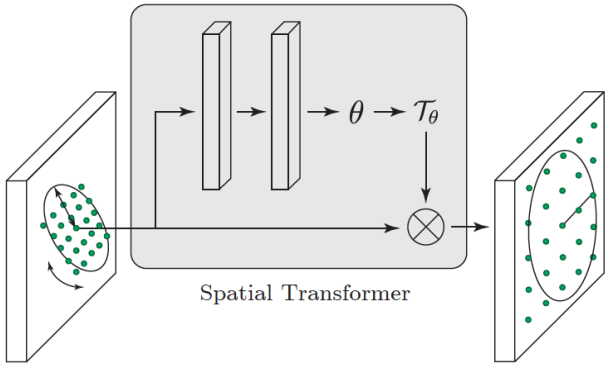
- **Grid generator** takes uniform grid in output map and deforms it by transformation
- This determines a deformed grid to sample the input feature map U

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

Patch Normalization in UCN

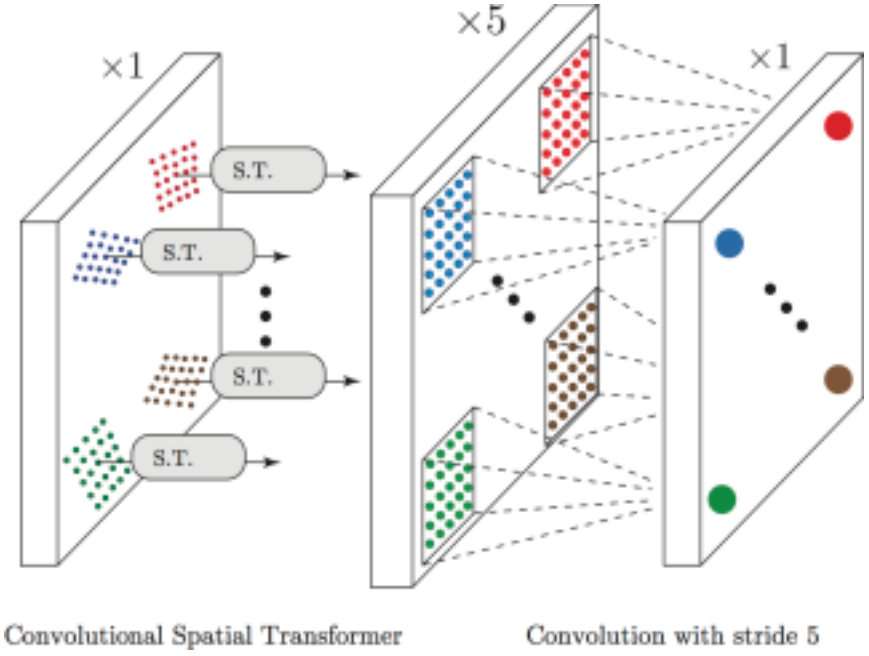


SIFT normalizes for scale and rotation



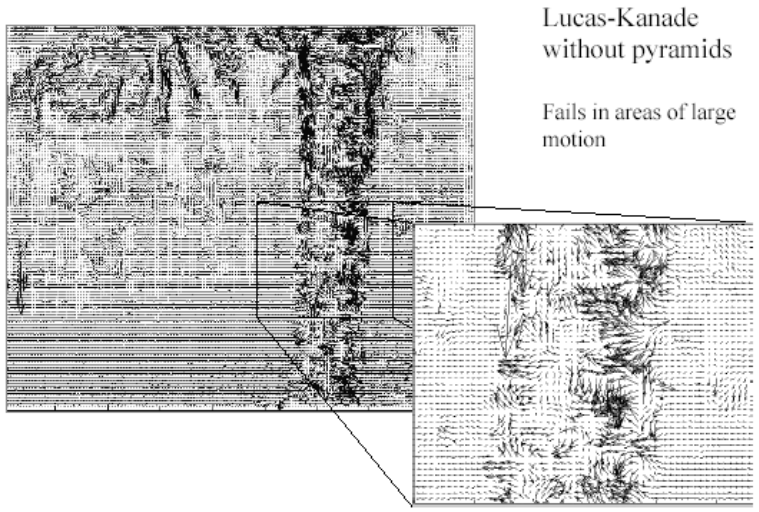
Spatial transformer for global transformation

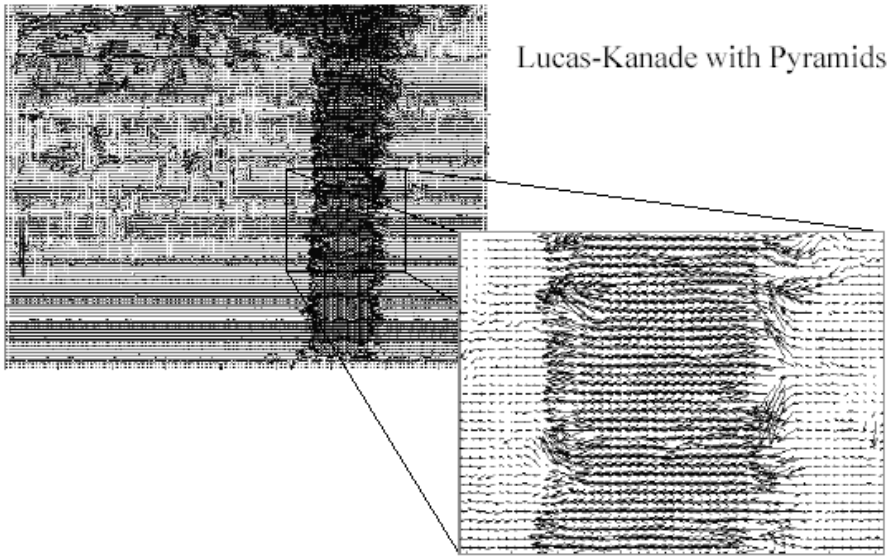
Convolutional spatial transformer for independent normalization at each pixel

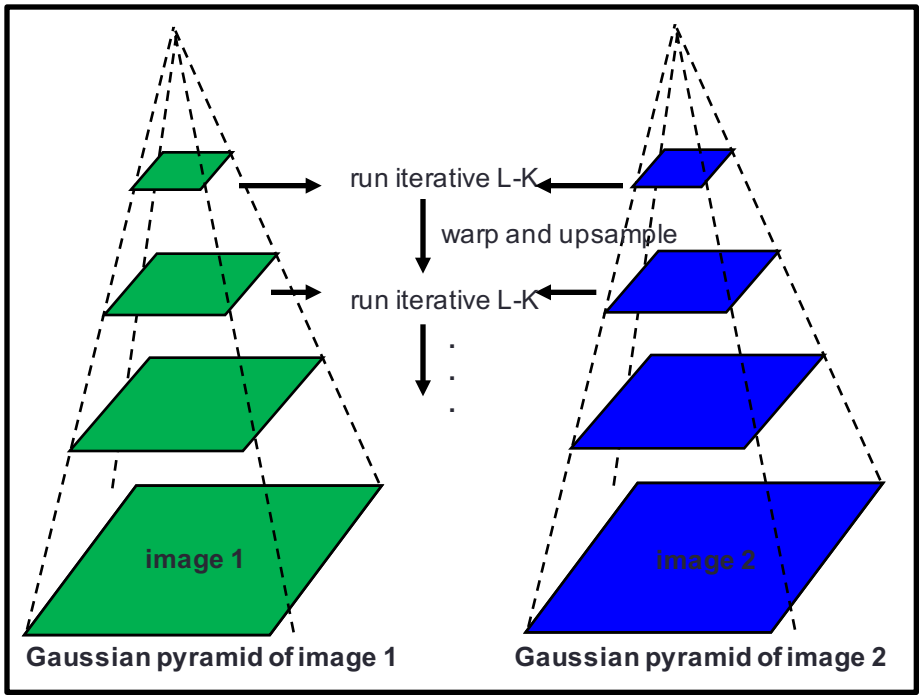


Optical Flow

Optical flow: Coarse-to-Fine

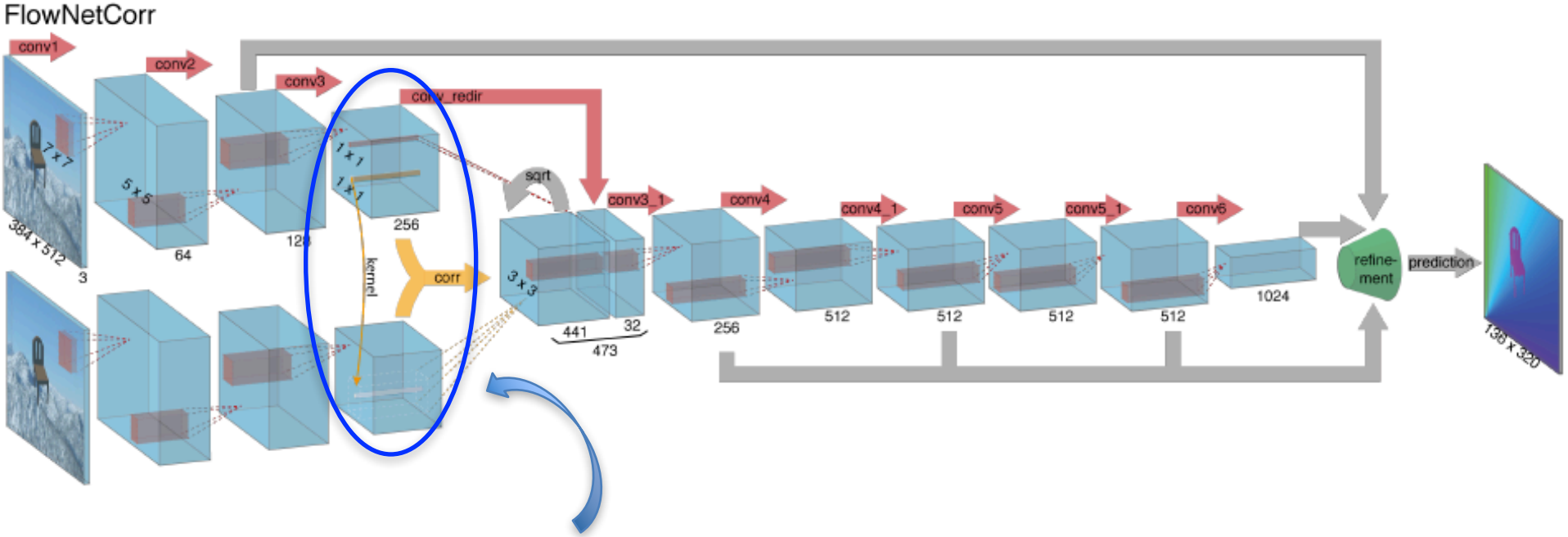






* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003

FlowNetCorr: Correlation Layer

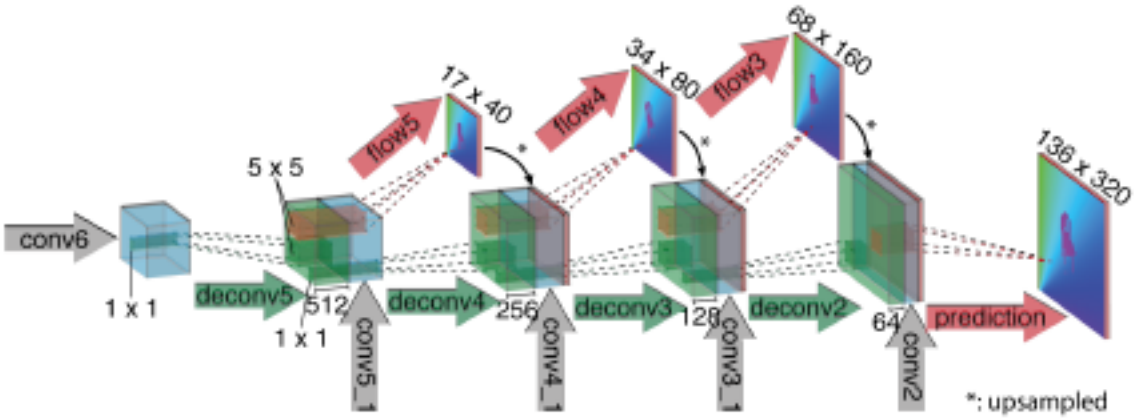


- Multi-channel feature maps $\mathbf{f}_1$ and $\mathbf{f}_2$, of size w x h and n channels
- Correlation of patches centered at $\mathbf{x}_1$ and $\mathbf{x}_2$:

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

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

FlowNetCorr: 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.

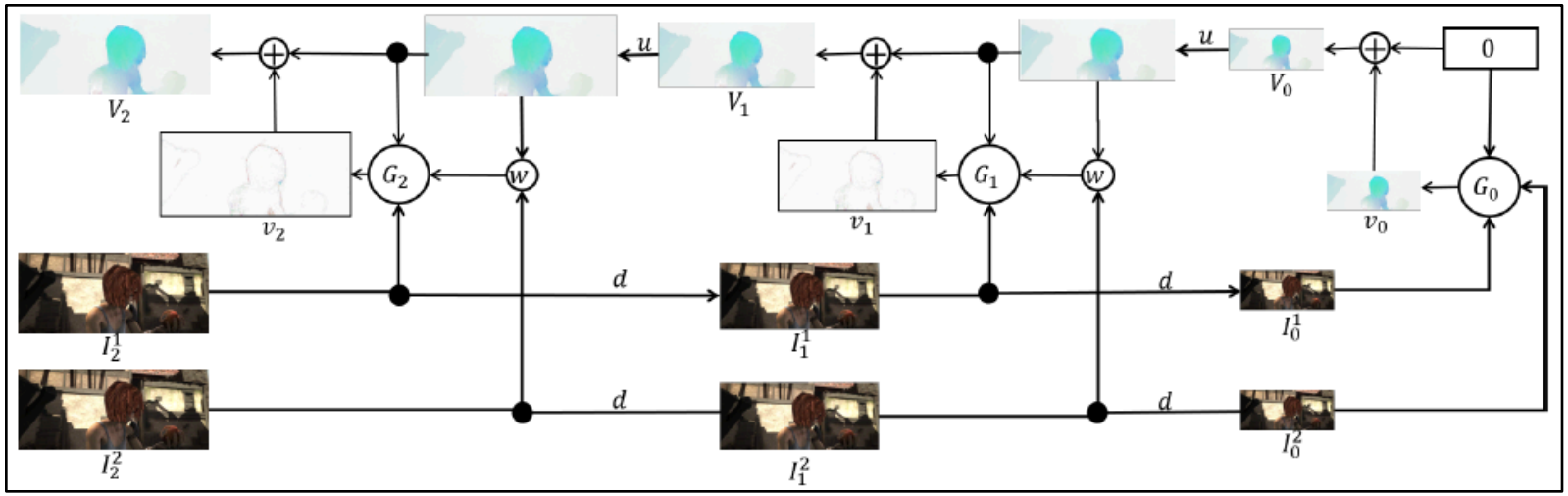
- Can be learned as unpooling followed by convolution
- Can be learned as a transposed convolution filter

