Computer Vision – Lecture 18

Repetition

09.07.2019

Bastian Leibe
Visual Computing Institute
RWTH Aachen University
http://www.vision.rwth-aachen.de/

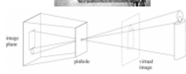
leibe@vision.rwth-aachen.de

Announcements

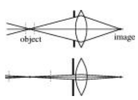
- Today, I'll summarize the most important points from the lecture.
  - It is an opportunity for you to ask questions…
  - …or get additional explanations about certain topics.
  - *So, please do ask.*
- Today's slides are intended as an index for the lecture.
  - But they are not complete, won't be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

Repetition

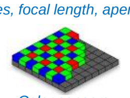
- Image Processing Basics
  - Image Formation
  - Linear Filters
  - Edge & Structure Extraction
- Segmentation & Grouping
- Object Recognition
- Local Features & Matching
- Deep Learning
- 3D Reconstruction



Pinhole camera model



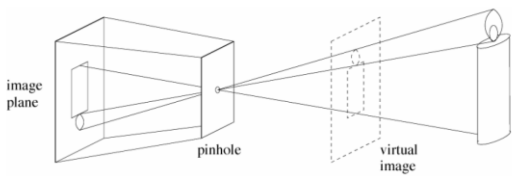
Lenses, focal length, aperture



Color sensors

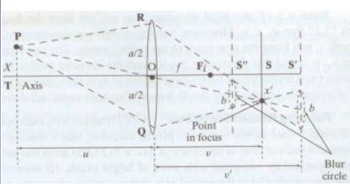
Recap: Pinhole Camera

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



Source: Forsyth & Ponce

Recap: Focus and Depth of Field



"circles of confusion"

Thin lens: scene points at distinct depths come in focus at different image planes.

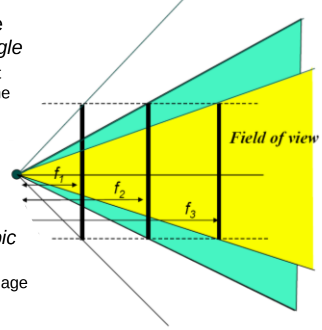
(Real camera lens systems have greater depth of field.)

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

Source: Shapiro & Stockman

Recap: Field of View and Focal Length

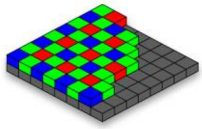
- As *f* gets smaller, image becomes more *wide angle*
  - More world points project onto the finite image plane
- As *f* gets larger, image becomes more *telescopic*
  - Smaller part of the world projects onto the finite image plane



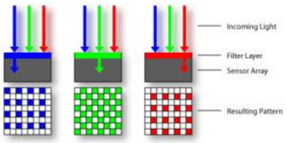
from R. Duraiswami

Recap: Color Sensing in Digital Cameras

Bayer grid



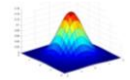
Estimate missing components from neighboring values (demosaicing)



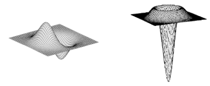
Source: Steve Seitz

Repetition

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



Gaussian Smoothing

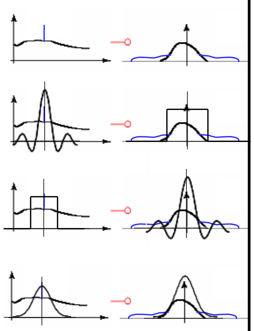


Derivative operators

Gaussian/Laplacian pyramid

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.



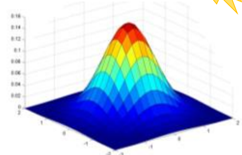
Recap: Gaussian Smoothing

see Exercise 1.1!

- Gaussian kernel

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

- 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



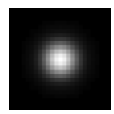
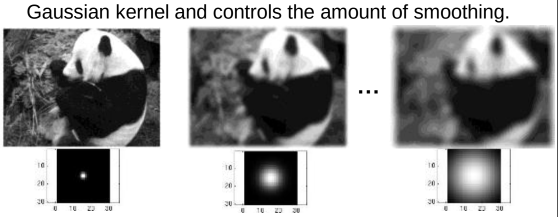


