Announcements

- Today, I’ll summarize the most important points from the lecture.
  - It is an opportunity for you to ask questions…
  - …or get additional explanations about certain topics.
  - So, please do ask.
- Today’s slides are intended as an index for the lecture.
  - But they are not complete, won’t be sufficient as only tool.
  - Also look at the exercises – they often explain algorithms in detail.

Recap: Pinhole Camera

- (Simple) standard and abstract model today
  - Box with a small hole in it
  - Works in practice

Recap: Focus and Depth of Field

- Depth of field: distance between image planes where blur is tolerable

Recap: Field of View and Focal Length

- As \( f \) gets smaller, image becomes more wide angle
  - More world points project onto the finite image plane

- As \( f \) gets larger, image becomes more telescopic
  - Smaller part of the world projects onto the finite image plane
Recap: Color Sensing in Digital Cameras

Estimate missing components from neighboring values (demosaicing).

Recap: Effect of Filtering

- Noise introduces high frequencies. To remove them, we want to apply a "low-pass" filter.
- The ideal filter shape in the frequency domain would be a box. But this transfers to a spatial sinc, which has infinite spatial support.
- A compact spatial box filter transfers to a frequency sinc, which creates artifacts.
- A Gaussian has compact support in both domains. This makes it a convenient choice for a low-pass filter.

Recap: Gaussian Smoothing

- Gaussian kernel
  \[ G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2 + y^2)}{2\sigma^2}} \]
- Rotationally symmetric
- Weights nearby pixels more than distant ones
  - This makes sense as "probabilistic" inference about the signal
- A Gaussian gives a good model of a fuzzy blob.

Recap: Resampling with Prior Smoothing

- Note: We cannot recover the high frequencies, but we can avoid artifacts by smoothing before resampling.

Repetition

- Image Processing Basics
  - Image Formation
  - Linear Filters
  - Edge & Structure Extraction
- Segmentation & Grouping
- Object Recognition
- Local Features & Matching
- Deep Learning
- 3D Reconstruction
Recap: The Gaussian Pyramid

- $G_2 = (G_1 \ast \text{gaussian}) \downarrow 2$
- $G_1 = (G_0 \ast \text{gaussian}) \downarrow 2$
- $G_0 = \text{Image} \ast \text{blur}$

Recap: Derivatives and Edges...

- 1st derivative
- 2nd derivative
- "zero crossings" of second derivative

Recap: 2D Edge Detection Filters

- Laplacian of Gaussian
- Derivative of Gaussian
- $h_{uv}(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$
- $\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$

Recap: Canny Edge Detector

1. Filter image with derivative of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression:
   - Thin multi-pixel wide "ridges" down to single pixel width
4. Linking and thresholding (hysteresis):
   - Define two thresholds: low and high
   - Use the high threshold to start edge curves and the low threshold to continue them
5. MATLAB:
   - >> edge(image, 'canny')
   - >> help edge

Recap: Edges vs. Boundaries

- Edges useful signal to indicate occluding boundaries, shape.
- Here the raw edge output is not so bad...
- ...but quite often boundaries of interest are fragmented, and we have extra "clutter" edge points.
Recap: Fitting and Hough Transform

Given a model of interest, we can overcome some of the missing and noisy edges using fitting techniques. With voting methods like the Hough transform, detected points vote on possible model parameters.

Recap: Hough Transform

- How can we use this to find the most likely parameters $(m, b)$ for the most prominent line in the image space?
- Let each edge point in image space vote for a set of possible parameters in Hough space
- Accumulate votes in discrete set of bins; parameters with the most votes indicate line in image space.

Recap: Hough Transf. Polar Parametrization

- Usual $(m,b)$ parameter space problematic: can take on infinite values, undefined for vertical lines.
- Point in image space $\Rightarrow$ sinusoid segment in Hough space

Recap: Hough Transform for Circles

- Circle: center $(a, b)$ and radius $r$
- For an unknown radius $r$, unknown gradient direction

Recap: Generalized Hough Transform

- What if want to detect arbitrary shapes defined by boundary points and a reference point?
- At each boundary point, compute displacement vector: $r = a - p_i$
- For a given model shape, store these vectors in a table indexed by gradient orientation $\theta$.

Repetition

- Image Processing Basics
- Segmentation & Grouping
  - Segmentation and Grouping
  - Segmentation as Energy Minimization
- Object Recognition
- Local Features & Matching
- Deep Learning
- 3D Reconstruction
Recap: Gestalt Theory

- Gestalt: whole or group
  - Whole is greater than sum of its parts
  - Relationships among parts can yield new properties/features
- Psychologists identified series of factors that predispose set of elements to be grouped (by human visual system)

"I stand at the window and see a house, trees, sky. Theoretically I might say there were 327 brightnesses and nuances of colour. Do I have "327"? No, I have sky, house, and trees."

Max Wertheimer (1880-1943)

Untersuchungen zur Lehre von der Gestalt, Psychologische Forschung, Vol. 4, pp. 301-350, 1923
http://psy.ed.asu.edu/~classics/Wertheimer/Forms/forms.htm

Recap: Gestalt Factors

- These factors make intuitive sense, but are very difficult to translate into algorithms.