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
- Instead of flow, learn increment over upsampled flow at each pyramid level
  - Residual flow has small magnitude, easier to learn
- Train each level G_k sequentially to predict residual at level k, given trained G_{k-1}

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

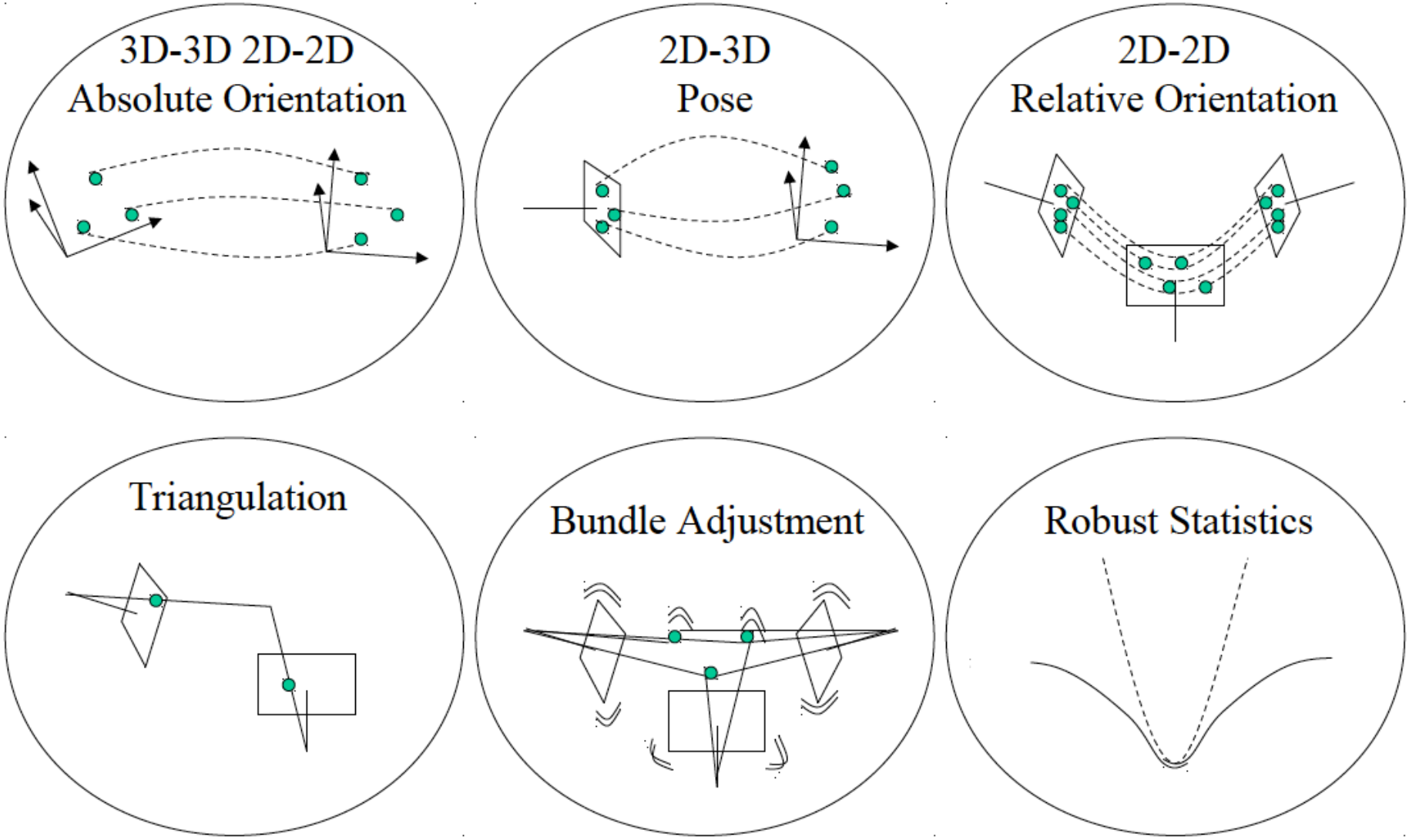
- Each level solves a simple problem, so each level G_k can have a simple architecture



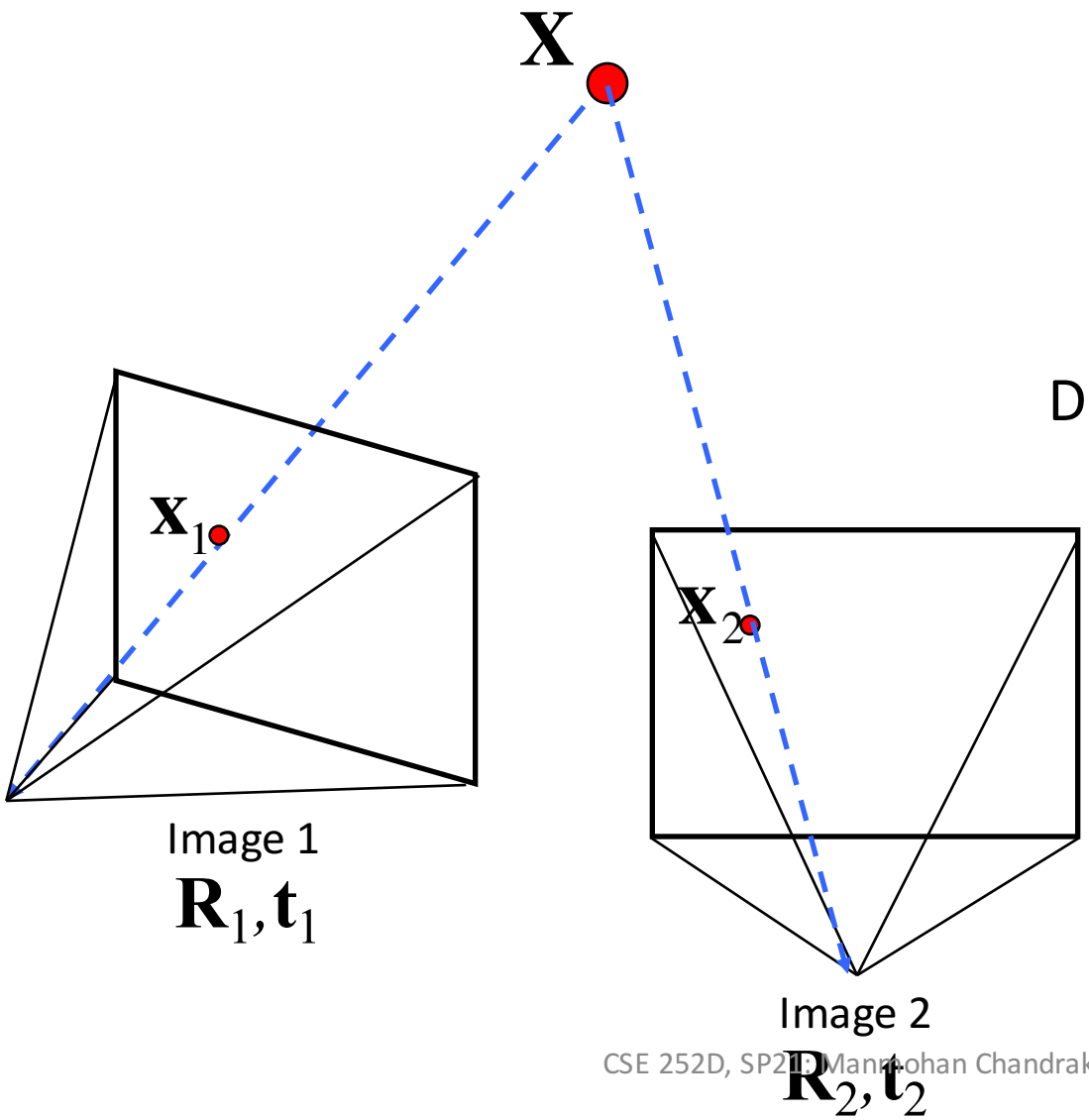
[Optical Flow Estimation Using a Spatial Pyramid Network, CVPR 2017]

Structure from Motion

Toolkit for Practical SFM



Fundamental Matrix



$$\mathbf{x}_1 \leftrightarrow \mathbf{x}_2$$

$$\mathbf{x}_1^T \mathbf{F} \mathbf{x}_2 = 0$$

Degrees of freedom for F: 7

So, 7 points suffice to find F, but use 8 for linear method

Direct Linear Transform

Given *n* point correspondences, set up a system of equations:

$$\begin{pmatrix} u_1u_1' & v_1u_1' & u_1' & u_1v_1' & v_1v_1' & v_1' & u_1 & v_1 & 1 \\ u_2u_2' & v_2u_2' & u_2' & u_2v_2' & v_2v_2' & v_2' & u_2 & v_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_nu_n' & v_nu_n' & u_n' & u_nv_n' & v_nv_n' & v_n' & u_n & v_n & 1 \end{pmatrix} \begin{pmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{pmatrix} = 0$$

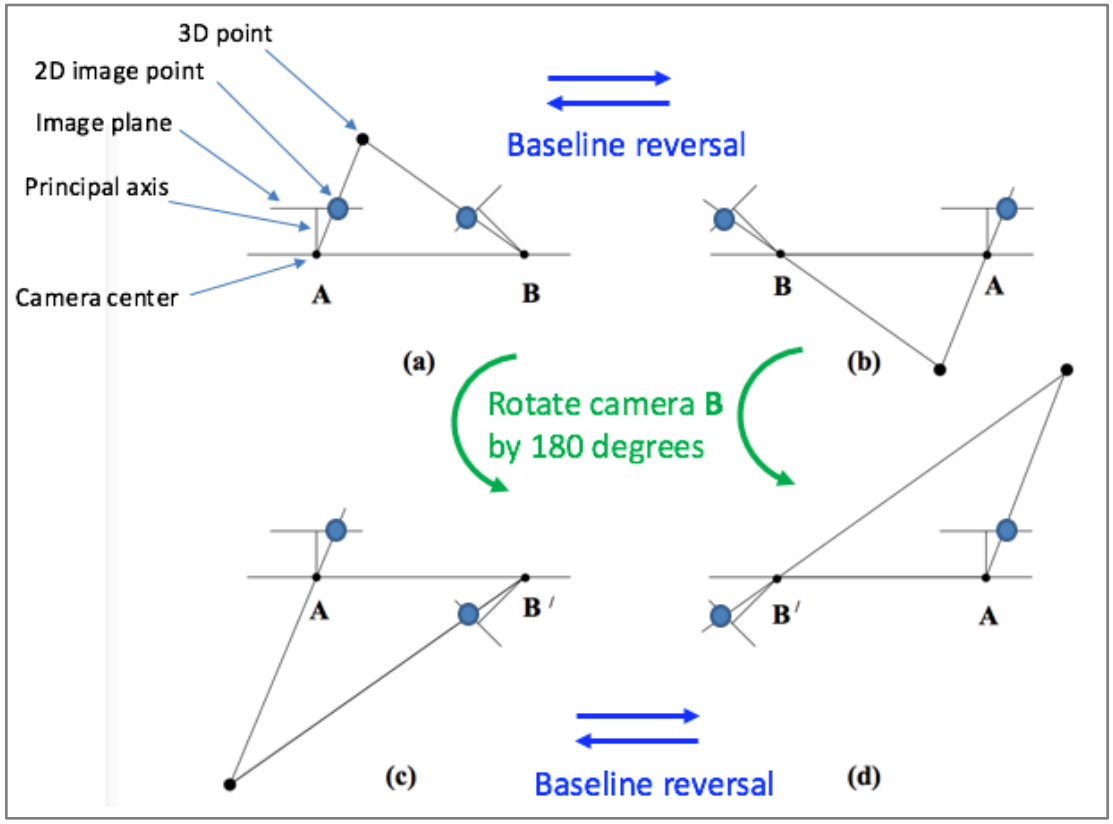
- In reality, instead of solving $\mathbf{Af} = 0$, we seek $\mathbf{f}$ to minimize $\|\mathbf{Af}\|$, using SVD.

RANSAC to Estimate Fundamental Matrix

- For *N* times
  - Pick 8 points
  - Compute a solution for $\mathbf{F}$ using these 8 points
  - Count number of inliers with $\mathbf{x}_1^T \mathbf{F} \mathbf{x}_2$ close to 0
- Pick the one with the largest number of inliers

Motion from correspondences

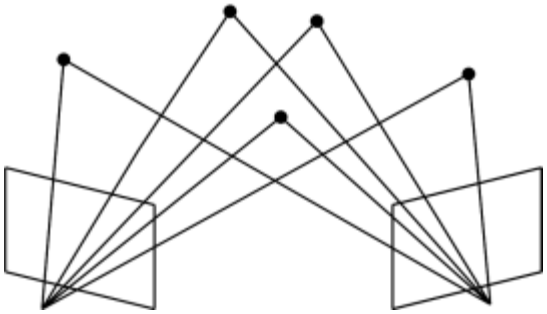
- Use 8-point algorithm to estimate F
- Get E from F: $\mathbf{E} = \mathbf{K}_2{}^\top \mathbf{F} \mathbf{K}_1$
- Decompose E into skew-symmetric and rotation matrices: $\mathbf{E} = [\mathbf{t}]_\times \mathbf{R}$
- Four possible solutions