Image Source: Forsyth & Ponce

Recap: Smoothing with a Gaussian

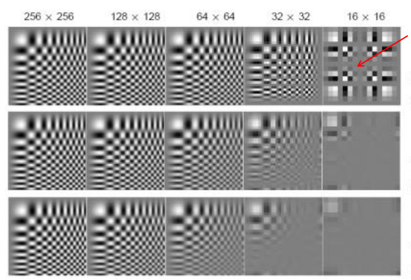
- Parameter σ is the "scale" / "width" / "spread" of the Gaussian kernel and controls the amount of smoothing.



```
for sigma=1:3:10
  h = fspecial('gaussian', fsize, sigma);
  out = imfilter(im, h);
  imshow(out);
  pause;
end
```

Slide credit: Kristen Grauman

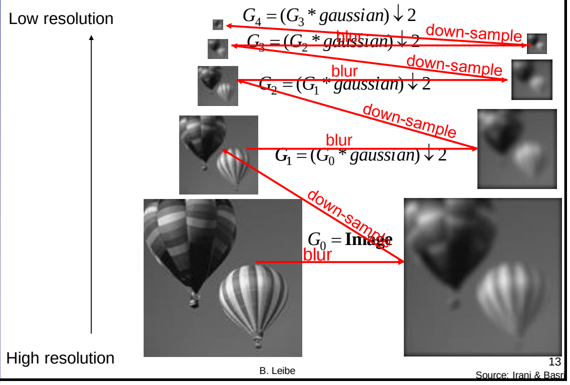
Recap: Resampling with Prior Smoothing



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

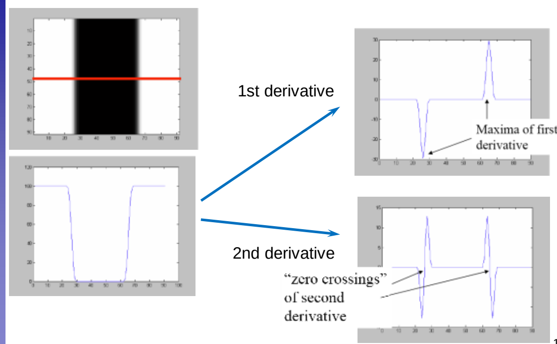
Image Source: Forsyth & Ponce

Recap: The Gaussian Pyramid



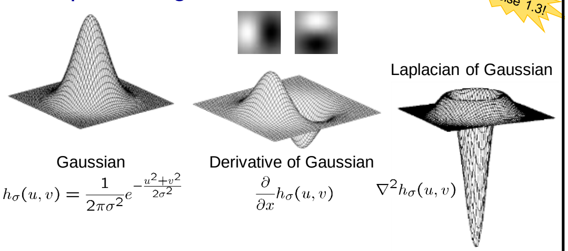
Source: Irani & Basri

Recap: Derivatives and Edges…



Recap: 2D Edge Detection Filters

see Exercise 1.3!



$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$ $\quad \frac{\partial}{\partial x} h_\sigma(u,v)$ $\quad \nabla^2 h_\sigma(u,v)$

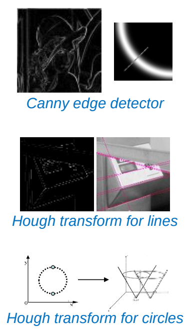
- ∇^2 is the Laplacian operator:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Slide credit: Kristen Grauman

Repetition

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



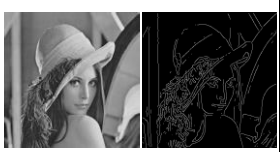
Recap: Canny Edge Detector

see Exercise 1.4!

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

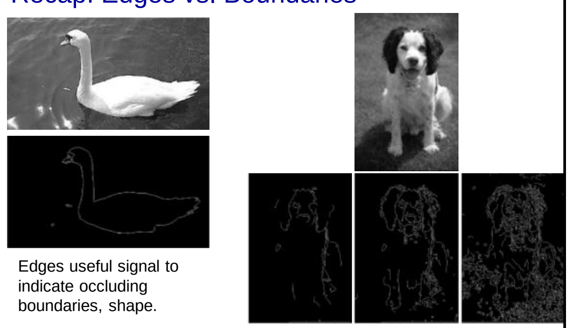
- MATLAB:

```
>> edge(image,'canny');
>> help edge
```



adapted from D. Lowe, L. Fei-Fei

Recap: Edges vs. Boundaries



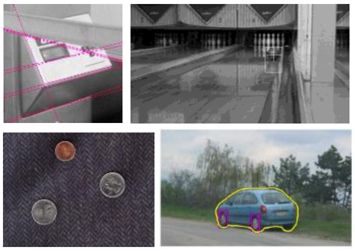
Edges useful signal to indicate occluding boundaries, shape.