Recap: Image Segmentation

- Goal: identify groups of pixels that go together

Recap: K-Means Clustering

- Basic idea: randomly initialize the k cluster centers, and iterate between the two following steps
  1. Randomly initialize the cluster centers, $c_1, \ldots, c_k$
  2. Given cluster centers, determine points in each cluster
     - For each point $p$, find the closest $c_i$. Put $p$ into cluster $i$
  3. Given points in each cluster, solve for $c_i$
     - Set $c_i$ to be the mean of points in cluster $i$
  4. If $c_i$ have changed, repeat Step 2
- Properties
  - Will always converge to some solution
  - Can be a “local minimum”
  - Does not always find the global minimum of objective function:
    $$\sum_{i} \sum_{\text{points } p \text{ in cluster } i} ||p - c_i||^2$$

Recap: Expectation Maximization (EM)

- Goal
  - Find blob parameters $\theta$ that maximize the likelihood function:
    $$p(\text{data}|\theta) = \prod_{n=1}^{N} p(x_n|\theta)$$
- Approach:
  1. E-step: given current guess of blobs, compute ownership of each point
  2. M-step: given ownership probabilities, update blobs to maximize likelihood function
  3. Repeat until convergence

Recap: Mean-Shift Algorithm

- Iterative Mode Search
  1. Initialize random seed, and window $W$
  2. Calculate center of gravity (the “mean”) of $W$
  3. Shift the search window to the mean
  4. Repeat Step 2 until convergence
Recap: Mean-Shift Clustering

- Cluster: all data points in the attraction basin of a mode
- Attraction basin: the region for which all trajectories lead to the same mode

Recap: Mean-Shift Segmentation

- Find features (color, gradients, texture, etc)
- Initialize windows at individual pixel locations
- Perform mean shift for each window until convergence
- Merge windows that end up near the same “peak” or mode

Repetition

- Image Processing Basics
- Segmentation & Grouping
  - Segmentation and Grouping
  - Segmentation as Energy Minimization
- Object Recognition
- Local Features & Matching
- Deep Learning
- 3D Reconstruction

Recap: MRFs for Image Segmentation

- MRF formulation
- Unary potentials
  - E.g. color model, modeled with a Mixture of Gaussians
  - \( \phi(x_i; y_i) = \log \sum_k \theta_k p(k|x_i) N(y_i; \hat{y}_k, \Sigma_k) \)
  - How likely is a pixel/patch to belong to a certain class (e.g. foreground/background)?
- Pairwise potentials
  - How different is a pixel/patch’s label from that of its neighbor? (e.g. based on intensity/color/texture difference, edges)

Recap: How to Set the Potentials?

- Unary potentials
- E.g. color model, modeled with a Mixture of Gaussians
- \( \phi(x_i; y_i; \theta_k) = \log \sum_k \theta_k p(k|x_i) N(y_i; \hat{y}_k, \Sigma_k) \)
  - Learn color distributions for each label
- Pairwise potentials
- How different is a pixel/patch’s label from that of its neighbor? (e.g. based on intensity/color/texture difference, edges)
Recap: How to Set the Potentials?

- Pairwise potentials
  - Potts Model
    \[ \psi(x_i, x_j; \theta_\psi) = \theta_\psi \delta(x_i \neq x_j) \]
    - Simplest discontinuity preserving model.
    - Discontinuities between any pair of labels are penalized equally.
    - Useful when labels are unordered or number of labels is small.
  - Extension: "Contrast sensitive Potts model"
    \[ \psi(x_i, x_j, g_{ij}(y); \theta_\psi) = \theta_\psi g_{ij}(y) \delta(x_i \neq x_j) \]
    where
    \[ g_{ij}(y) = e^{\beta |y_i - y_j|} \]
    ⇒ Discourages label changes except in places where there is also a large change in the observations.

Recap: Graph-Cuts Energy Minimization

- Solve an equivalent graph cut problem
  1. Introduce extra nodes: source and sink
  2. Weight connections to source/sink (t-links)
     \[ \phi(x_i = s) \] and \[ \phi(x_i = t) \], respectively.
  3. Weight connections between nodes (n-links)
     \[ \psi(x_i, x_j) \]
  4. Find the minimum cost cut that separates source from sink.
    ⇒ Solution is equivalent to minimum of the energy.

Recap: When Can s-t Graph Cuts Be Applied?

\[ E(L) = \sum_{\text{t-links}} E_p(L_p) + \sum_{\text{n-links}} E(L_p, L_q) \]

- s-t graph cuts can only globally minimize binary energies that are submodular. [Boros & Hammer, 2002, Kolmogorov & Zabih, 2004]

\[ E(L) \text{ can be minimized by s-t graph cuts } \iff E(s,s) + E(t,t) \leq E(s,t) + E(t,s) \]

Submodularity ("convexity")

- Submodularity is the discrete equivalent to convexity.
  - Implies that every local energy minimum is a global minimum.
  - Solution will be globally optimal.

First Applications Take Up Shape…

- Skin color detection
- Simple shape recognition
- Circle detection
- Line detection
- Car/non-car detection

Recap: Sliding-Window Object Detection

- If object may be in a cluttered scene, slide a window around looking for it.

- Essentially, this is a brute-force approach with many local decisions.