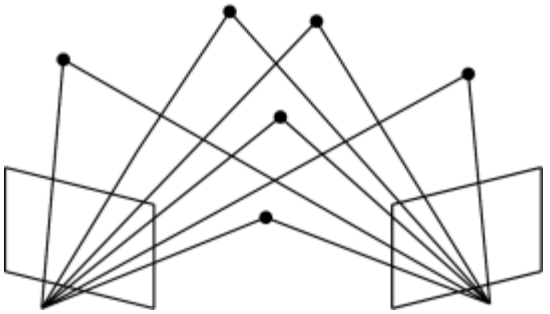
Minimal Problems in Computer Vision

- There is a large zoo of minimal problems that have known solutions

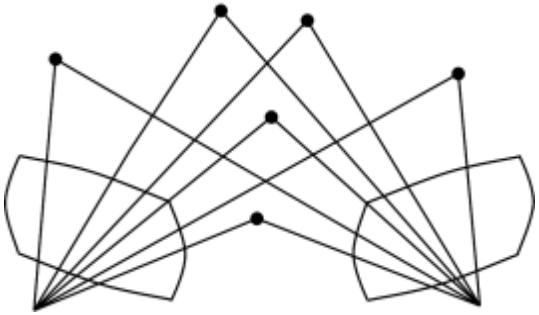
5-point relative pose with known K



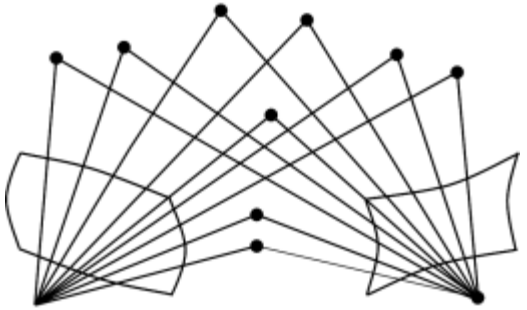
6-point relative pose with unknown f



6-point relative pose with unknown r



9-point relative pose with unknown and different f, r



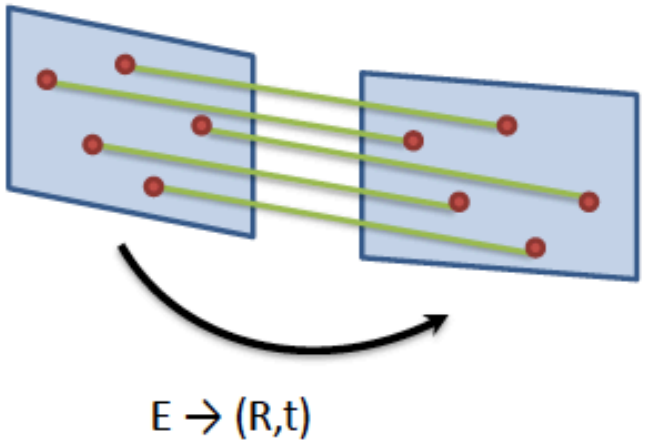
Calibrated Relative Pose Estimation

- Camera 1: $K_1[I \mid 0]$, camera 2: $K_2[R \mid t]$
- Fundamental matrix: $F \equiv K_2^{-\top}[t]_\times R K_1^{-1}$

$$[t]_\times \equiv \begin{bmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix}$$

Essential matrix, *E*

- Five degrees of freedom for the essential matrix
- Goal: estimate E *efficiently* from 5 correspondences

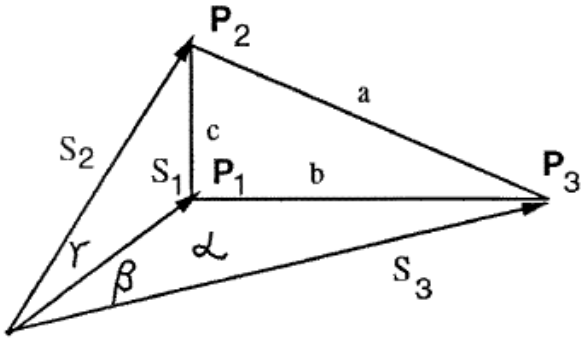


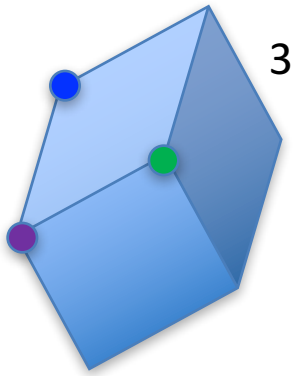
10 cubic equations in entries of *E* (not all independent)

Reduce to single degree 10 polynomial in 1 variable.

Three-Point Absolute Pose Estimation

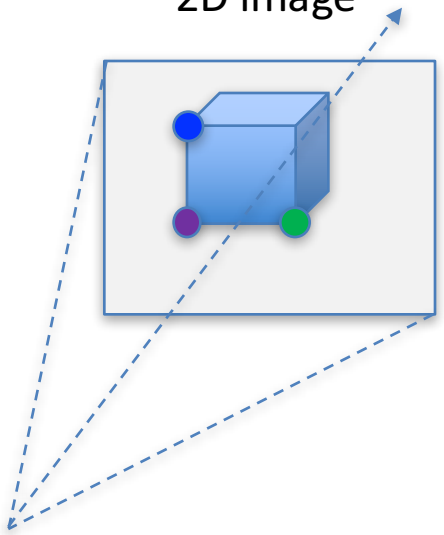
- Find camera pose from given 3D-2D correspondences
- Minimal case is 3 points: 6 degrees of freedom, 2 constraints per point (x, y)
- Given p_1, p_2, p_3 in world coordinates, find positions in camera coordinates





3D scene

2D image

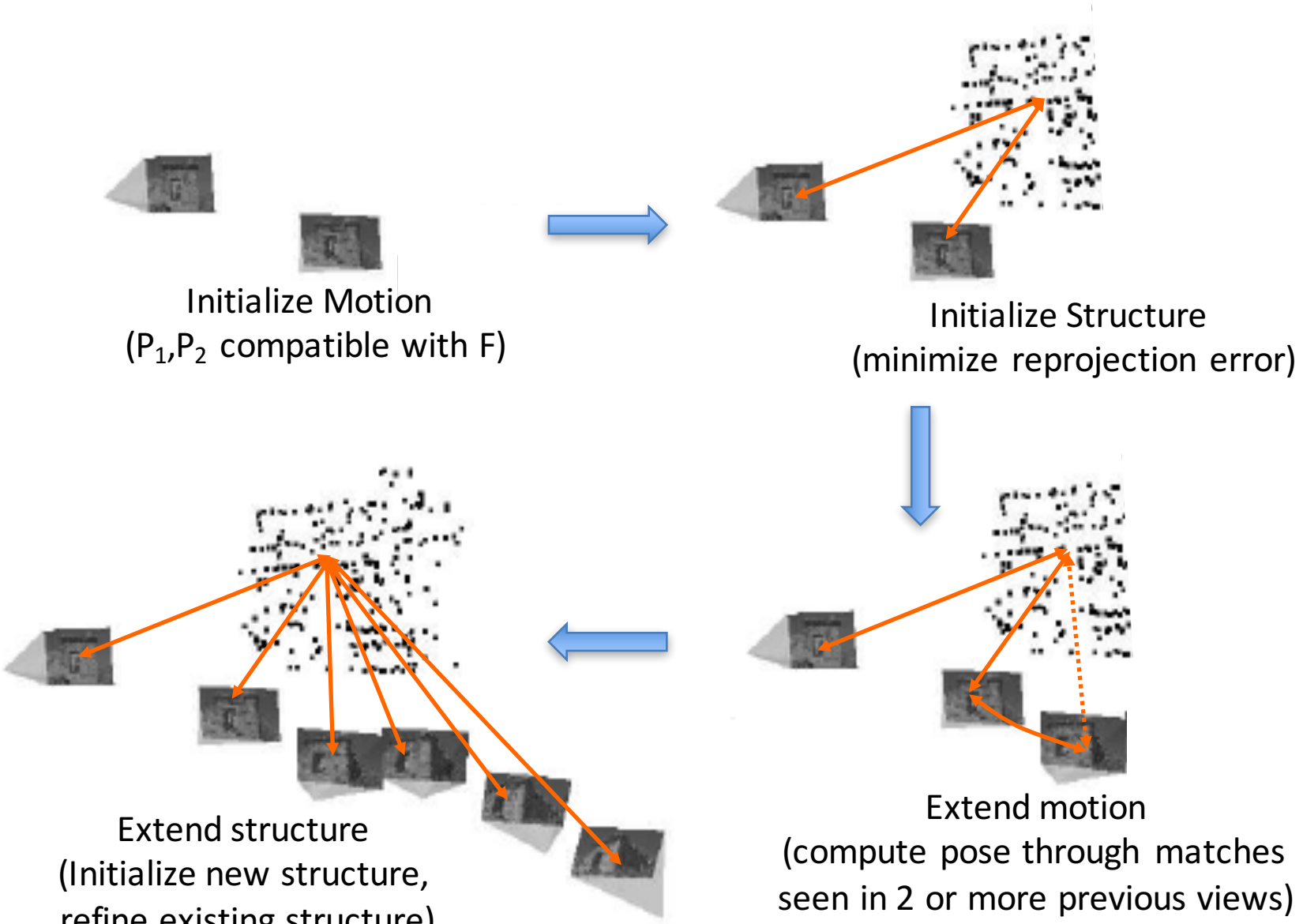


3 quadratic equations in three variables of R

Reduce to single degree 4 polynomial in 1 variable.

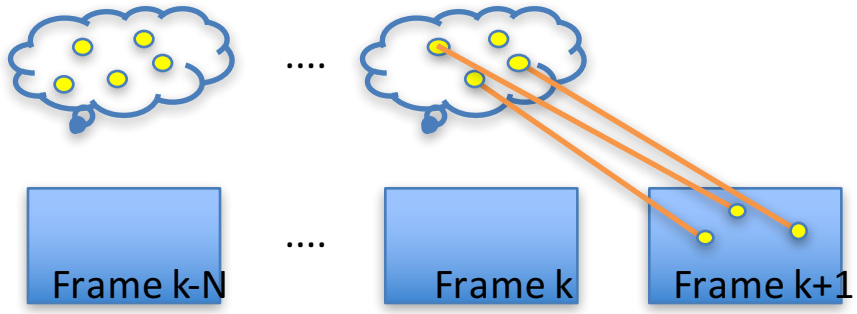
Determine how to move camera, such that corresponding 2D-3D points align

Sequential Structure from Motion



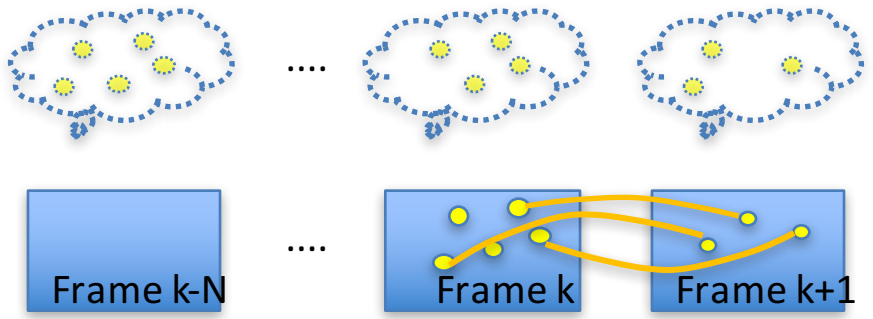
Real-Time SFM: Steady-State

- Long feature tracks can be constructed (same 3D point visible in many view)
- Repeated bundle adjustment and outlier removal
- Cheaper, less error to solve 3-point absolute pose



3D-2D correspondences

- Rely on per-frame matches, shorter baselines for reconstruction
- New points in every frame, less chances for outlier removal
- More expensive and higher error to solve 5-point relative pose

