CSE 252C: Advanced Computer Vision
Manmohan Chandraker

Lecture 18: Review
Course Evaluation

- Course evaluation:
  - You should have received an email

- Link: https://academicaffairs.ucsd.edu/Modules/Evals?e4900527

- Too long!
  - Short link: https://bit.ly/2ESQLwI

- Please take 5 minutes to do this!
Learning Correspondence
Measuring patch similarity

- **Hand Designed Features**
  - SIFT, SURF, ORB, FREAK, DAISY

- **Neural Networks**
  - Compare Patches [Zagoruyko. et al.]
  - MatchNet [Han et al.]
  - See by Moving [Agrawal et al.]
  - Stereo Matching Cost [Zbontar et al.]

- **Global Optimization**
  - Flow, DSP, etc.
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

\[ \| f(x) - f(x_+) \| \rightarrow 0 \]
\[ \| f(x) - f(x_-) \| \geq m \]
Loss function for Siamese CNNs

The final loss is defined as:

\[ L = \sum \text{loss of positive pairs} + \sum \text{loss of negative pairs} \]
Loss function for Siamese CNNs

- Combined into a contrastive loss

- 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 mini-batch of triplets

\[ \mathcal{T} = (x_a, x_p, x_n) \quad y_a = y_p \neq y_n \]

Anchor  Positive  Negative

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

\[ \|x_a - x_p\|^2 + m \leq \|x_a - x_n\|^2 \]

• Triplet loss:

\[ l_{tri}(\mathcal{T}) = \left[ \|x_a - x_p\|^2 - \|x_a - x_n\|^2 + m \right]_+ \]
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

Differentiable image sampling (bilinear)

$$V_{i}^{c} = \sum_{n}^{H} \sum_{m}^{W} U_{nm}^{c} \max(0, 1 - |x_{i}^{s} - m|) \max(0, 1 - |y_{i}^{s} - n|)$$

**target value**  
**source value**  
**sampling grid coordinate**

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

$$\frac{\partial V_{i}^{c}}{\partial U_{nm}^{c}} = \sum_{n}^{H} \sum_{m}^{W} \max(0, 1 - |x_{i}^{s} - m|) \max(0, 1 - |y_{i}^{s} - n|)$$

$$\frac{\partial V_{i}^{c}}{\partial x_{i}^{s}} = \sum_{n}^{H} \sum_{m}^{W} U_{nm}^{c} \max(0, 1 - |y_{i}^{s} - n|) \begin{cases} 0 & \text{if } |m - x_{i}^{s}| \geq 1 \\ 1 & \text{if } m \geq x_{i}^{s} \\ -1 & \text{if } m < x_{i}^{s} \end{cases}$$
Patch Normalization in UCN

SIFT normalizes for scale and rotation

Spatial transformer for global transformation

Convolutional spatial transformer for independent normalization at each pixel
UCN Architecture
Optical Flow
Optical flow: Coarse-to-Fine

Lucas-Kanade without pyramids
Fails in areas of large motion

Lucas-Kanade with Pyramids

* From Khurram Hassan-Shafique  CAP5415 Computer Vision 2003
FlowNet

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

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

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

- Hand-design architecture to encourage spatial matching
  - Meaningful image representation, then match at higher feature layer

- Correlation layer
  - Perform a multiplicative patch comparison
FlowNetCorr: Correlation Layer

- Multi-channel feature maps $f_1$ and $f_2$, of size $w \times h$ and $n$ channels
- Correlation of patches centered at $x_1$ and $x_2$:
  \[
  c(x_1, x_2) = \sum_{o \in [-k, k] \times [-k, k]} \langle f_1(x_1 + o), f_2(x_2 + o) \rangle
  \]
- Number of trainable parameters? None.
- Cost of computing correlations? $(2k+1)^2 \times (w \times h)^2 \times n$. 
FlowNetCorr: Correlation Layer

- Efficiency considerations:
  - Limit $x_2$ to lie within maximum disparity of $d$ pixels from $x_1$
  - Stride on $f_1$ to limit number of pixels where correlation is computed

- Stack each of $(2d+1)^2$ “disparity” maps as output channels
- Also stack $1 \times 1$ output of input feature to retain image information
FlowNetCorr: Refinement

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

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

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

Max Pooling
Remember which element was max!