Repetition

- Image Processing Basics
- Segmentation & Grouping
- Object Recognition
  - Sliding Window based Object Detection
- Local Features & Matching
- Deep Learning
- 3D Reconstruction

[Image Source: http://www.flickr.com/photos/angelsk/2806412807/]

[Image Source: http://www.flickr.com/photos/angelsk/2806412807/]

[Image Source: http://www.flickr.com/photos/angelsk/2806412807/]

[Image Source: http://www.flickr.com/photos/angelsk/2806412807/]

[Image Source: http://www.flickr.com/photos/angelsk/2806412807/]

[Image Source: http://www.flickr.com/photos/angelsk/2806412807/]

[Image Source: http://www.flickr.com/photos/angelsk/2806412807/]
Recap: Gradient-based Representations

- Consider edges, contours, and (oriented) intensity gradients
- Summarize local distribution of gradients with histogram
  - Locally orderless: offers invariance to small shifts and rotations
  - Contrast-normalization: try to correct for variable illumination

Classifier Construction: Many Choices...

- Nearest Neighbor
  - Berg, Berg, Malik 2005, Chum, Zisserman 2007, Boiman, Shechtman, Irani 2008, ...
- Neural networks
  - LeCun, Bottou, Bengio, Haffner 1998, Rowley, Bala, Kanade 1998, ...
- Boosting
  - Viola, Jones 2001, Torralba et al. 2004, Opelt et al. 2006, Benenson 2012, ...
- Support Vector Machines
- Randomized Forests

Recap: Support Vector Machines (SVMs)

- Discriminative classifier based on optimal separating hyperplane (i.e. line for 2D case)
- Maximize the margin between the positive and negative training examples

Recap: Non-Linear SVMs

- General idea: The original input space can be mapped to some higher-dimensional feature space where the training set is separable:
  \[ \Phi: x \rightarrow \varphi(x) \]

Recap: HOG Descriptor Processing Chain

- SVM Classification
  - Typically using a linear SVM
  - Object/Non-object
    - Linear SVM
    - Collect HOVs over detection window
    - Contrast normalize over overlapping spatial cells
    - Weighted vote in spatial & orientation cells
    - Compute gradients
    - Gamma compression
  - Image Window

Recap: Non-Maximum Suppression
Recap: Viola-Jones Face Detection

"Rectangular" filters

Feature output is difference between adjacent regions

Efficiently computable with integral image: any sum can be computed in constant time.

Avoid scaling images for scale features directly for same cost.

Recap: AdaBoost Feature+Classifier Selection

Want to select the single rectangle feature and threshold that best separates positive (faces) and negative (non-faces) training examples, in terms of weighted error.

Resulting weak classifier:

For next round, reweight the examples according to errors, choose another filter/threshold combo.

Recap: AdaBoost

Final classifier is combination of the weak classifiers.

LeCun, Bottou, Bengio, Haffner 1998
Rowley, Baluja, Kanade 1998

B. Leibe

Recap: Viola-Jones Face Detector

Train cascade of classifiers with AdaBoost

New image

Selected features, thresholds, and weights.

• Train with 5K positives, 350M negatives
• Real-time detector using 38 layer cascade
• 6061 features in final layer
• [Implementation available in OpenCV: http://sourceforge.net/projects/opencvlibrary/]

Repetition

• Image Processing Basics
• Segmentation & Grouping
• Object Recognition
• Local Features & Matching
  • Local Features – Detection and Description
  • Recognition with Local Features
• Deep Learning
• 3D Reconstruction

B. Leibe
Recap: Local Feature Matching Pipeline

1. Find a set of distinctive keypoints
2. Define a region around each keypoint
3. Extract and normalize the region content
4. Compute a local descriptor from the normalized region
5. Match local descriptors

Recap: Requirements for Local Features

- Problem 1: Detect the same point independently in both images
- Problem 2: For each point correctly recognize the corresponding one

We need a repeatable detector!

We need a reliable and distinctive descriptor!

Recap: Harris Detector [Harris88]

- Compute second moment matrix (autocorrelation matrix)

\[ M(\sigma_x, \sigma_y) = g(\sigma_x) I_x I_x, g(\sigma_y) I_y I_y, g(\sigma_x) I_x I_y, g(\sigma_y) I_y I_x \]

1. Image derivatives
2. Square of derivatives
3. Gaussian filter \( g(\sigma) \)
4. Cornerness function – two strong eigenvalues

\[
R = \det(M(\sigma_x, \sigma_y)) - \alpha \text{tr}(M(\sigma_x, \sigma_y))
= g(f_x')g(f_y') - \alpha(g(f_x')^2 + g(f_y')^2)
\]
5. Perform non-maximum suppression

Recap: Harris Detector Responses [Harris88]

Effect: A very precise corner detector.