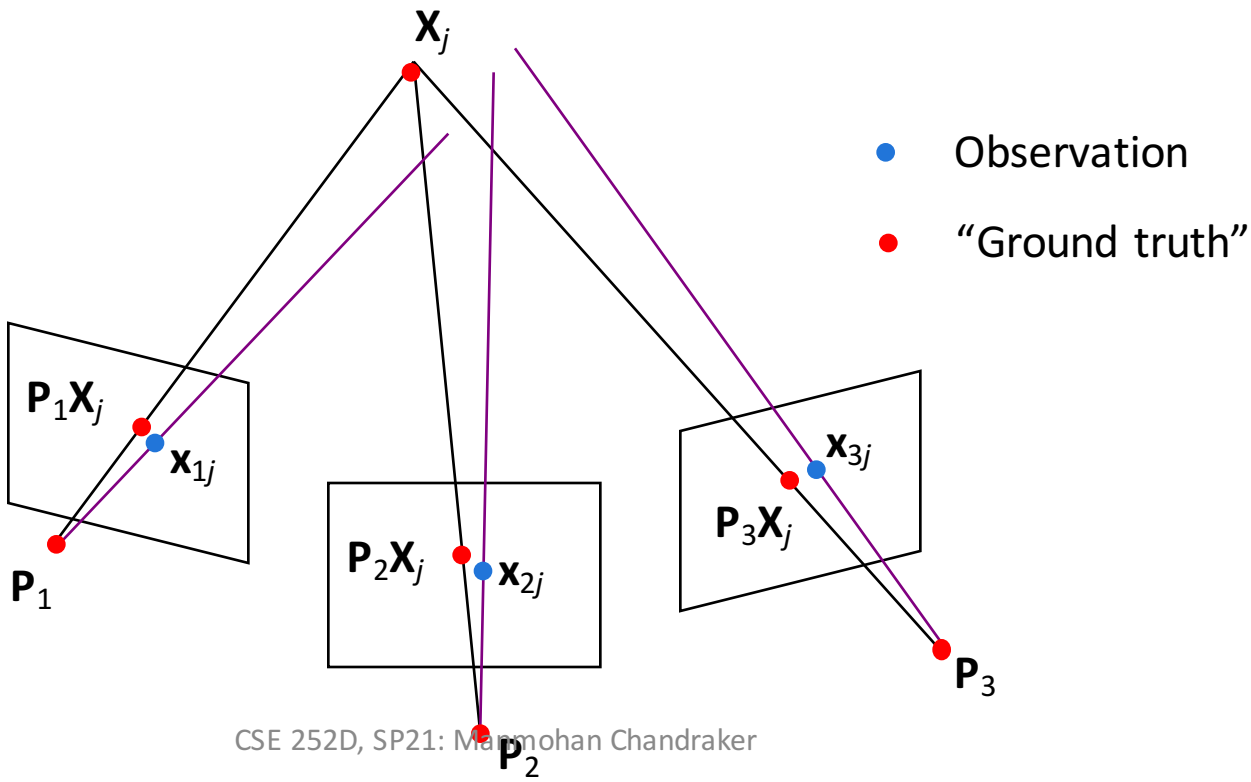


2D-2D correspondences

Bundle Adjustment

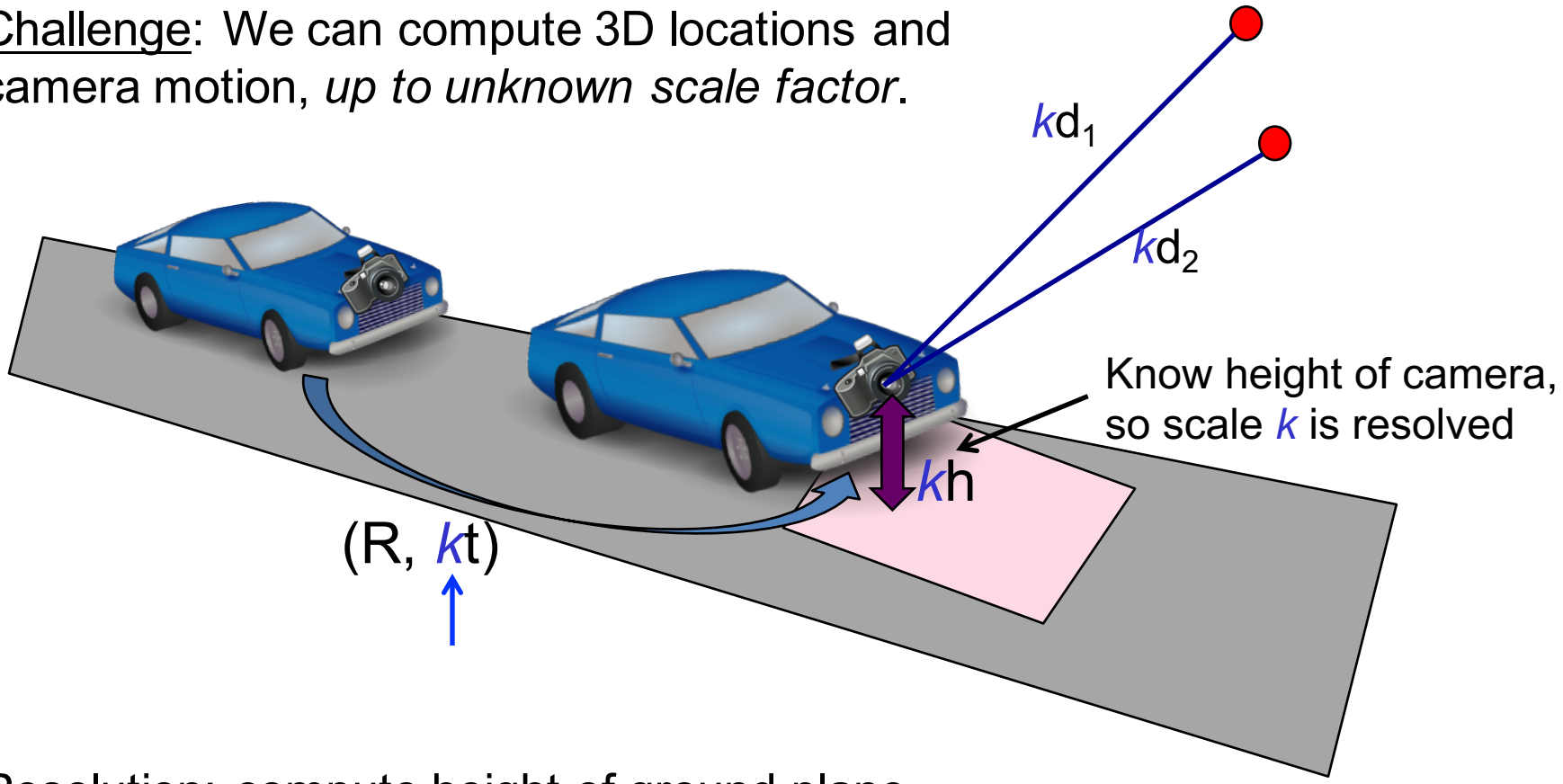
- Non-linear method for refining structure and motion
- Minimizing reprojection error

$$Error\,(P,X) = \sum_{Cameras\ Pi} \sum_{Points\ Xj} |x_{ij} - PiXj|^2$$



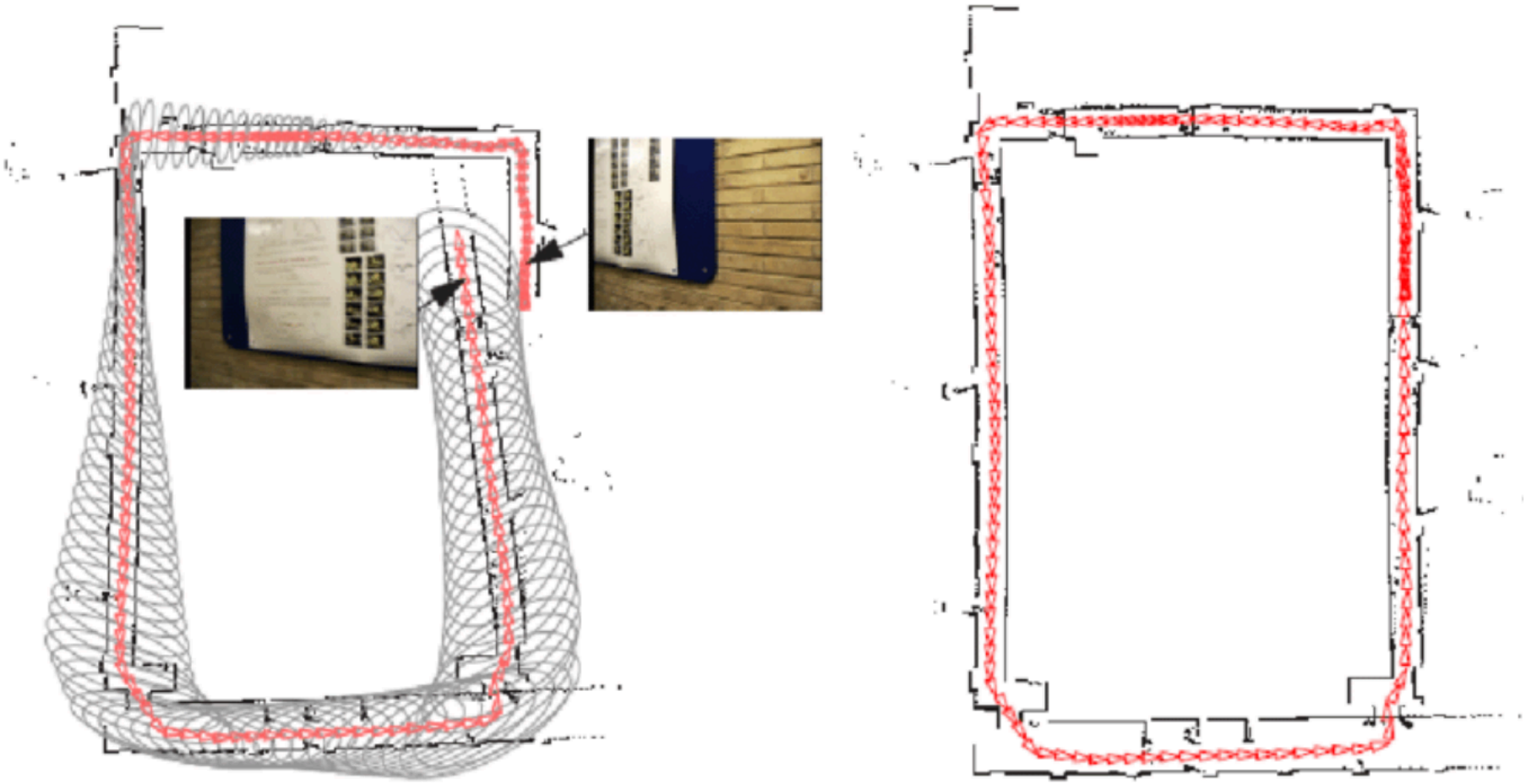
Scale Drift Correction

Challenge: We can compute 3D locations and camera motion, *up to unknown scale factor*.



Resolution: compute height of ground plane.

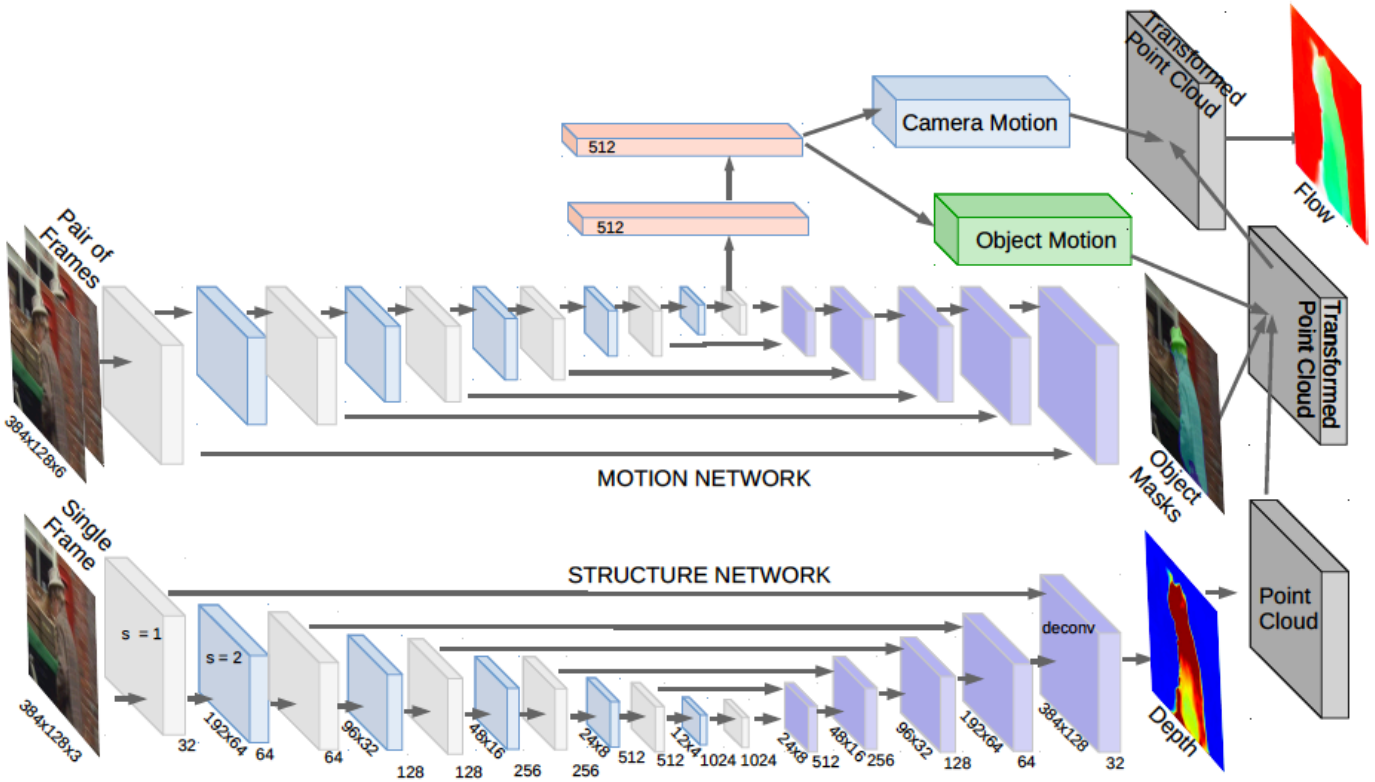
Loop Closure for Scale Drift Correction



Recognize whether same image is observed as in some previous frame.

General Recipe to Learn Structure and Motion

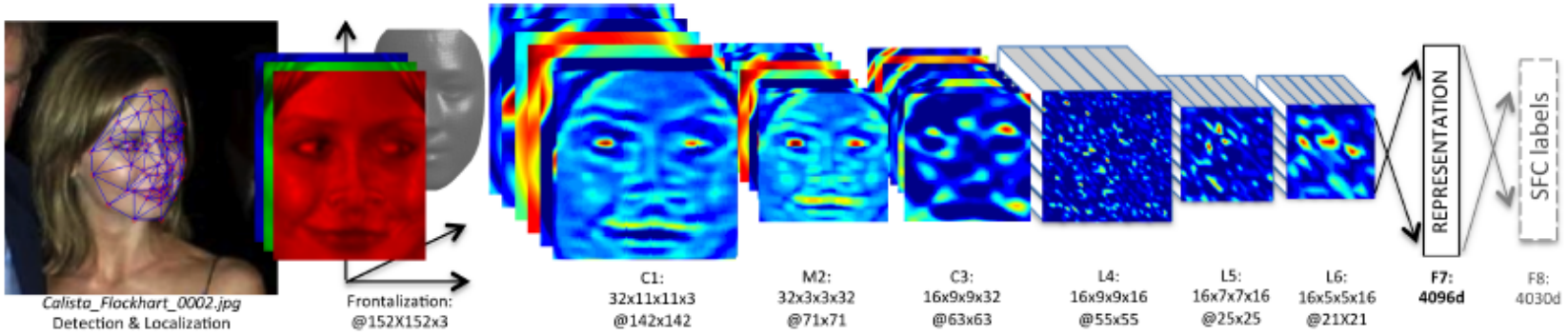
- Estimate depths (convert to 3D points given calibration) in frame t
- Estimate motion from frame t to t+1 for background and objects
- Project estimated 3D points to frame t+1 using the estimated motions
- Use a consistency condition to declare matches as good

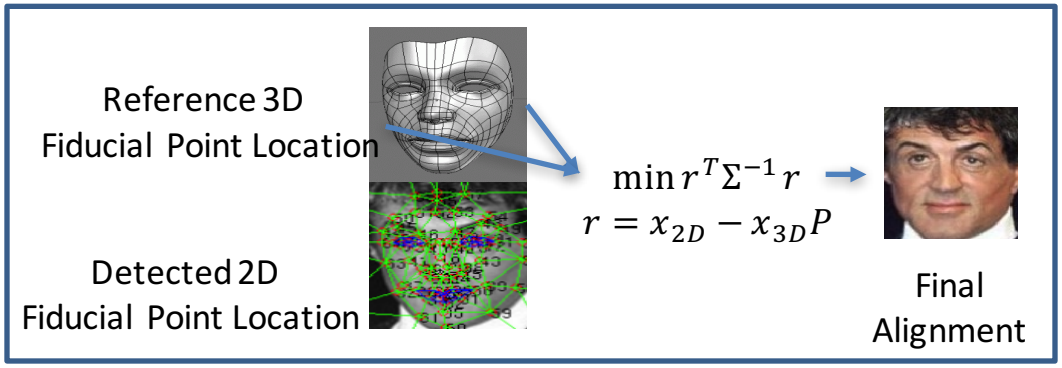


[SFM-Net, ICCV 2017]

Face Recognition

DeepFace



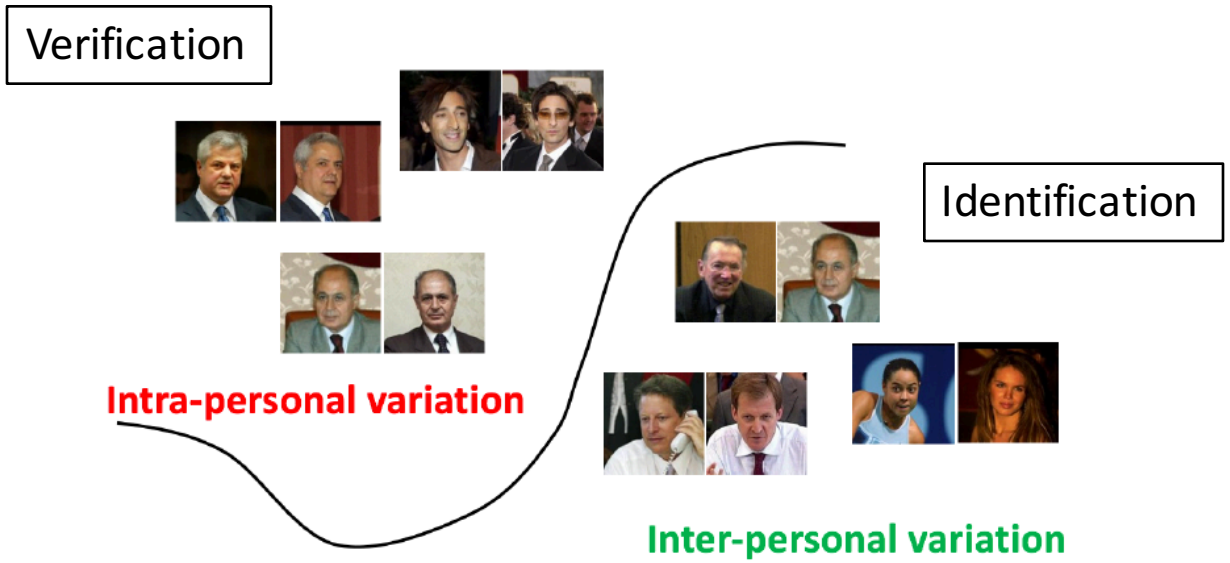


Layer 4-6: Intuition

- Apply filters to different locations on the map
- Similar to a conv. layer but spatially dependent
- F8 calculates probability with softmax $p_k = \frac{\exp(o_k)}{\sum_h \exp(o_h)}$
- Cross-entropy loss function: $L = -\sum_k \log(p_k)$