Here the raw edge output is not so bad…

…but quite often boundaries of interest are fragmented, and we have extra "clutter" edge points.

Slide credit: Kristen Grauman

Recap: Fitting and Hough Transform



Given a model of interest, we can overcome some of the missing and noisy edges using fitting techniques.

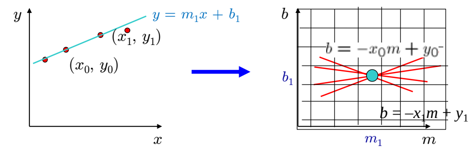
With voting methods like the Hough transform, detected points vote on possible model parameters.

Computer Vision Summer'19

Slide credit: Kristen Grauman

Recap: Hough Transform

see Exercise 1.5!



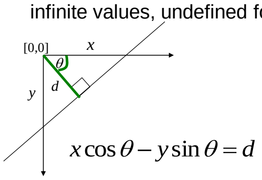
- How can we use this to find the most likely parameters (m,b) for the most prominent line in the image space?
  - Let each edge point in image space *vote* for a set of possible parameters in Hough space
  - Accumulate votes in discrete set of bins; parameters with the most votes indicate line in image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

Computer Vision Summer'19

Slide credit: Steve Seitz

Recap: Hough Transf. Polar Parametrization

- Usual (m,b) parameter space problematic: can take on infinite values, undefined for vertical lines.

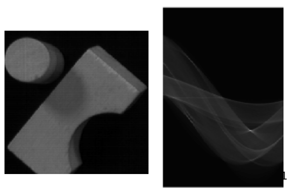


$$x\cos\theta - y\sin\theta = d$$

d : perpendicular distance from line to origin

θ : angle the perpendicular makes with the x-axis

- Point in image space ⇒ sinusoid segment in Hough space



Computer Vision Summer'19

Slide credit: Steve Seitz

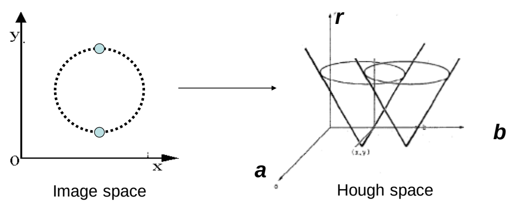
Recap: Hough Transform for Circles

see Exercise 1.5!

- Circle: center (a,b) and radius r

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

- For an unknown radius r, unknown gradient direction

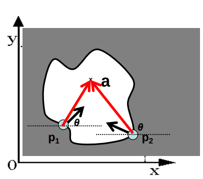


Computer Vision Summer'19

Slide credit: Kristen Grauman

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 θ.

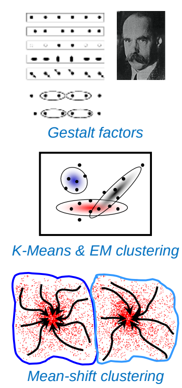
D.H. Ballard, Generalizing the Hough Transform to Detect Arbitrary Shapes, 1980.

Computer Vision Summer'19

Slide credit: Kristen Grauman

Repetition

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



Computer Vision Summer'19

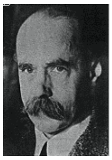
B. Leibe

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)

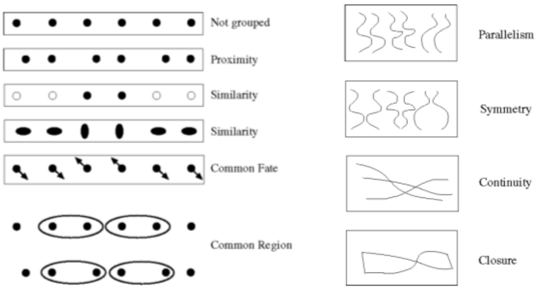
"I stand at the window and see a house, trees, sky. Theoretically I might say there were 327 brightnesses and nuances of colour. Do I have "327"? No. I have sky, house, and trees."