Recap: Hessian Detector [Beaudet78]

- Hessian determinant

\[
\text{Hessian}(I) = \begin{bmatrix}
I_{xx} & I_{xy} \\
I_{xy} & I_{yy}
\end{bmatrix}
\]

\[
\det(\text{Hessian}(I)) = I_{xx}I_{yy} - I_{xy}^2
\]

In Matlab:

\[
I_{xx} * I_{yy} - (I_{xy})^2 \times 2
\]

Recap: Hessian Detector Responses [Beaudet78]

Effect: Responses mainly on corners and strongly textured areas.
Recap: Automatic Scale Selection

- Function responses for increasing scale (scale signature)

Recap: Laplacian-of-Gaussian (LoG)

- Interest points:
  - Local maxima in scale space of Laplacian-of-Gaussian

Recap: LoG Detector Responses

Recap: Key point localization with DoG

- Efficient implementation
  - Approximate LoG with a difference of Gaussians (DoG)

- Approach DoG Detector
  - Detect maxima of difference-of-Gaussian in scale space
  - Reject points with low contrast (threshold)
  - Eliminate edge responses

Recap: Harris-Laplace

1. Initialization: Multiscale Harris corner detection
2. Scale selection based on Laplacian (same procedure with Hessian ⇒ Hessian-Laplace)

Recap: Orientation Normalization

- Compute orientation histogram
  - Select dominant orientation
  - Normalize: rotate to fixed orientation
Local Features & Matching

Image content is transformed into local features that are

- Scale Invariant Feature Transform
- Descriptor computation:
  - Divide patch into $4 \times 4$ sub-patches: 16 cells
  - Compute histogram of gradient orientations (8 reference angles) for all pixels inside each sub-patch
  - Resulting descriptor: $4 \times 4 \times 8 = 128$ dimensions

Object Recognition

- Descriptor computation:
  - Mikolajczyk

Recognition with Local Features

Mikolajczyk

Segmentation & Grouping

Resulting descriptor: $4 \times 4 \times 8 = 128$ dimensions

Assuming we know the correspondences, how do we get the transformation?

\[ \mathbf{B} \mathbf{A} = \mathbf{B} \mathbf{O} \]

Recognition pipeline

- Image Processing Basics
- Segmentation & Grouping
- Object Recognition
- Local Features & Matching
  - Local Features – Detection and Description
  - Recognition with Local Features
- Deep Learning
- 3D Reconstruction

Recap: Recognition with Local Features

- Image content is transformed into local features that are invariant to translation, rotation, and scale
- Goal: Verify if they belong to a consistent configuration

Recap: Fitting an Affine Transformation

- Assuming we know the correspondences, how do we get the transformation?

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix}
= 
\begin{bmatrix}
  m_1 & m_2 & m_3 \\
  m_4 & m_5 & m_6
\end{bmatrix}
\begin{bmatrix}
  x \\
  y
\end{bmatrix} 
+ 
\begin{bmatrix}
  t_1 \\
  t_2
\end{bmatrix}
\]

Recognition pipeline

Fitting affine transformations & homographies

Gen. Hough Transform

Recap: Fitting a Homography

- Estimating the transformation

\[ \mathbf{B} \mathbf{A} = \mathbf{B} \mathbf{O} \]

\[ \mathbf{A} \mathbf{h} = 0 \]

Repetition

Homogeneous coordinates

Image coordinates

Matrix rotation

\[ x' = Hx \]

\[ \mathbf{A} \mathbf{h} = \begin{bmatrix}
  a_{11} & a_{12} & a_{13} & a_{14} \\
  a_{21} & a_{22} & a_{23} & a_{24} \\
  a_{31} & a_{32} & a_{33} & a_{34} \\
  a_{41} & a_{42} & a_{43} & a_{44}
\end{bmatrix}
\]

\[ \mathbf{h} = \begin{bmatrix}
  h_1 \\
  h_2 \\
  h_3 \\
  h_4
\end{bmatrix} \]

\[ \mathbf{Ah} = 0 \]

\[ a_{11}x + a_{12}y + a_{13} = 0 \]

\[ a_{21}x + a_{22}y + a_{23} = 0 \]

\[ a_{31}x + a_{32}y + a_{33} = 0 \]

\[ a_{41}x + a_{42}y + a_{43} + a_{44} = 0 \]
Recap: Fitting a Homography

- Estimating the transformation
- Solution:
  - Null-space vector of $A$
  - Corresponds to smallest eigenvector

$$A_h = 0$$

SVD

$$A = UDV^T$$

Minimizes least square error

Recap: RANSAC

RANSAC loop:
1. Randomly select a seed group of points on which to base transformation estimate (e.g., a group of matches)
2. Compute transformation from seed group
3. Find inliers to this transformation
4. If the number of inliers is sufficiently large, re-compute least-squares estimate of transformation on all of the inliers
   - Keep the transformation with the largest number of inliers

Recap: RANSAC Line Fitting Example

- Task: Estimate the best line

Sample two points

Fit a line to them

Total number of points within a threshold of line.
Recap: RANSAC Line Fitting Example

- Task: Estimate the best line

Repeat, until we get a good result.

Recap: Feature Matching Example

- Find best stereo match within a square search window (here 300 pixels

Global transformation model: epipolar geometry

Recap: Generalized Hough Transform

- Suppose our features are scale- and rotation-invariant
  - Then a single feature match provides an alignment hypothesis (translation, scale, orientation).
  - Of course, a hypothesis from a single match is unreliable.
  - Solution: let each match vote for its hypothesis in a Hough space with very coarse bins.