[Taigman et al., DeepFace, CVPR 2014]

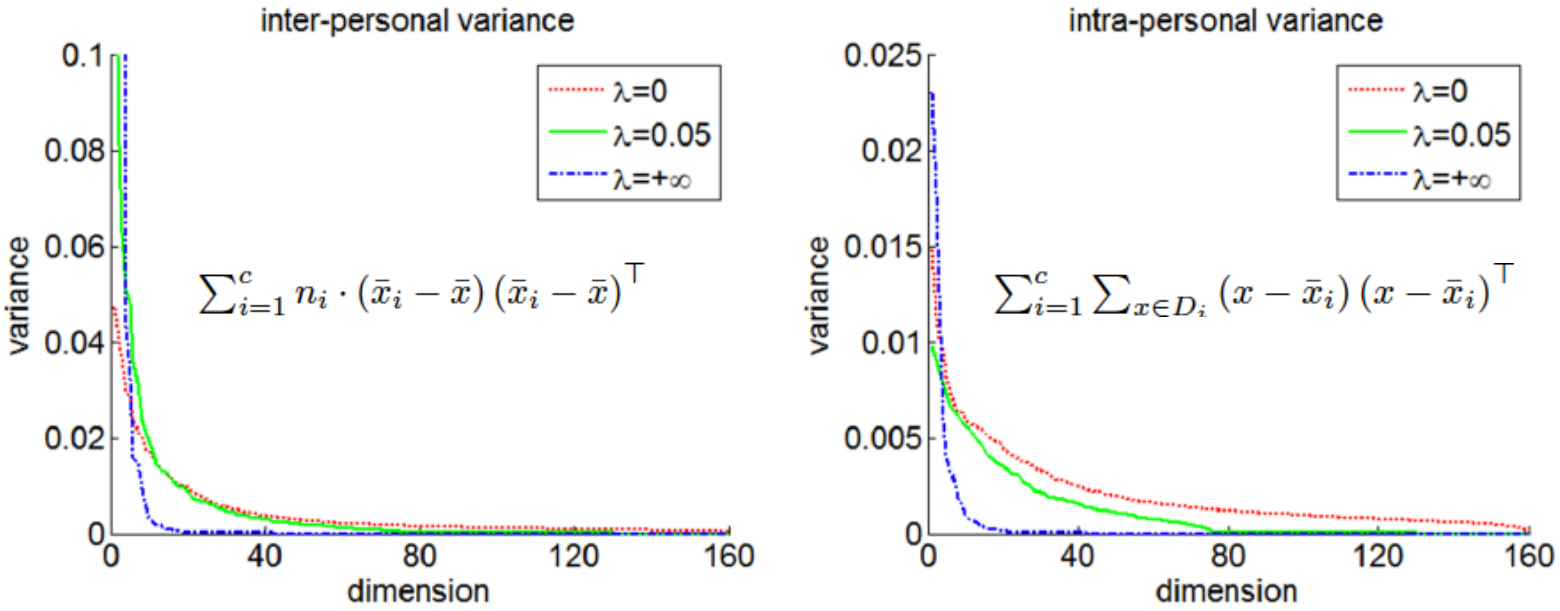
DeepID2: Verification and Identification Signals



- Verification:
  - Match two images of an individual across large appearance variations
  - Favors tight clusters for each identity

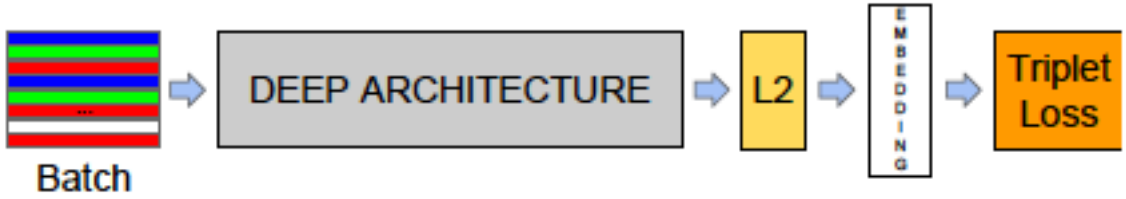
- Identification:
  - Distinguish images of one identity from another identity
  - Favors large distance between clusters

Balancing Identification and Verification



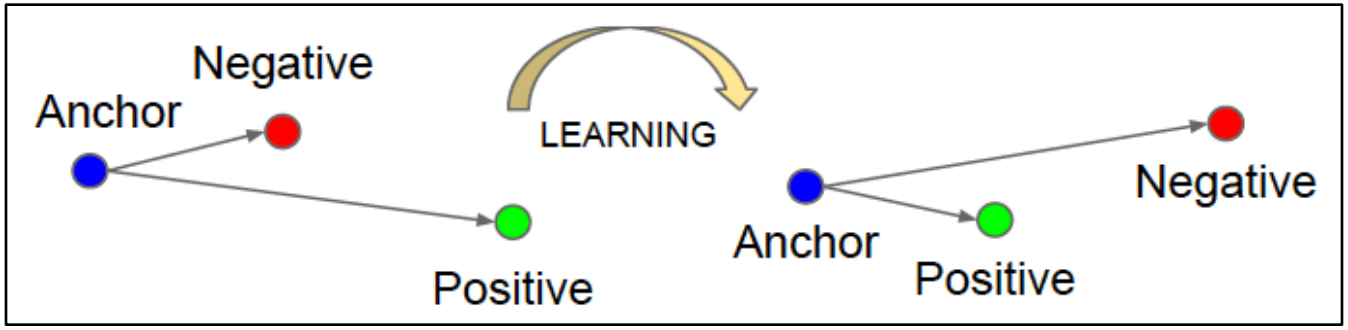
- When only identification signal is used $(\lambda = 0)$:
  - High diversity in both inter-personal and intra-personal features
  - Good for identification since it helps distinguish different identities
  - But large intra-personal variance is noise for verification
- When only verification signal is used (λ approaches $+\infty$):
  - Both intra-personal and inter-personal variance collapse to few directions
  - Good for verification, but cannot distinguish many classes in identification

Learn an Embedding for Face Recognition



- Goal: learn d-dimensional embedding *f(x)* for face image *x*
- Constrain embedding to lie on unit sphere: $\|f(x)\|_2 = 1$

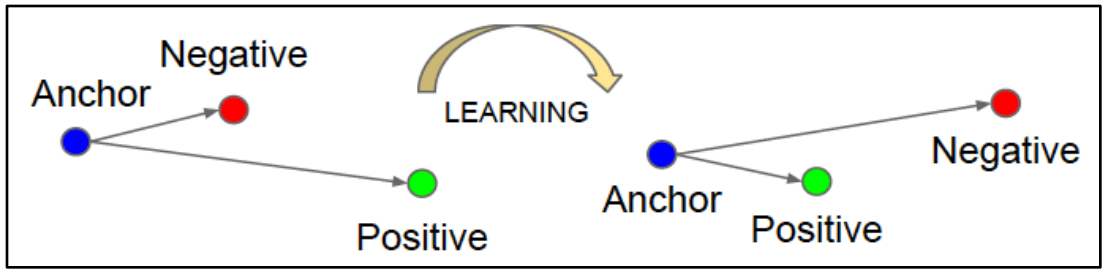
- Goal for triplet loss:
  - Minimize distance between anchor and a positive (from same class)
  - Maximize distance between anchor and a negative (from different classes)



[Schroff et al., FaceNet, CVPR 2015]

Triplet Loss for Training

- Goal for triplet loss:
  - Minimize distance between anchor image x_i^a and a positive x_i^p
  - Maximize distance between anchor x_i^a and a negative x_i^n



$$\|f(x_i^a) - f(x_i^p)\|_2^2 + \alpha < \|f(x_i^a) - f(x_i^n)\|_2^2 \quad , \quad \forall \left(f(x_i^a), f(x_i^p), f(x_i^n)\right) \in \mathcal{T}$$

- Total loss to minimize:

$$\sum_i^N \left[\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha\right]_+$$

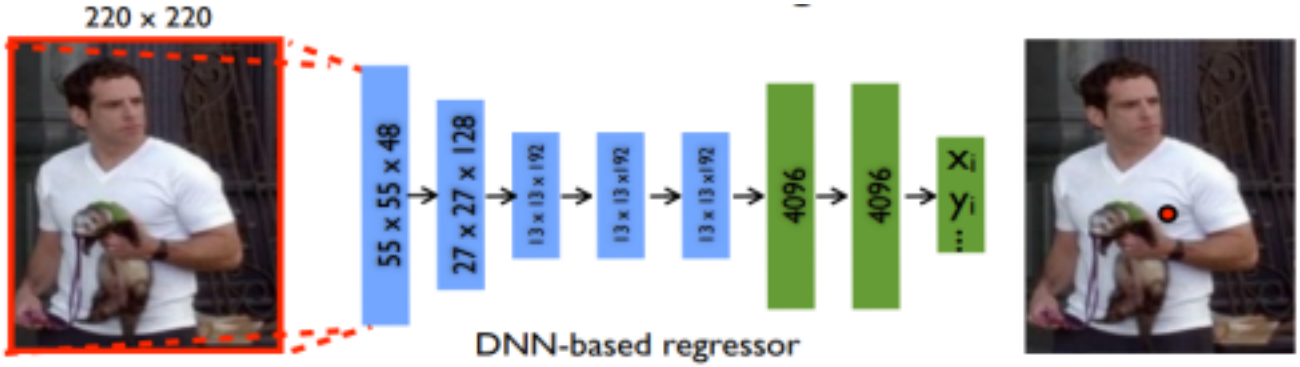
- Challenge: too many triplets satisfy the margin easily
- Need to select hard examples that are active and improve the model

Human Pose Estimation

Deep Networks for Holistic Pose Estimation

- Advantages:
  - Simple, yet holistic
  - No need to define losses that capture interactions
  - Instead, all hidden layers are shared by joint regressors

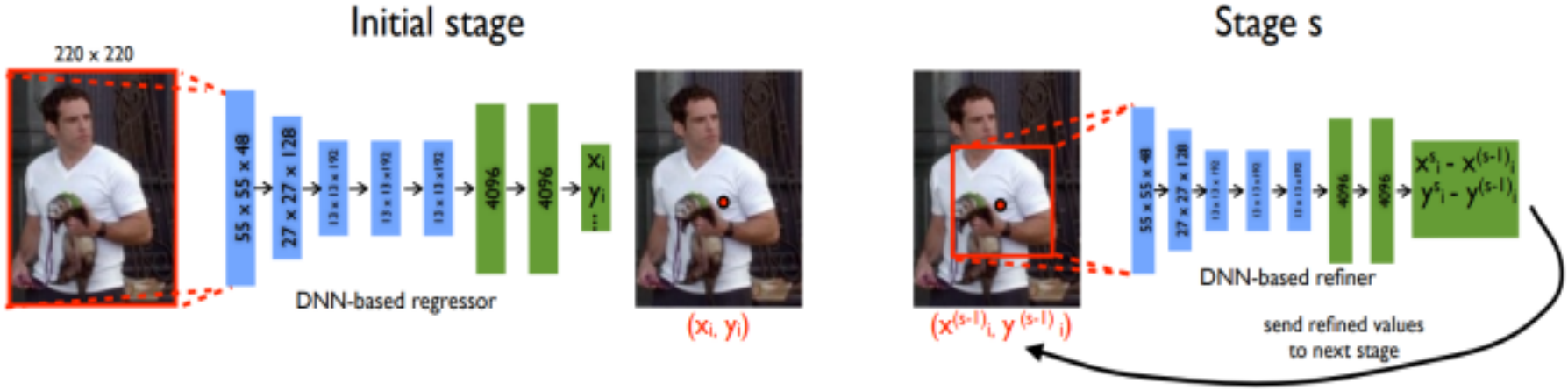
- Disadvantages:
  - Limited ability to consider details



[Toshev and Szegedy, DeepPose]

Cascade of Regressors

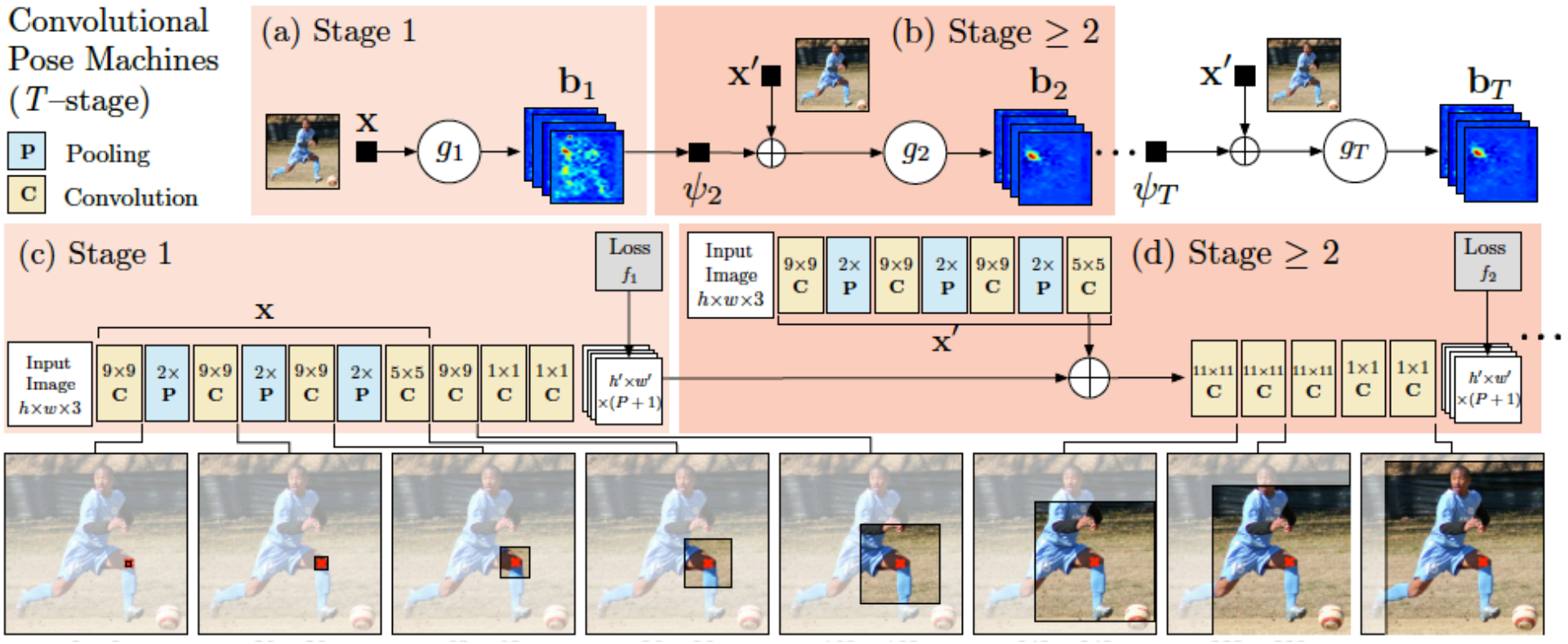
- Advantages:
  - Simple, yet holistic
  - No need to define losses that capture interactions
  - Instead, all hidden layers are shared by joint regressors
  - Increasingly detailed prediction along cascade stages
- Disadvantages:
  - ~~Limited ability to consider details~~
  - Depends on quality of initial prediction
  - Not enough modeling of spatial structure
  - Cannot reason about occluded parts, or those outside window