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

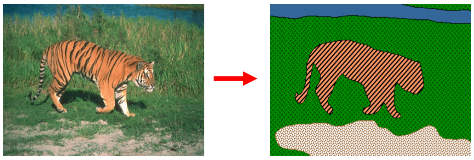
Recap: Gestalt Factors



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

Recap: Image Segmentation

- Goal: identify groups of pixels that go together



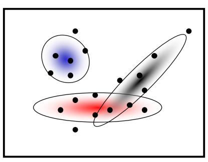
Recap: K-Means Clustering

- Basic idea: randomly initialize the *k* cluster centers, and iterate between the two following steps
  1. Randomly initialize the cluster centers, $c_1, ..., c_K$
  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:

$$\sum_{\text{clusters } i} \sum_{\text{points p in cluster } i} \|p - c_i\|^2$$

Recap: Expectation Maximization (EM)



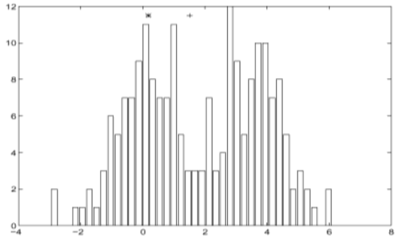
- Goal
  - Find blob parameters θ that maximize the likelihood function:

$$p(data|\theta) = \prod_{n=1}^{N} p(\mathbf{x}_n|\theta)$$

- 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

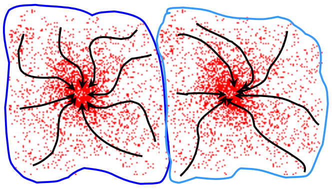
see Exercise 3.1!



- Iterative Mode Search
  1. Initialize random seed, and window W
  2. Calculate center of gravity (the "mean") of W: $\sum_{x \in W} x H(x)$
  3. Shift the search window to the mean
  4. Repeat Step 2 until convergence

Recap: Mean-Shift Clustering

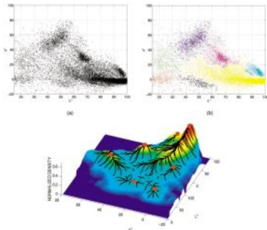
- Cluster: all data points in the attraction basin of a mode
- Attraction basin: the region for which all trajectories lead to the same mode



Recap: Mean-Shift Segmentation

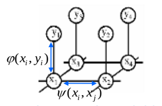
see Exercise 2.1!

- 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

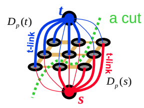


Repetition

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



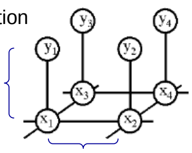
Markov Random Fields



Graph cuts

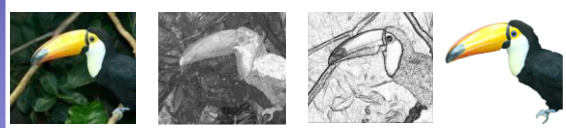
Recap: MRFs for Image Segmentation

- MRF formulation



$\Rightarrow$ Minimize the energy

$$E(\mathbf{x},\mathbf{y}) = \sum_i \phi(x_i, y_i) + \sum_{i,j} \psi(x_i, x_j)$$

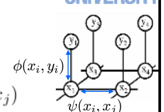


Data (D) Unary likelihood Pair-wise Terms MAP Solution

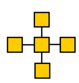
Recap: Energy Formulation

- Energy function

$$E(\mathbf{x},\mathbf{y}) = \underbrace{\sum_i \phi(x_i, y_i)}_{\text{Unary potentials}} + \underbrace{\sum_{i,j} \psi(x_i, x_j)}_{\text{Pairwise potentials}}$$



- Unary potentials ϕ
  - Encode local information about the given pixel/patch
  - How likely is a pixel/patch to belong to a certain class (e.g. foreground/background)?
- Pairwise potentials ψ
  - Encode neighborhood information
  - How different is a pixel/patch's label from that of its neighbor? (e.g. based on intensity/color/texture difference, edges)

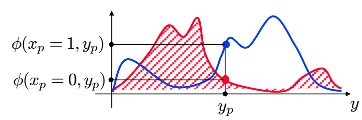


Recap: How to Set the Potentials?

- Unary potentials
  - E.g. color model, modeled with a Mixture of Gaussians

$$\phi(x_i, y_i; \theta_\phi) = \log \sum_k \theta_\phi(x_i, k) p(k|x_i) \mathcal{N}(y_i; \bar{y}_k, \Sigma_k)$$

$\Rightarrow$ Learn color distributions for each label



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
    $$g_{ij}(y) = e^{-\beta \|y_i - y_j\|^2} \qquad \beta = 2 / avg\left(\|y_i - y_j\|^2\right)$$

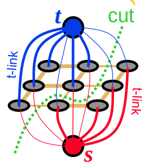
$\Rightarrow$ Discourages label changes except in places where there is also a large change in the observations.