Recap: Convolutional Neural Networks

- Neural network with specialized connectivity structure
  - Stack multiple stages of feature extractors
  - Higher stages compute more global, more invariant features
  - Classification layer at the end

Application: Panorama Stitching

http://www.cs.ubc.ca/~mbrown/autostitch/autostitch.html

Repetition

- Image Processing Basics
- Segmentation & Grouping
- Object Recognition
- Local Features & Matching
- Deep Learning
  - Convolutional Neural Networks (CNNs)
  - Deep Learning Background
  - CNNs for Object Detection
  - CNNs for Semantic Segmentation
  - CNNs for Matching & RNNs
- 3D Reconstruction

"LeNet" architecture

Recap: CNN Structure

- Feed-forward feature extraction
  1. Convolve input with learned filters
  2. Non-linearity
  3. Spatial pooling
  4. (Normalization)
- Supervised training of convolutional filters by back-propagating classification error

Recap: Intuition of CNNs

- Convolutional network
  > Share the same parameters across different locations
  > Convolutions with learned kernels
- Learn multiple filters
  > E.g. 1000 × 1000 image
  > 100 filters
  > 10 × 10 filter size
  > => only 10k parameters
- Result: Response map
  > size: 1000 × 1000 × 100
  > Only memory, not params!

Recap: Convolution Layers

- All Neural Net activations arranged in 3 dimensions
  - Multiple neurons all looking at the same input region, stacked in depth
  - Form a single [1 × 1 × depth] depth column in output volume.

Recap: Activation Maps

- Each activation map is a depth slice through the output volume.

Recap: Pooling Layers

- Effect:
  > Make the representation smaller without losing too much information
  > Achieve robustness to translations

Recap: Effect of Multiple Convolution Layers

- Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]
Recap: AlexNet (2012)

- Similar framework as LeNet, but
  - Bigger model (7 hidden layers, 650k units, 60M parameters)
  - More data ($10^8$ images instead of $10^3$)
  - GPU implementation
  - Better regularization and up-to-date tricks for training (Dropout)


Recap: GoogLeNet (2014)

- Main ideas
  - Deeper network
  - Stack convolutional layers with smaller filters (+ nonlinearity)

- Results
  - Improved ILSVRC top-5 error rate to 6.7%.

Recap: Residual Networks

- Core component
  - Skip connections bypassing each layer
  - Better propagation of gradients to the deeper layers
  - This makes it possible to train (much) deeper networks.

Recap: Transfer Learning with CNNs

1. Train on ImageNet
2. If small dataset: fix all weights (treat CNN as fixed feature extractor), retrain only the classifier
   - I.e., replace the Softmax layer at the end
3. If you have a medium sized dataset, “finetune” instead: use the old weights as initialization, train the full network or only some of the higher layers.

Repetition

- Image Processing Basics
- Segmentation & Grouping
- Object Recognition
- Local Features & Matching
- Deep Learning
  - Convolutional Neural Networks (CNNs)
  - Deep Learning Background
  - CNNs for Object Detection
  - CNNs for Semantic Segmentation
  - CNNs for Matching & RNNs
- 3D Reconstruction
Recap: Multi-Layer Perceptrons

- Deep network = Also learning the feature transformation
  - Output
    \[ y_k(x) = g^{(2)} \left( \sum_{i=0}^{n} W_{ki}^{(2)} g^{(1)} \left( \sum_{j=0}^{d} W_{ij}^{(1)} x_j \right) \right) \]

- General formulation (used in deep learning packages)
  - By very careful implementation (e.g., adjusting the weights in the direction of the gradient)
  - By choosing suitable nonlinearities (e.g., ReLU)

- In multilayer nets, gradients need to be propagated backward pass that spits out a number.

- The learning rate?

- Vanishing gradients problem
  - By restricting the network depth (shallow networks are easier)
  - By very careful implementation (numerics matter)

Recap: Supervised Learning

- Two main steps
  1. Computing the gradients for each weight (backprop)
  2. Adjusting the weights in the direction of the gradient

- Gradient Descent: Basic update equation
  \[ w_{kj}^{(t+1)} = w_{kj}^{(t)} - \eta \frac{\partial E(w)}{\partial w_{kj}} \]

- Important considerations
  - On what data do we want to apply this? (Minibatches)
  - How should we choose the step size \( \eta \) (the learning rate)?
  - More advanced optimizers (Momentum, RMSProp, Adam, …)

Recap: Glorot Initialization

- Variance of neuron activations
  - Suppose we have an input \( X \) with \( n \) components and a linear neuron with random weights \( W \) that splits out a number \( Y \).
  - We want the variance of the input and output of a unit to be the same, therefore \( \text{Var}(W_i) \) should be 1. This means
    \[ \text{Var}(W_i) = \frac{1}{n_{in}} \]
    - Or for the backpropagated gradient
      \[ \text{Var}(W_i) = \frac{1}{n_{out}} \]
    - As a compromise, Glorot & Bengio propose to use
      \[ \text{Var}(W) = \frac{2}{n_{in} + n_{out}} \]
    - Randomly sample the initial weights with this variance.