[Toshev and Szegedy, DeepPose]

Convolutional Pose Machines

- Goal: learn complex and long-range interactions
- Larger receptive field sizes through successive stages
- Intermediate supervision to prevent vanishing gradients

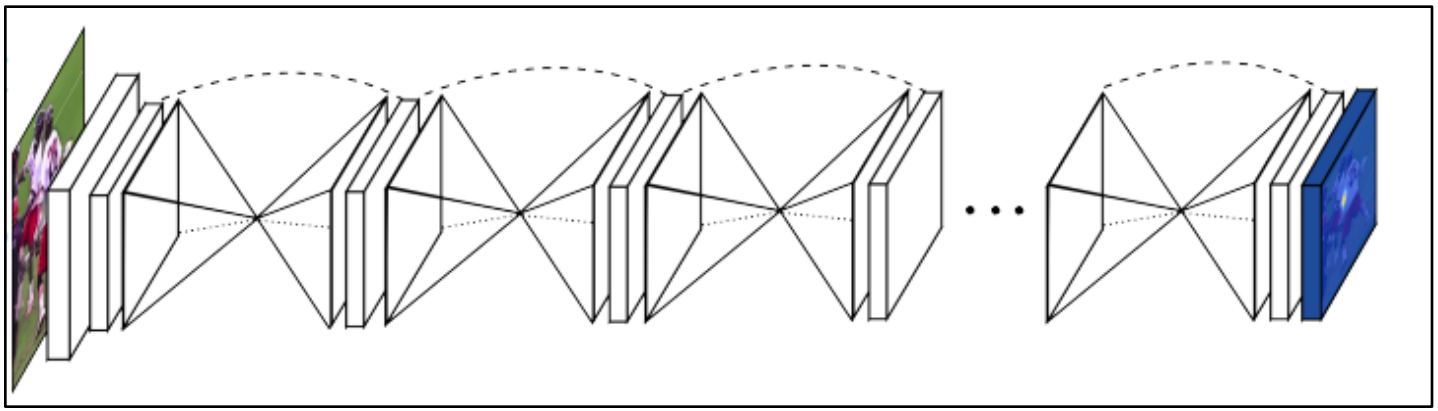


[Wei et al., Convolutional Pose Machines]

Stacked Hourglass Network

- Bottom-up reasoning: Convolution and max-pooling to very low 4x4 resolution
- Top-down reasoning: Upsample and combine with skip connection

- Multiple iterative stages allow refinement
- Each stage does full bottom-up and top-down processing (no weights shared)
- Can apply intermediate supervision for each stage
  - Subsequent hourglass modules can reassess high-order spatial relations
  - Ask network to repeatedly reason across scales

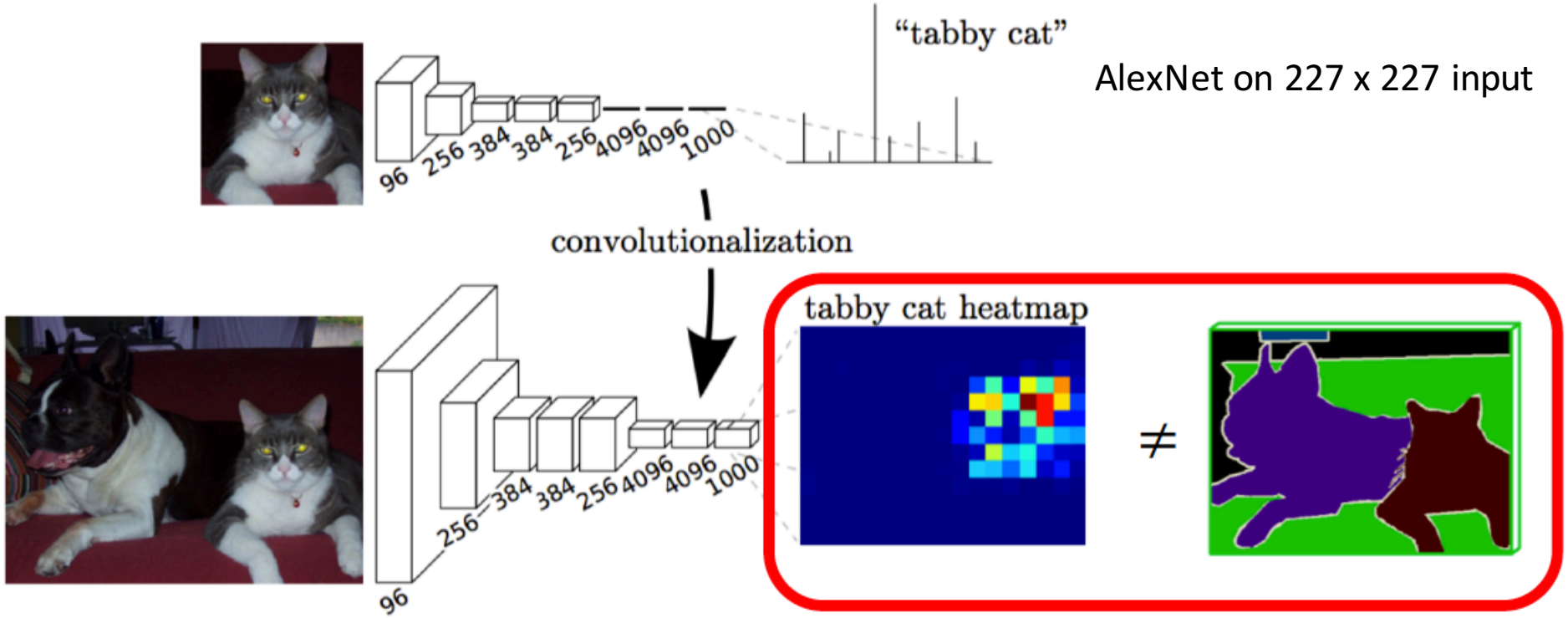


[Newell et al., Stacked Hourglass Networks]

Semantic Segmentation

Fully Convolutional Network

Fully-connected layer with *k* units = Convolution layer with *k* filters of size that covers input



AlexNet on 227 x 227 input

Given 500 x 500 image, slide FCN with stride 32 to get 10 x 10 output.

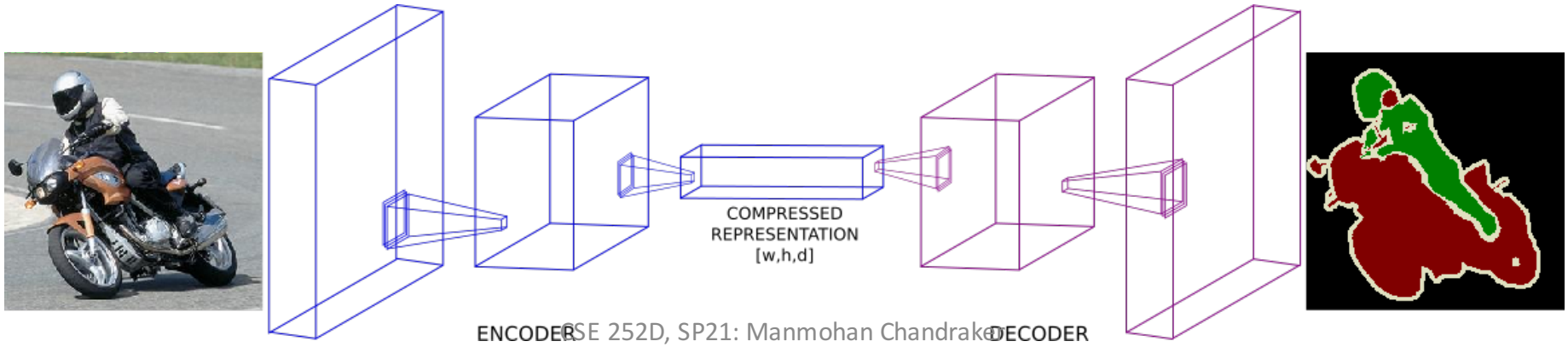
We want a segmentation output at image resolution.

Output Going to Image Resolution

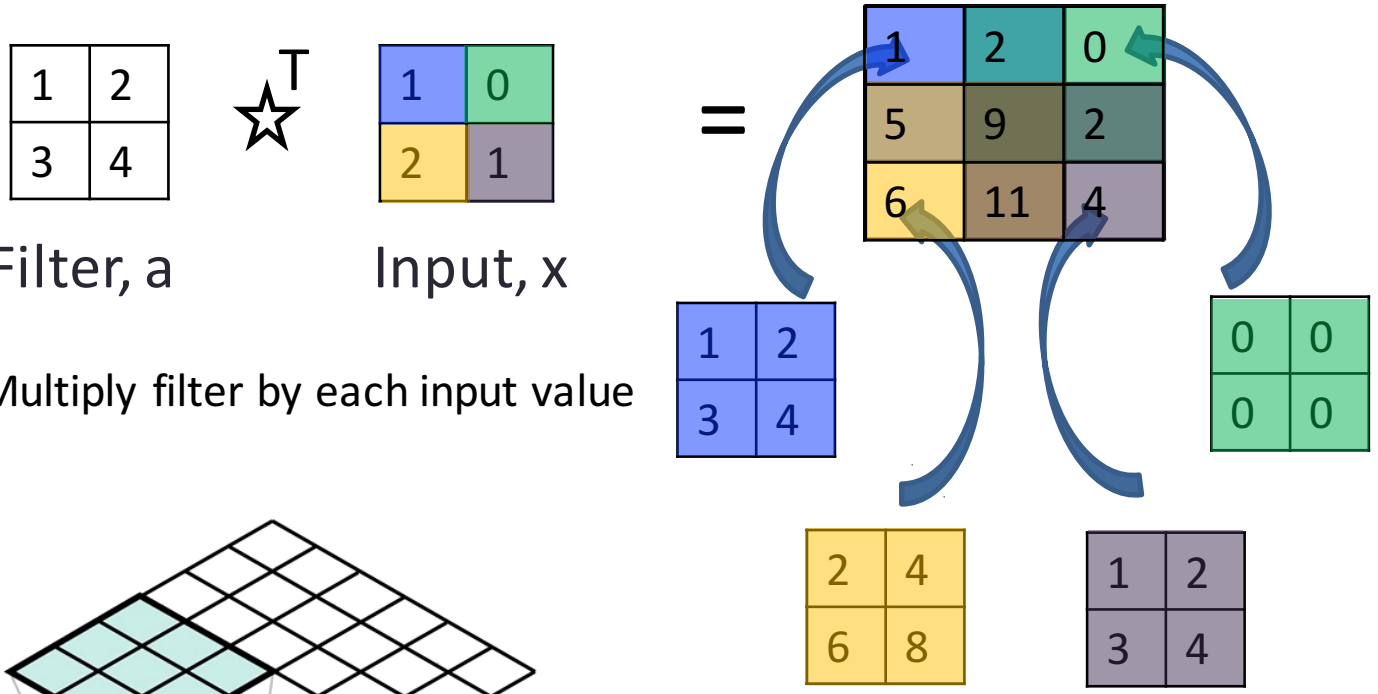
- Encoder aggressively pools and subsamples image
- Necessary to capture context information which is necessary for segmentation
- But spatial detail is also necessary

- Goal for decoder: obtain output at image *resolution*
- Goal for decoder: recover *detail* in encoder feature maps before subsampling

- Two approaches:
  - Transposed convolutions (FCN)
  - Unpooling (SegNet)