1 2 6 3
3 5 2 1
1 2 2 1
7 3 4 8
Input: 4 x 4
Output: 2 x 2

Max Unpooling
Use positions from pooling layer

1 2
3 4
Input: 2 x 2
Output: 4 x 4

Rest of the network

Corresponding pairs of downsampling and upsampling layers

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

\[
\begin{bmatrix}
1 & 2 \\
3 & 4 \\
\end{bmatrix}
\star^T
\begin{bmatrix}
1 & 0 \\
2 & 1 \\
\end{bmatrix}
= 
\begin{bmatrix}
1 & 2 & 0 \\
5 & 9 & 2 \\
6 & 11 & 4 \\
\end{bmatrix}
\]

Multiply filter by each input value

Tile the scaled filter at output locations
Add overlapping values
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

- 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
Spatial Pyramid Network

- Basic goal: learn a CNN $G_k$ to predict residual flow at each level:
  \[ v_k = G_k(I^1_k, w(I^2_k, u(V_{k-1})), u(V_{k-1})) \]

- 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
Structure from Motion
Toolkit for Practical SFM

3D-3D 2D-2D Absolute Orientation

2D-3D Pose

2D-2D Relative Orientation

Triangulation

Bundle Adjustment

Robust Statistics
Fundamental Matrix

$x_1 \leftrightarrow x_2$

$x_1^T F x_2 = 0$

Degrees of freedom for $F$: 7

So, 7 points suffice to find $F$, but use 8 for linear method
Estimating F: Linear Method

• The fundamental matrix $F$ is defined by

$$x' \ F \ x = 0$$

for any pair of matches $x$ and $x'$ in two images.

• Let $x = (u,v,1)^T$ and $x' = (u',v',1)^T$, \[ F = \begin{pmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{pmatrix} \]

• Each match gives a linear equation:

$$uu'f_{11} + vu'f_{12} + u'f_{13} + uv'f_{21} + vv'f_{22} + v'f_{23} + uf_{31} + vf_{32} + f_{33} = 0$$
Direct Linear Transform

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

$$\begin{pmatrix} u_1 u'_1 & v_1 u'_1 & u'_1 & u_1 v'_1 & v_1 v'_1 & v'_1 & u_1 & v_1 & 1 \\ u_2 u'_2 & v_2 u'_2 & u'_2 & u_2 v'_2 & v_2 v'_2 & v'_2 & u_2 & v_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_n u'_n & v_n u'_n & u'_n & u_n v'_n & v_n v'_n & v'_n & u_n & v_n & 1 \end{pmatrix} \begin{pmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{pmatrix} = 0$$

- In reality, instead of solving $Af = 0$, we seek $f$ to minimize $\|Af\|$, using SVD.
Enforcing Rank Condition after DLT

1. \( \mathbf{F} \) should have rank 2
2. To enforce that \( \mathbf{F} \) is of rank 2, \( \mathbf{F} \) is replaced by the closest \( \mathbf{F}' \) that obeys the rank constraint.

3. This is achieved by SVD. Let \( \mathbf{F} = \mathbf{U} \Sigma \mathbf{V} \), where

\[
\Sigma = \begin{pmatrix}
  s_1 & 0 & 0 \\
  0 & s_2 & 0 \\
  0 & 0 & s_3 \\
\end{pmatrix}, \quad \Sigma' = \begin{pmatrix}
  s_1 & 0 & 0 \\
  0 & s_2 & 0 \\
  0 & 0 & 0 \\
\end{pmatrix}
\]

then \( \mathbf{F}' = \mathbf{U} \Sigma' \mathbf{V} \) is the solution.
RANSAC to Estimate Fundamental Matrix

• For $N$ times
  – Pick 8 points
  – Compute a solution for $F$ using these 8 points
  – Count number of inliers with $x_1^T F x_2$ 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 $F$:
  \[
  F = K_2^{-\top} E K_1^{-1} \\
  E = K_2^{-\top} F K_1
  \]
- Decompose $E$ into skew-symmetric and rotation matrices:
  \[
  E = [t] \times R
  \]
Four Possible Solutions

- Baseline reversal
- Rotate camera B by 180 degrees
- Baseline reversal

Diagram:
- 3D point
- 2D image point
- Image plane
- Principal axis
- Camera center

(a) Rotate camera B by 180 degrees
(b) Baseline reversal
(c) Baseline reversal
(d)
Triangulation

\[ Q_1 = O_1 + \lambda_1 q_1 = a + \lambda_1 b \]
\[ Q_2 = O_2 + \lambda_2 q_2 = c + \lambda_2 d \]

\[ [\lambda_1, \lambda_2] = \arg \min_{\lambda_1, \lambda_2} \text{dist}(Q_1, Q_2) \]
Bundle adjustment

- Minimize sum of squared reprojection errors:

\[ g(X, R, T) = \sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij} \cdot \| P(x_i, R_j, t_j) - [u_{i,j}] \|^2 \]

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

- Practical challenges?
  - Initialization
  - Outliers
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 $f$ and different $r$
Bundle Adjustment

- Refine a visual reconstruction to produce jointly optimal 3D structures $P$ and camera poses $C$.
- Minimize total reprojection errors $\Delta z$.