Recap: He Initialization

- Extension of Glorot Initialization to ReLU units
  - Use Rectified Linear Units (ReLU)
    \[ g(a) = \max \{ 0, a \} \]
    - Effect: gradient is propagated with a constant factor
      \[ \frac{\partial g(a)}{\partial a} = \begin{cases} 1, & a > 0 \\ 0, & \text{else} \end{cases} \]
  - Same basic idea: Output should have the input variance
    - However, the Glorot derivation was based on tanh units, linearity assumption around zero does not hold for ReLU.
    - He et al. made the derivations, proposed to use instead
      \[ \text{Var}(W) = \frac{2}{n_{in}} \]
Recap: Batch Normalization
[Ioffe & Szegedy ’14]

- **Motivation**: Optimization works best if all inputs of a layer are normalized.
- **Idea**
  - Introduce intermediate layer that centers the activations of the previous layer per minibatch.
  - I.e., perform transformations on all activations and undo those transformations when backpropagating gradients
- **Complication**: centering + normalization also needs to be done at test time, but minibatches are no longer available at that point.
  - Learn the normalization parameters to compensate for the expected bias of the previous layer (usually a simple moving average)
- **Effect**
  - Much improved convergence (but parameter values are important!)
  - Widely used in practice

Recap: Dropout
[Srivastava, Hinton ’12]

- **Idea**
  - Randomly switch off units during training.
  - Change network architecture for each data point, effectively training many different variants of the network.
  - When applying the trained network, multiply activations with the probability that the unit was set to zero.
- **Effect**
  - Improved performance

Recap: Reducing the Learning Rate

- Final improvement step after convergence is reached
  - Reduce learning rate by a factor of 10.
  - Continue training for a few epochs.
  - Do this 1-3 times, then stop training.
- **Effect**
  - Turning down the learning rate will reduce the random fluctuations in the error due to different gradients on different minibatches.
- **Be careful! Do not turn down the learning rate too soon!**
  - Further progress will be much slower after that.

Recap: Data Augmentation

- **Effect**
  - Much larger training set
  - Robustness against expected variations
- **During testing**
  - When cropping was used during training, need to again apply crops to get same image size.
  - Beneficial to also apply flipping during test.
  - Applying several ColorPCA variations can bring another ~1% improvement, but at a significantly increased runtime.

Recap: R-CNN for Object Detection

- **Key ideas**
  - Extract region proposals (Selective Search)
  - Use a pre-trained/fine-tuned classification network as feature extractor (initially AlexNet, later VGGNet) on those regions

Recap: R-CNN for Object Detection

- Classify regions with SVMs
- Forward each region through ConvNet
- Regions of interest (RoI) from a proposal method (e.g., Fast R-CNN)
- Input image

Recap: Faster R-CNN

- One network, four losses
  - Remove dependence on external region proposal algorithm.
- Instead, infer region proposals from same CNN.
- Feature sharing
- Joint training
  - Object detection in a single pass becomes possible.

Recap: Mask R-CNN


Recap: YOLO / SSD

- Idea: Directly go from image to detection scores
- Within each grid cell
  - Start from a set of anchor boxes
  - Regress from each of the B anchor boxes to a final box
  - Predict scores for each of C classes (including background)

Recap: Fully Convolutional Networks

- CNN
  - FCN
- Intuition
  - Think of FCNs as performing a sliding-window classification, producing a heatmap of output scores for each class
Recap: Fully-Convolutional Networks

- Design a network as a sequence of convolutional layers
  - Fully Convolutional Networks (FCNs)
    - All operations formulated as convolutions
    - Fully-connected layers become 1x1 convolutions
  - Advantage: can process arbitrarily sized images

Recap: Encoder-Decoder Architecture

- Design a network as a sequence of convolutional layers
  - With downsampling and upsampling inside the network!
  - Downsampling
    - Pooling, strided convolution
  - Upsampling
    - Unpooling or strided transpose convolution

Recap: Skip Connections

- Encoder-Decoder Architecture with skip connections
  - Use skip connections to preserve this higher-resolution information

Recap: FCNs for Human Pose Estimation

- Formulate pose estimation as a segmentation problem
  - Define a target disk around each keypoint with radius r
  - Infer heatmaps for the joints as in semantic segmentation

Recap: Types of Models used for Matching Tasks

- Identification models (I)
  - Triplet Loss
  - Large-margin loss, Triplet loss
- Embedding models (E)
  - Same versus different
  - Multi-class classification loss
- Verification models (V)
  - Two-class classification loss
Triplet Loss Networks

- Learning a discriminative embedding
  - Present the network with triplets of examples
  - Apply triplet loss to learn an embedding $f()$ that groups the positive example closer to the anchor than the negative one.
  \[ \| f(x^p) - f(x^a) \|^2 < \| f(x^a) - f(x^n) \|^2 \]
  - Used with great success in Google's FaceNet face identification

Offline Hard Triplet Mining

- Considerable effort needed

Better: Online Hard Triplet Mining

- Core idea
  - The minibatch contains many more potential triplets than the ones that were mined!
  - Why not make use of those also?
- Possible improvement
  - Each member of another triplet becomes an additional negative candidate
  - But: need both hard negatives and hard positives!
- Better design
  - Sample K images from P classes (=people) for each minibatch
  - Triplets are only constructed within the minibatch

Recap: Recurrent Neural Networks

- RNNs are regular NNs whose hidden units have additional forward connections over time.
  - You can unroll them to create a network that extends over time.
  - When you do this, keep in mind that the weights for the hidden units are shared between temporal layers.
- RNNs are very powerful, because they combine two properties:
  - Distributed hidden state that allows them to store a lot of information about the past efficiently.
  - Non-linear dynamics that allows them to update their hidden state in complicated ways.
- With enough neurons and time, RNNs can compute anything that can be computed by your computer.
- Training is more challenging (unrolled networks are deep)
  - See Machine Learning lecture for details…
Recap: Applications – Image Tagging

- Simple combination of CNN and RNN
  - Use CNN to define initial state $h_0$ of an RNN.
  - Use RNN to produce text description of the image.