Recap: Graph-Cuts Energy Minimization

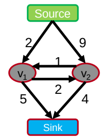
see Exercise 2.2!

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

  $\Rightarrow$ Solution is equivalent to minimum of the energy.



- s-t Mincut can be solved efficiently
  - Dual to the well-known max flow problem
  - Very efficient algorithms available for regular grid graphs (1-2 MPixels/s)
  - Globally optimal result for 2-class problems



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

$$E(L) = \underbrace{\sum_p E_p(L_p)}_{\text{t-links}} + \underbrace{\sum_{pq \in N} E(L_p, L_q)}_{\text{n-links}} \qquad L_p \in \{s,t\}$$

(Unary potentials; Pairwise potentials)

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

E(L) can be minimized by s-t graph cuts $\Leftrightarrow$ $E(s,s) + E(t,t) \leq E(s,t) + E(t,s)$

Submodularity ("convexity")

- Submodularity is the discrete equivalent to convexity.
  - Implies that every local energy minimum is a global minimum.

  $\Rightarrow$ Solution will be globally optimal.

First Applications Take Up Shape…

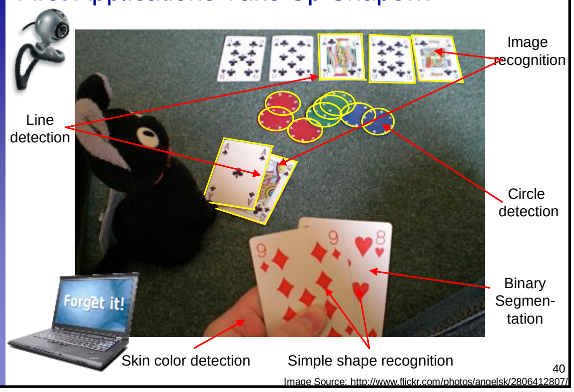
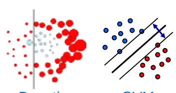


Image Source: http://www.flickr.com/photos/angelsk/2806412807/

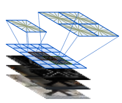
Repetition

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

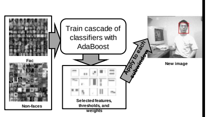
Sliding window principle



Boosting *SVM*



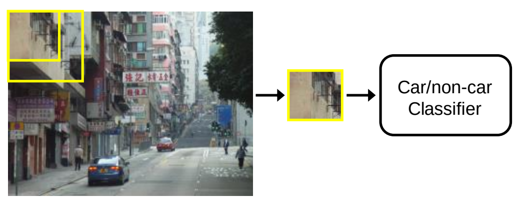
HOG detector



Viola-Jones face detector

Recap: Sliding-Window Object Detection

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

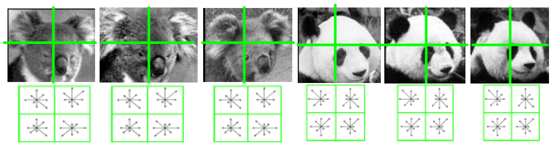


- Essentially, this is a brute-force approach with many local decisions.

Slide credit: Kristen Grauman

Recap: Gradient-based Representations

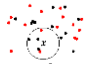
- Consider edges, contours, and (oriented) intensity gradients



- Summarize local distribution of gradients with histogram
  - Locally orderless: offers invariance to small shifts and rotations
  - Contrast-normalization: try to correct for variable illumination

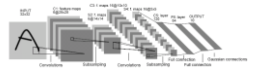
Classifier Construction: Many Choices…

Nearest Neighbor



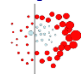
Berg, Berg, Malik 2005,
Chum, Zisserman 2007,
Boiman, Shechtman, Irani 2008, …

Neural networks



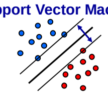
LeCun, Bottou, Bengio, Haffner 1998
Rowley, Baluja, Kanade 1998
…

Boosting



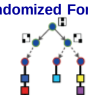
Viola, Jones 2001,
Torralba et al. 2004,
Opelt et al. 2006,
Benenson 2012, …