Transposed Convolutions

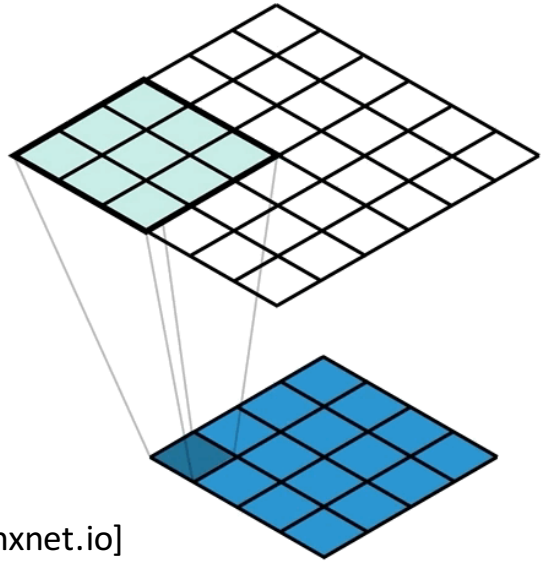


Filter, a

Input, x

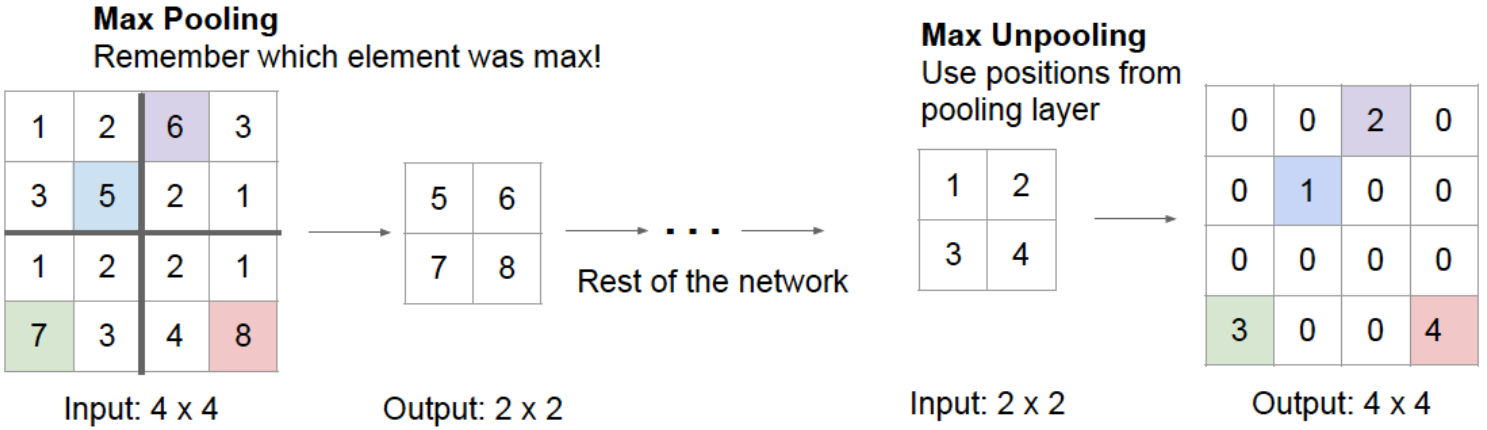
Multiply filter by each input value

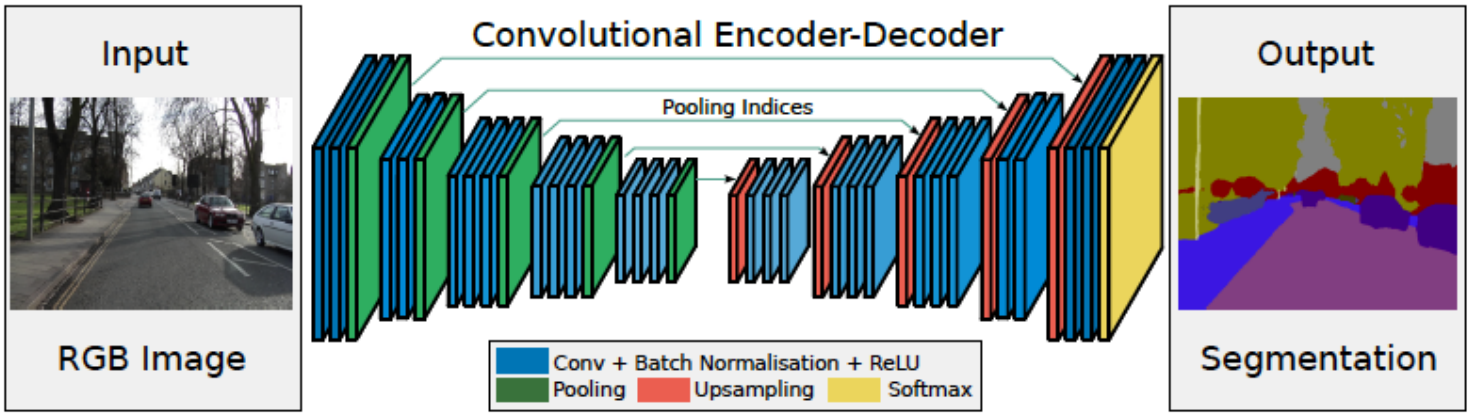
Tile the scaled filter at output locations
Add overlapping values



[Thom Lane, mxnet.io]

Refinement: Unpooling followed by convolution





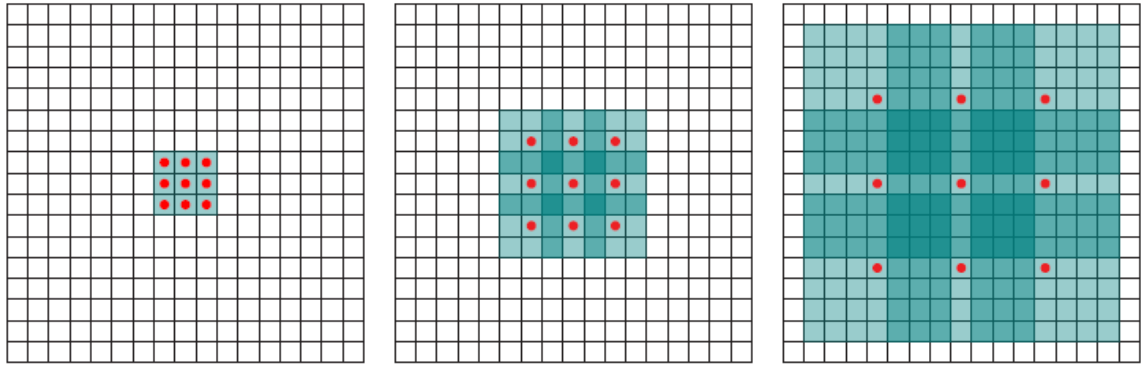
- Store 2x2 pooling index with 2 bits, as opposed to floating point feature map
- Significant memory reduction at inference time

 [Badrinarayanan et al., SegNet]

Wide Receptive Fields

- Most networks have similar encoders (inspired by classification networks)
- Downsample to save memory and obtain large receptive fields

- Consider not downsampling features, but still achieve large receptive fields
  - Once downsampled, signal might be lost for small objects
  - Hard to recover by subsequent layers during training
  - Challenge: maintain spatial resolution along with a large receptive field
  - Solution: use dilated convolutions



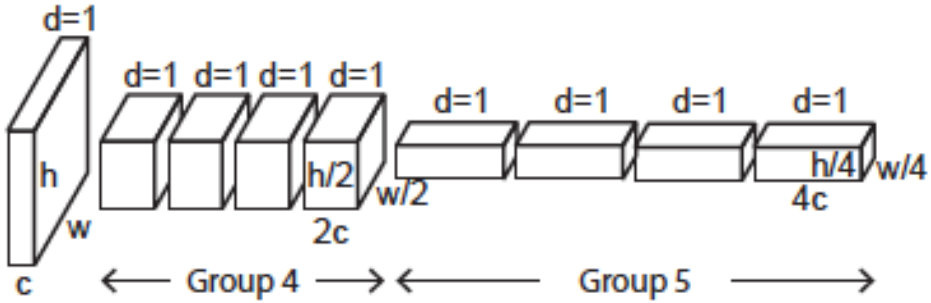
(i = 0) (i = 1) (i = 2)

$(l = 2^i)$

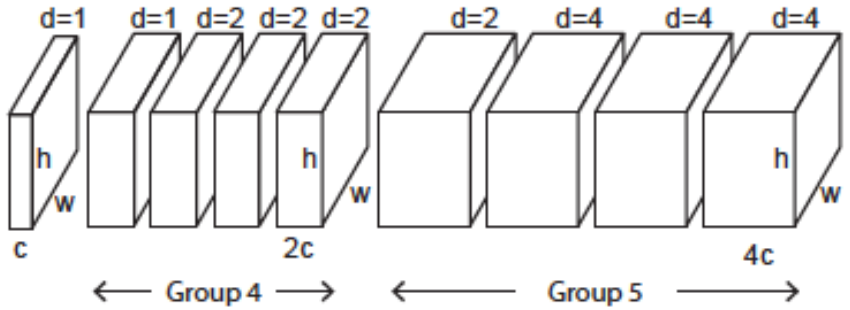
- Receptive field increases with greater dilation factors
- Number of parameters remains the same

Dilated Residual Network

- Goals: Exploit the power of residual networks with advantages of dilation
  - Preserve spatial resolution of feature maps
  - Provide training signals that densely cover the input field
  - Backpropagation can now learn to preserve small but salient details
- ResNet uses stride 2 to downsample from block G_k to G_{k+1}
- Do not use stride to downsample, maintain resolution
- Dilate subsequent layers by another factor of 2 to maintain receptive field
- But two problems with DRN:
  - Gridding artifacts
  - Memory consumption



ResNet



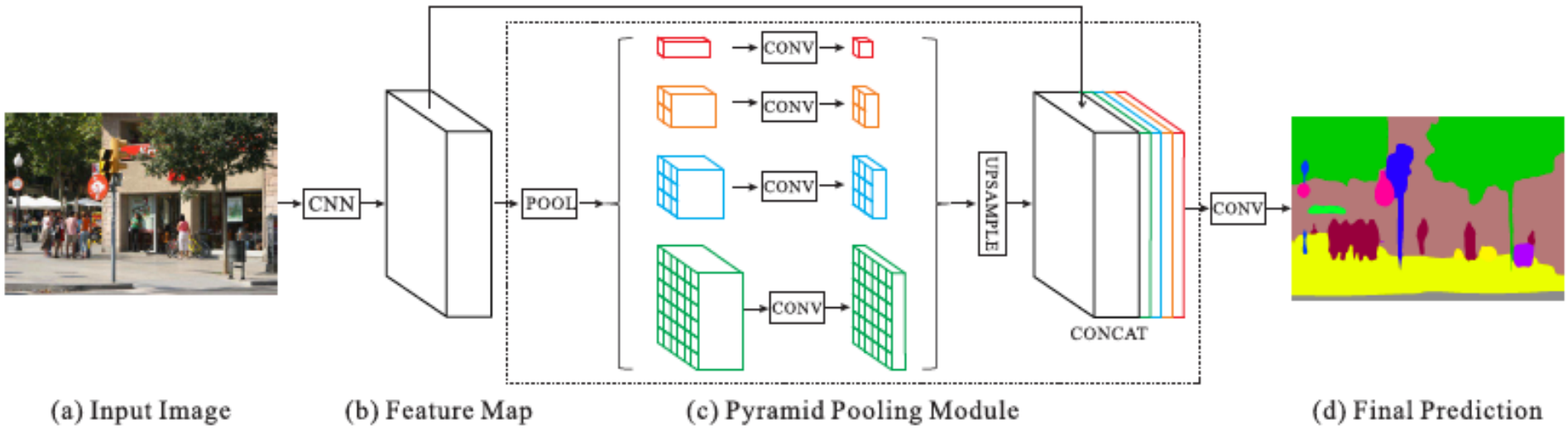
Dilated ResNet

[Yu et al., Dilated Residual Networks]

More Context Information: Pyramid Pooling

- Role in segmentation: Extract both global and regional context
  - A hierarchical global prior, with information across scales and regions

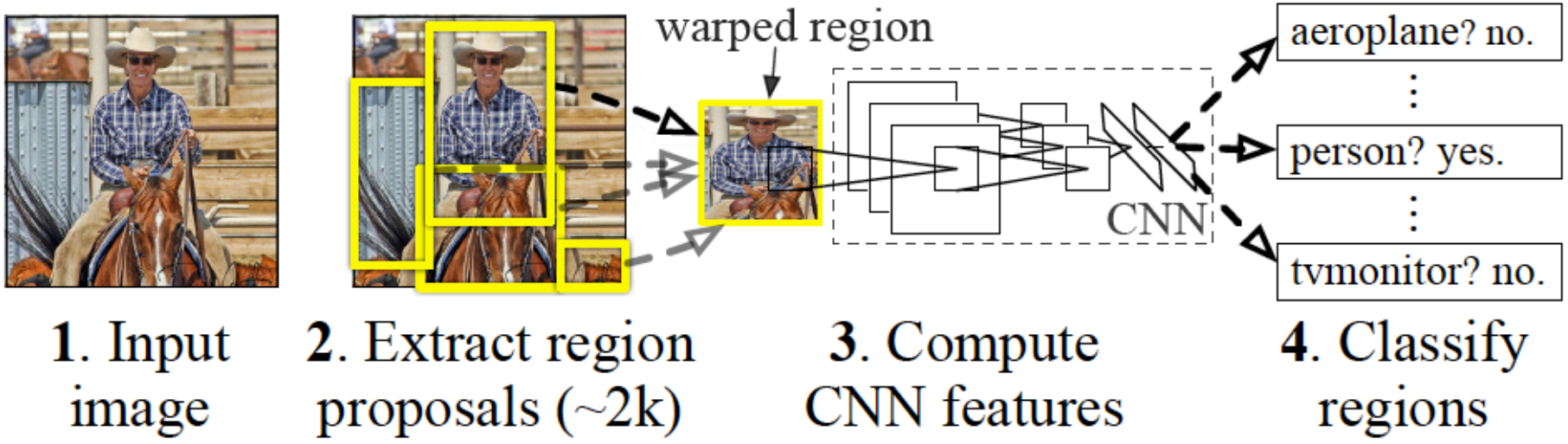
- Fuse features in several different pyramid scales
  - Pool input feature map into hierarchy of context features
  - Do 1x1 convolution to ensure each pyramid level gets equal weight
  - Upsample to original resolution and concatenate



[Zhao et al., PSPNet]

Object Detection

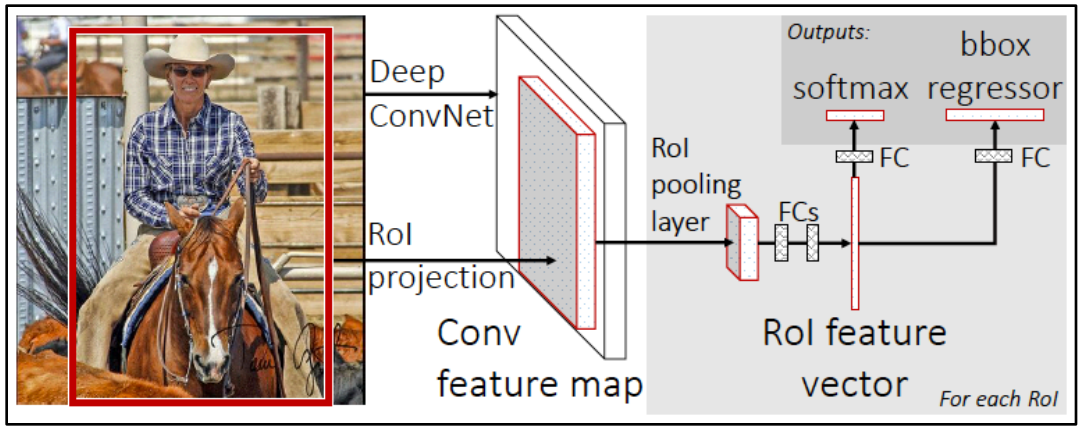
R-CNN

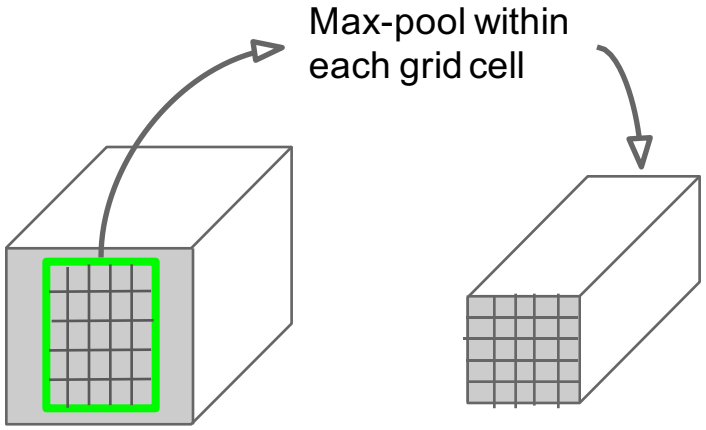


- Expensive
  - No feature reuse
  - Full forward pass of CNN for each region proposal
- Classification and regression are disjoint from feature extraction
  - CNN features not updated together with detection
- Complex training pipeline