Repetition

- Image Processing Basics
- Segmentation & Grouping
- Object Recognition
- Local Features & Matching
- Deep Learning
- 3D Reconstruction
  - Epipolar Geometry and Stereo Basics
  - Camera Calibration & Uncalibrated Reconstruction
  - Structure-from-Motion

Recap: What Is Stereo Vision?

- Generic problem formulation: given several images of the same object or scene, compute a representation of its 3D shape

Recap: Depth with Stereo – Basic Idea

- Basic Principle: Triangulation
  - Gives reconstruction as intersection of two rays
  - Requires
    - Camera pose (calibration)
    - Point correspondence

Recap: Epipolar Geometry

- Geometry of two views allows us to constrain where the corresponding pixel for some image point in the first view must occur in the second view.

- Epipolar constraint:
  - Correspondence for point $p$ in $\Pi$ must lie on the epipolar line $l^\prime$ in $\Pi^\prime$ (and vice versa)
  - Reduces correspondence problem to 1D search along conjugate epipolar lines.

Recap: Stereo Geometry With Calibrated Cameras

- Camera-centered coordinate systems are related by known rotation $R$ and translation $T$:
  $$X' = RX + T$$
This holds for the rays \( p \) and \( p' \) that are parallel to the camera-centered position vectors \( X \) and \( X' \), so we have:

\[
\begin{bmatrix}
\mathbf{p}
\end{bmatrix}^T \mathbf{E} \begin{bmatrix}
\mathbf{p}'
\end{bmatrix} = 0
\]

- \( E \) is called the essential matrix, which relates corresponding image points [Longuet-Higgins 1981]

Recap: Essential Matrix

In practice, it is convenient if image scanlines are the epipolar lines.

- Algorithm
  - Reproject image planes onto a common plane parallel to the line between optical centers
  - Pixel motion is horizontal after this transformation
  - Two homographies (3x3 transforms), one for each input image reprojection

Recap: Stereo Image Rectification

Recap: Dense Correspondence Search

• For each pixel in the first image
  - Find corresponding epipolar line in the right image
  - Examine all pixels on the epipolar line and pick the best match (e.g. SSD, correlation)
  - Triangulate the matches to get depth information

• This is easiest when epipolar lines are scanlines

→ Rectify images first

Recap: Essential Matrix and Epipolar Lines

Epipolar constraint: if we observe point \( p \) in one image, then its position \( p' \) in second image must satisfy this equation.

\[
\begin{bmatrix}
\mathbf{l}
\end{bmatrix} = \begin{bmatrix}
\mathbf{E}^T \mathbf{p}'
\end{bmatrix}
\]

is the coordinate vector representing the epipolar line for point \( p \)

(i.e., the line is given by: \( \begin{bmatrix}
\mathbf{E}
\end{bmatrix} \begin{bmatrix}
\mathbf{x}
\end{bmatrix} = 0 \))

Recap: Dense Correspondence Search

Recap: Effect of Window Size

Want window large enough to have sufficient intensity variation, yet small enough to contain only pixels with about the same disparity.

Repetition

• Image Processing Basics
• Segmentation & Grouping
• Object Recognition
• Local Features & Matching
• Deep Learning
• 3D Reconstruction
  - Epipolar Geometry and Stereo Basics
  - Camera Calibration & Uncalibrated Reconstruction
  - Structure-from-Motion
Recap: A General Point

- Equations of the form
  \[ \mathbf{Ax} = 0 \]

- How do we solve them? (always!)
  - Apply SVD
    \[ \mathbf{A} = \mathbf{UDV}^T = \mathbf{U} \begin{bmatrix} d_{11} & \cdots & d_{2n} \cr \vdots & \ddots & \vdots \cr d_{n1} & \cdots & d_{nn} \end{bmatrix} \begin{bmatrix} u_1 & \cdots & u_n \end{bmatrix}^T \]
    - Singular values \( \lambda \)
    - Singular vectors \( \mathbf{U} \)

  - Singular values of \( \mathbf{A} \) = square roots of the eigenvalues of \( \mathbf{A}^T \mathbf{A} \).
  - The solution of \( \mathbf{Ax} = 0 \) is the null space vector of \( \mathbf{A} \).
  - This corresponds to the smallest singular vector of \( \mathbf{A} \).

Recap: Camera Parameters

- Intrinsic parameters
  - Principal point coordinates \( \mathbf{m} \)
  - Focal length \( \mathbf{f} \)
  - Pixel magnification factors \( \mathbf{s} \)
  - Skew (non-rectangular pixels) \( \mathbf{a} \)
  - Radial distortion

- Extrinsic parameters
  - Rotation \( \mathbf{R} \)
  - Translation \( \mathbf{t} \)
  - (both relative to world coordinate system)

- Camera projection matrix
  \[ \mathbf{P} = \mathbf{K} [\mathbf{R} | \mathbf{t}] \]
  \[ \Rightarrow \text{General pinhole camera:} \quad 9 \text{ DoF} \]
  \[ \Rightarrow \text{CCD Camera with square pixels:} \quad 10 \text{ DoF} \]
  \[ \Rightarrow \text{General camera:} \quad 11 \text{ DoF} \]

Recap: Calibrating a Camera

Goal
- Compute intrinsic and extrinsic parameters using observed camera data.