Support Vector Machines



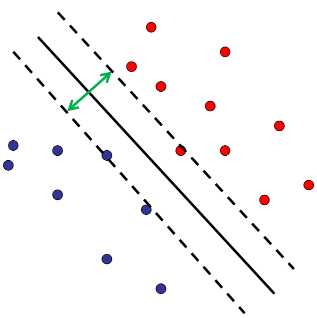
Vapnik, Schölkopf 1995,
Papageorgiou, Poggio '01,
Dalal, Triggs 2005,
Vedaldi, Zisserman 2012

Randomized Forests



Amit, Geman 1997,
Breiman 2001,
Lepetit, Fua 2006,
Gall, Lempitsky 2009,…

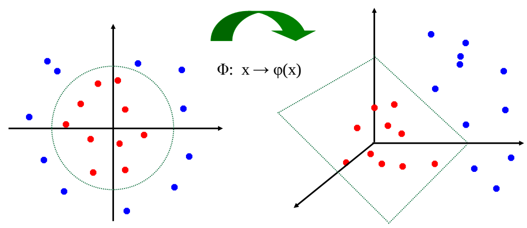
Recap: Support Vector Machines (SVMs)



- 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

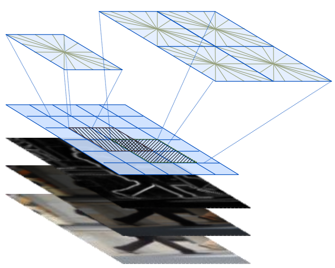
- General idea: The original input space can be mapped to some higher-dimensional feature space where the training set is separable:



Slide from Andrew Moore's tutorial: http://www.autonlab.org/tutorials/svm.html

Recap: HOG Descriptor Processing Chain

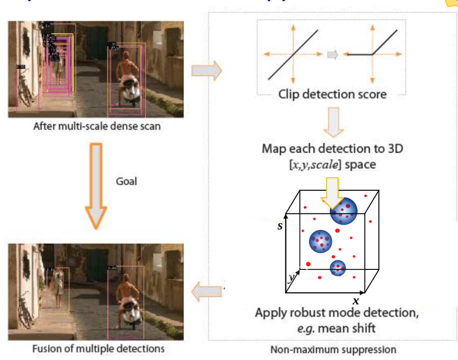
- SVM Classification
  - Typically using a linear SVM



Object/Non-object
↑
Linear SVM
↑
Collect HOGs over detection window
↑
Contrast normalize over overlapping spatial cells
↑
Weighted vote in spatial & orientation cells
↑
Compute gradients
↑
Gamma compression
↑
Image Window

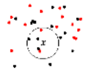
Recap: Non-Maximum Suppression

see Exercise 2.3!



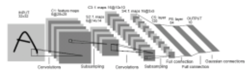
Classifier Construction: Many Choices…

Nearest Neighbor



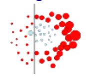
Shakhnarovich, Viola, Darrell 2003
Berg, Berg, Malik 2005,
Boiman, Shechtman, Irani 2008, …

Neural networks



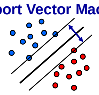
LeCun, Bottou, Bengio, Haffner 1998
Rowley, Baluja, Kanade 1998
…

Boosting



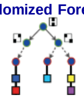
Viola, Jones 2001,
Torralba et al. 2004,
Opelt et al. 2006,
Benenson 2012, …

Support Vector Machines



Vapnik, Schölkopf 1995,
Papageorgiou, Poggio '01,
Dalal, Triggs 2005,
Vedaldi, Zisserman 2012