[Girshick et al., Rich feature hierarchies]

Fast R-CNN





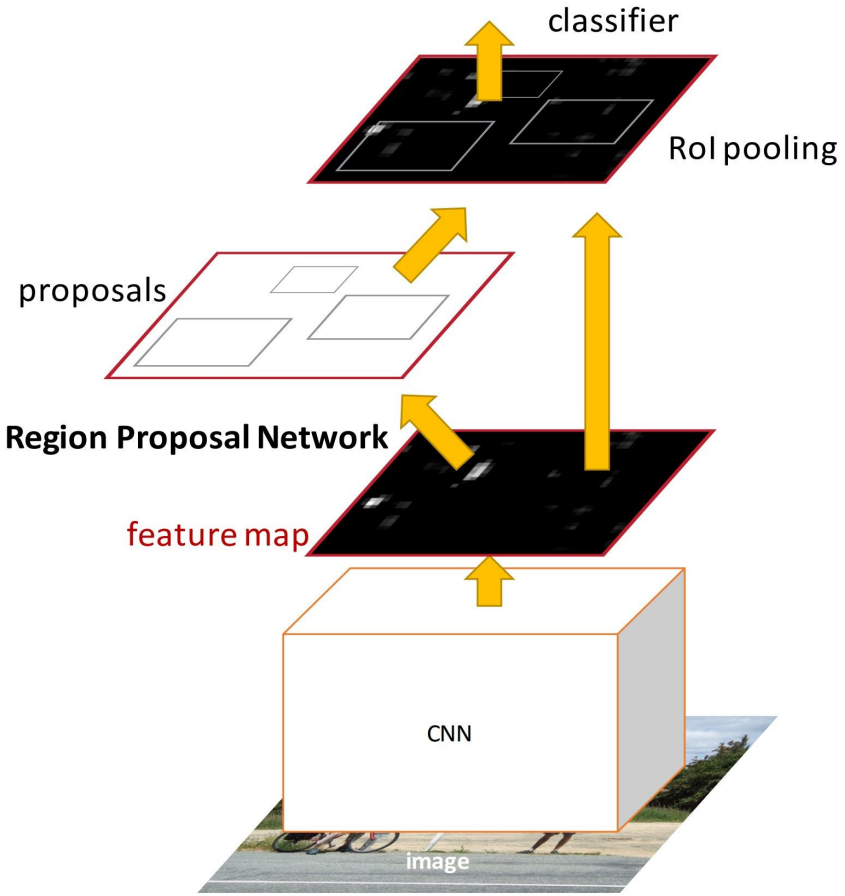
Hi-res conv features:
C x H x W
with region proposal

RoI conv features:
Fixed size C x h x w
to classifier

- RoI pooling: connects convolutional layers to classifier
  - Features and classifier trained together
- Feature extraction just once for the whole image
  - Faster inference and training
- Proposals still from external mechanism

[Girschick, Fast R-CNN]

Faster R-CNN

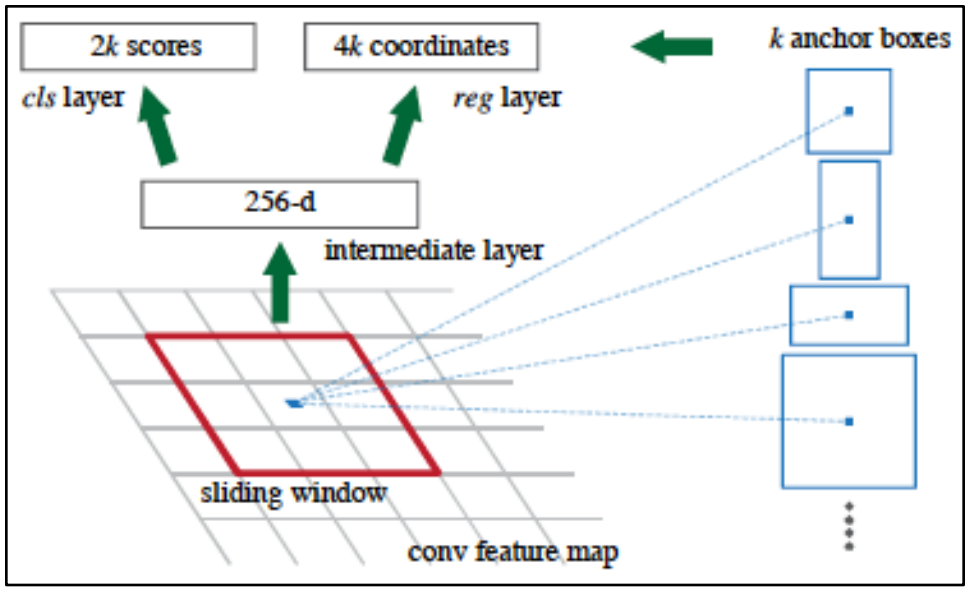


- Proposals still from separate mechanism in Fast R-CNN
- Insert a **Region Proposal Network (RPN)** after last convolutional layer
- RPN trained to produce region proposals directly, no need for external region proposals!
- After RPN, use RoI Pooling and an upstream classifier and regressor just like Fast R-CNN

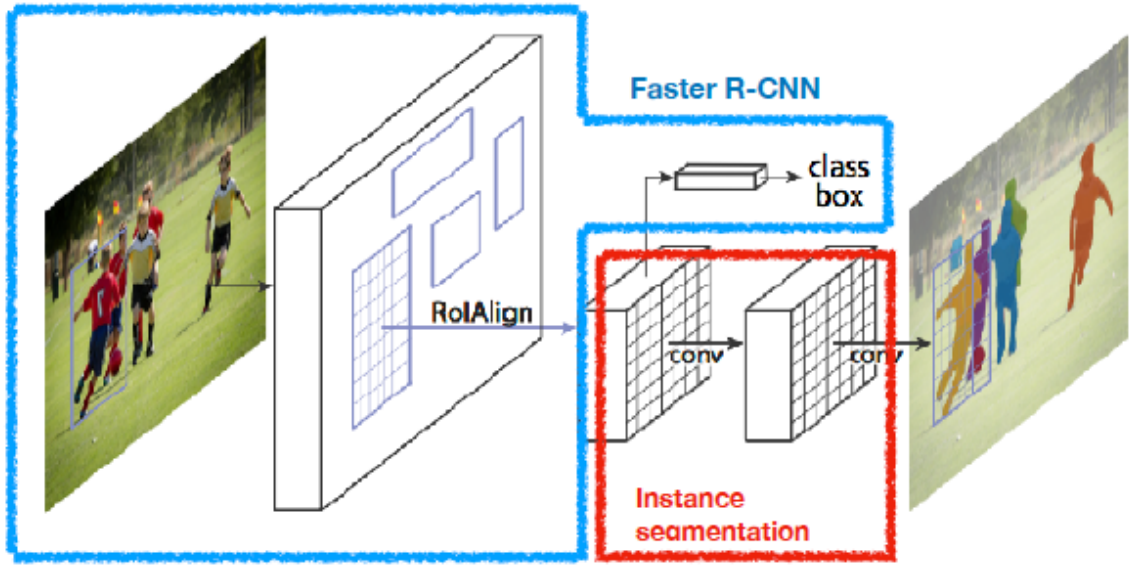
[Ren et al., Faster R-CNN]

Faster R-CNN: Anchors

- Slide a small window on the feature map
- Build a small network for:
  - classifying object or not-object
  - regressing bounding box locations
- Position of sliding window gives location information with respect to the image
- Box regression gives finer localization with respect to this sliding window
- Anchors: a set of reference positions on the feature map
  - Anchor with high ground truth overlap responsible for regressing position
  - Determines reference and spatial extent for predicting object



Instance Segmentation

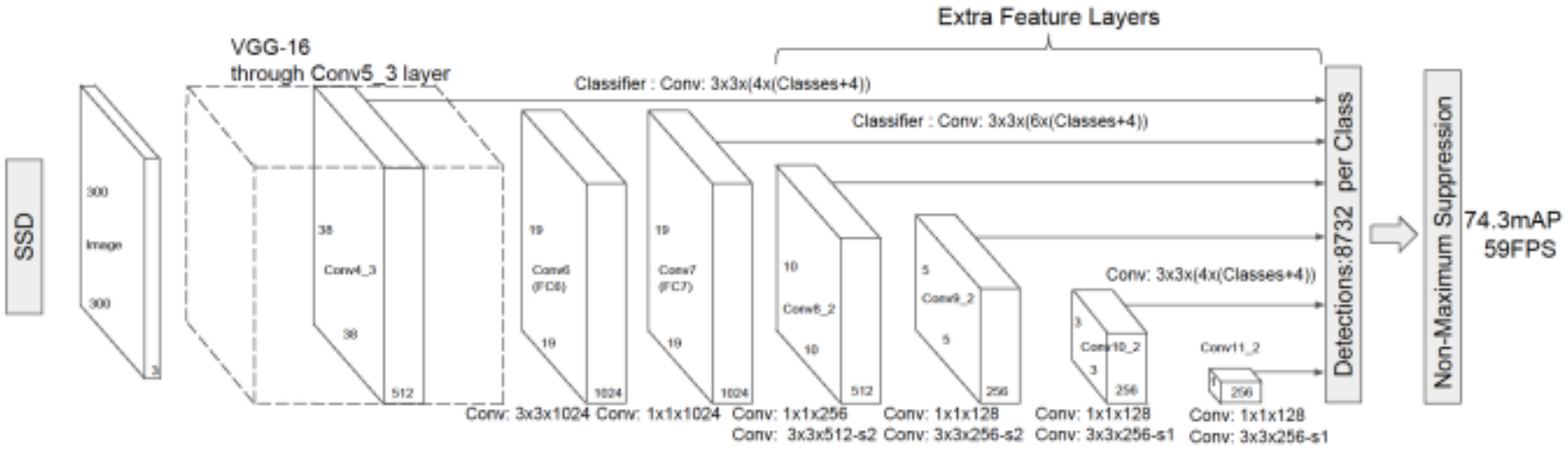


- Two choices:
  - Predict instance first, then classify
  - Segment first, then break into instances
  - Mask R-CNN is instance-first
- Decouple mask (FCN) and class prediction (Faster R-CNN)

Instance segmentation

[He et al., Mask R-CNN]

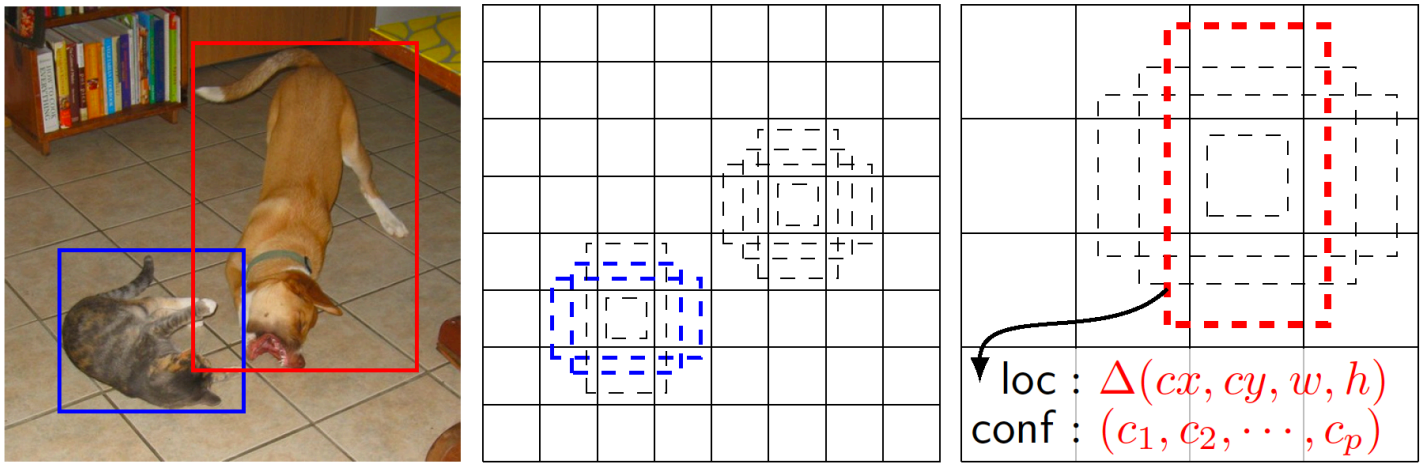
Single-Shot Detector



- Standard VGG (or ResNet) base network
- Add further layers with progressively decreasing size
  - Used for predicting detections at multiple scales
  - Layers with wider receptive fields expected to detect larger objects

[Liu et al., SSD]

SSD: Default boxes (anchors)



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

- SSD uses default boxes, which are similar to anchors in Faster R-CNN
- Default boxes located at each cell on the feature map
  - Multiple boxes corresponding to different scales and aspect ratios
- Anchor boxes of different feature maps have different scales
  - Various feature maps responsible for detecting objects of different sizes

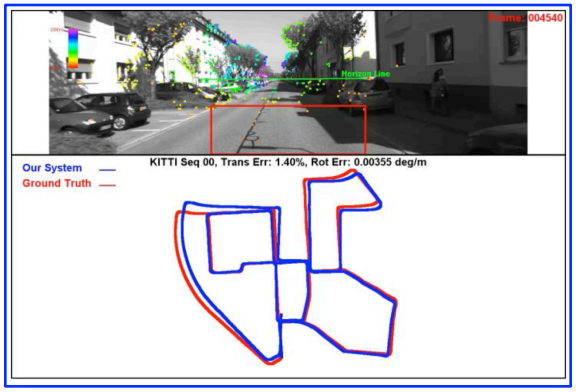
Computer Vision

A Great Time to Study Computer Vision!

Where is our car?

Structure from Motion

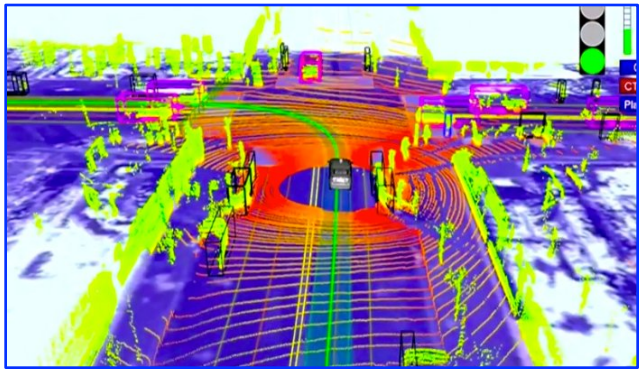
Visual SLAM



What is a safe path?

Behavior prediction

Path planning



Where are other agents?

Object detection

3D localization

Where are scene elements?

Semantic segmentation