Cost Function:

$$\arg\min_X \sum_i \sum_j \left( x_{ij} - \pi(P_j, C_i) \right)^2_{W_{ij}}$$

$W_{ij}^{-1}$: Measurement error covariance

$X = [P, C]$
4. Levenberg-Marquardt

Regularized Gauss-Newton with damping factor $\lambda$.

$$(J^TWJ + \lambda I)\delta = -J^TW\Delta Z$$

$H_{LM}$

$\lambda \to 0$: Gauss-Newton (when convergence is rapid)

$\lambda \to \infty$: Gradient descent (when convergence is slow)
Bundle Adjustment: Problem Structure

- Primary structure: Camera and 3D point blocks
Bundle Adjustment: Primary Structure

\[ H_{LM} \delta = -J^T W \Delta Z \]

\[
\begin{bmatrix}
H_S & H_{SC} \\
H_{SC}^T & H_C
\end{bmatrix}
\begin{bmatrix}
\delta_S \\
\delta_C
\end{bmatrix} =
\begin{bmatrix}
\epsilon_S \\
\epsilon_C
\end{bmatrix}
\]

Multiply both sides by:

\[
\begin{bmatrix}
I & -H_{SC}H_C^{-1} \\
0 & I
\end{bmatrix}
\]

\[
\begin{bmatrix}
H_S - H_{SC}H_C^{-1}H_{SC}^T & 0 \\
H_{SC}^T & H_C
\end{bmatrix}
\begin{bmatrix}
\delta_S \\
\delta_C
\end{bmatrix} =
\begin{bmatrix}
\epsilon_S - H_{SC}H_C^{-1}\epsilon_C \\
\epsilon_C
\end{bmatrix}
\]

Solve for \( \delta_S \):

\[ \delta_S = (H_S - H_{SC}H_C^{-1}H_{SC}^T)^{-1}(\epsilon_S - H_{SC}H_C^{-1}\epsilon_C) \]

Solve for \( \delta_C \) by back-substitution:

\[ \delta_C = H_C^{-1}(\epsilon_C - H_{SC}^T\delta_S) \]

Schur complement: block diagonal matrix that is easy to invert

Small number of camera parameters: also inexpensive easy to invert
Real-Time SFM: Steady-State

- Usually absolute pose estimations rather than relative pose
Scale Drift Correction

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

Know height of camera, so scale $k$ is resolved

Spaces $kd_1$ and $kd_2$.

Resolution: compute height of ground plane.

$(R, kt)$
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

**Intra-personal variation**

**Inter-personal variation**
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 ($\lambda$ 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 $x^a_i$ and a positive $x^p_i$
  - Maximize distance between anchor $x^a_i$ and a negative $x^n_i$

\[ \| f(x^a_i) - f(x^p_i) \|_2^2 + \alpha < \| f(x^a_i) - f(x^n_i) \|_2^2, \ \forall (f(x^a_i), f(x^p_i), f(x^n_i)) \in T \]

- Total loss to minimize:
\[ \sum_{i=1}^{N} \left[ \| f(x^a_i) - f(x^p_i) \|_2^2 - \| f(x^a_i) - f(x^n_i) \|_2^2 + \alpha \right]_+ \]

- 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
  - One prediction per image, no candidates
  - Depends on quality of initial prediction

[Toshev and Szegedy, DeepPose]
Cascade of Heat Maps

- **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
  - Encodes probability of joint appearing at a pixel

- **Disadvantages:**
  - Limited ability to consider details
  - One prediction per image, no candidates
  - Depends on quality of initial prediction
  - Not enough modeling of spatial structure
  - Cannot reason about occluded parts, or those outside window

[Tompson et al., Efficient Object Localization]
Convolutional Pose Machines

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

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)
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 downsampling, 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

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

[Dilated Residual Network] [Yu et al., Dilated Residual Networks]
Object Detection
R-CNN

1. Input image
2. Extract region proposals (~2k)
3. Compute CNN features
4. Classify regions

- 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

- 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

Hi-res conv features: $C \times H \times W$ with region proposal

RoI conv features: Fixed size $C \times h \times w$ to classifier

[Girshick, 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
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) \(8 \times 8\) feature map | (c) \(4 \times 4\) 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

Where are other agents?
Object detection
3D localization

What is a safe path?
Behavior prediction
Path planning

Where are scene elements?
Semantic segmentation
Course Evaluation

- Course evaluation:
  - You should have received an email

- Link: [https://academicaffairs.ucsd.edu/Modules/Evals?e4900527](https://academicaffairs.ucsd.edu/Modules/Evals?e4900527)

- Too long!

- Please take 5 minutes to do this!