Main idea
- Place "calibration object" with known geometry in the scene
- Get correspondences
- Solve for mapping from scene to image: estimate \( \mathbf{P} = \mathbf{P}_n \mathbf{P}_e \)

Recap: Camera Calibration (DLT Algorithm)

\[ \begin{bmatrix} 0^T & \mathbf{X}_1^T & -y_1 \mathbf{X}_1^T \\ \vdots & \vdots & \vdots \\ 0^T & \mathbf{X}_n^T & -y_n \mathbf{X}_n^T \end{bmatrix} \begin{bmatrix} \mathbf{P}_1 \\ \vdots \\ \mathbf{P}_n \end{bmatrix} = 0 \quad \mathbf{Ap} = 0 \]

- \( \mathbf{P} \) has 11 degrees of freedom.
- Two linearly independent equations per independent 2D/3D correspondence.
- Solve with SVD (similar to homography estimation)
  - Solution corresponds to smallest singular vector.
- 5 ½ correspondences needed for a minimal solution.


- Two independent equations each in terms of three unknown entries of \( \mathbf{X} \).
- Stack equations and solve with SVD.
- This approach nicely generalizes to multiple cameras.

Recap: Epipolar Geometry – Calibrated Case

Camera matrix: \((\mathbf{I} \mathbf{0})\)
\[ \mathbf{x} = (u, v, w)^T \]

The vectors \( \mathbf{x}, \mathbf{r}, \) and \( \mathbf{Kx} \) are coplanar
Recap: Epipolar Geometry – Calibrated Case

\[ x \cdot [I \times (Rx')] = 0 \quad \Rightarrow \quad x' = EKx \]

- The calibration matrices \( K \) and \( K' \) of the two cameras are unknown.
- We can write the epipolar constraint in terms of unknown normalized coordinates:
  \[ x' = \hat{K} \hat{x}, \quad x = \hat{K}' \hat{x} \]

Recap: Epipolar Geometry – Uncalibrated Case

\[ Fx' = \text{the epipolar line associated with} \ x' (l = Fx) \]

\[ F'x = \text{the epipolar line associated with} \ x (l' = F'x) \]

Recap: The Eight-Point Algorithm

\[ x = (u, v, 1)^T, \quad x' = (u', v', 1)^T \]

\[ \begin{pmatrix} F_{21} & F_{22} & F_{23} \\ F_{31} & F_{32} & F_{33} \end{pmatrix} \begin{pmatrix} u' \\ v' \end{pmatrix} = 0 \]

1.) Solve with SVD.
2.) Enforce rank-2 constraint using SVD

- Problem: poor numerical conditioning

\[ \sum_{i=1}^{2}\left(x_iF\tilde{x}_j\right)^2 \]
Recap: Normalized Eight-Point Alg.

1. Center the image data at the origin, and scale it so the mean squared distance between the origin and the data points is 2 pixels.
2. Use the eight-point algorithm to compute $F$ from the normalized points.
3. Enforce the rank-2 constraint using SVD.
   $$ F = U D V^T $$
   where $D$ is a diagonal matrix with the singular values of $F$.
   Set $d_{max}$ to zero and reconstruct $F$.
4. Transform fundamental matrix back to original units: if $T$ and $T'$ are the normalizing transformations in the two images, than the fundamental matrix in original coordinates is $T^T F T'$.

Recap: Comparison of Estimation Algorithms

<table>
<thead>
<tr>
<th></th>
<th>8-point</th>
<th>Normalized 8-point</th>
<th>Nonlinear least squares</th>
</tr>
</thead>
<tbody>
<tr>
<td>Av. Dist. 1</td>
<td>2.33 pixels</td>
<td>0.92 pixel</td>
<td>0.86 pixel</td>
</tr>
<tr>
<td>Av. Dist. 2</td>
<td>2.18 pixels</td>
<td>0.85 pixel</td>
<td>0.80 pixel</td>
</tr>
</tbody>
</table>

Recap: Epipolar Transfer

- Assume the epipolar geometry is known
- Given projections of the same point in two images, how can we compute the projection of that point in a third image?

$$ l_{1i} = F^T x_j $$
$$ l_{2i} = F^T x_j $$

Recap: Structure from Motion

- Given: $m$ images of $n$ fixed 3D points
  $$ x_{ij} = P_i X_j, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n $$
- Problem: estimate $m$ projection matrices $P_i$, and $n$ 3D points $X_j$ from the $mn$ correspondences $x_{ij}$
Recap: Structure from Motion Ambiguity

- If we scale the entire scene by some factor $k$ and, at the same time, scale the camera matrices by the factor of $1/k$, the projections of the scene points in the image remain exactly the same.
- More generally: if we transform the scene using a transformation $Q$ and apply the inverse transformation to the camera matrices, then the images do not change

$$ x = PX = (PQ^{-1})QX $$

Recap: Hierarchy of 3D Transformations

- With no constraints on the camera calibration matrix or on the scene, we get a projective reconstruction.
- Need additional information to upgrade the reconstruction to affine, similarity, or Euclidean.

Any More Questions?

*Good luck for the exam!*