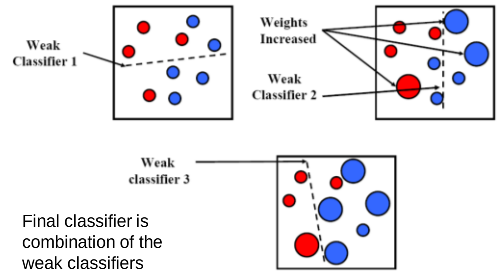
Randomized Forests



Amit, Geman 1997,
Breiman 2001,
Lepetit, Fua 2006,
Gall, Lempitsky 2009,…

Computer Vision Summer'19 | Slide adapted from Kristen Grauman | B. Leibe | 49

Recap: AdaBoost

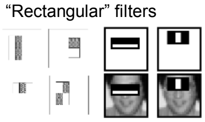


Final classifier is combination of the weak classifiers

Computer Vision Summer'19 | Slide credit: Kristen Grauman | B. Leibe | 50

Recap: Viola-Jones Face Detection

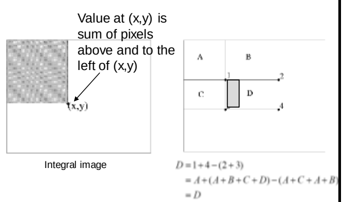
"Rectangular" filters



Feature output is difference between adjacent regions

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

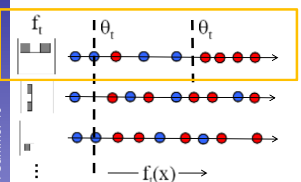
Avoid scaling images → scale features directly for same cost



Computer Vision Summer'19 | Slide credit: Kristen Grauman | B. Leibe | 51 | [Viola & Jones, CVPR 2001]

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.



Outputs of a possible rectangle feature on faces and non-faces.

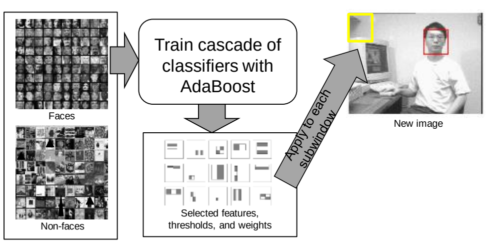
Resulting weak classifier:

$$h_t(x) = \begin{cases} +1 & \text{if } f_t(x) > \theta_t \\ -1 & \text{otherwise} \end{cases}$$

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

Computer Vision Summer'19 | Slide credit: Kristen Grauman | B. Leibe | 52 | [Viola & Jones, CVPR 2001]

Recap: Viola-Jones Face Detector



- 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/]

Computer Vision Summer'19 | Slide credit: Kristen Grauman | B. Leibe | 53

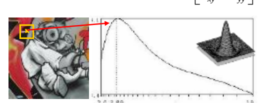
Repetition

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

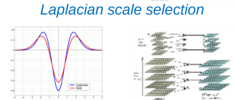
$$M(\sigma_I, \sigma_D) = g(\sigma_I) * \begin{bmatrix} I_x^2(\sigma_D) & I_x I_y(\sigma_D) \\ I_x I_y(\sigma_D) & I_y^2(\sigma_D) \end{bmatrix}$$

Harris & Hessian detector

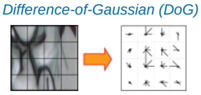
$$Hes(I) = \begin{bmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{bmatrix}$$



Laplacian scale selection



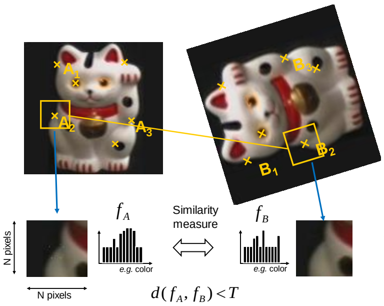
Difference-of-Gaussian (DoG)



SIFT descriptor

Computer Vision Summer'19 | B. Leibe | 54

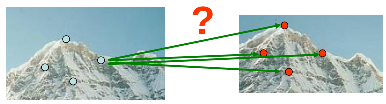
Recap: Local Feature Matching Pipeline



1. Find a set of distinctive key-points
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!

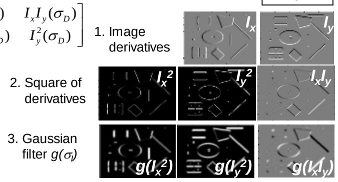
Slide credit: Darya Frolova, Denis Simakov

Recap: Harris Detector [Harris88]

see Exercise 3.2!

- Compute second moment matrix (autocorrelation matrix)

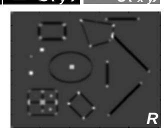
$$M(\sigma_I, \sigma_D) = g(\sigma_I) * \begin{bmatrix} I_x^2(\sigma_D) & I_x I_y(\sigma_D) \\ I_x I_y(\sigma_D) & I_y^2(\sigma_D) \end{bmatrix}$$



4. Cornerness function – two strong eigenvalues

$$R = \det[M(\sigma_I, \sigma_D)] - \alpha[\operatorname{trace}(M(\sigma_I, \sigma_D))]$$

$$= g(I_x^2)g(I_y^2) - [g(I_x I_y)]^2 - \alpha[g(I_x^2) + g(I_y^2)]^2$$

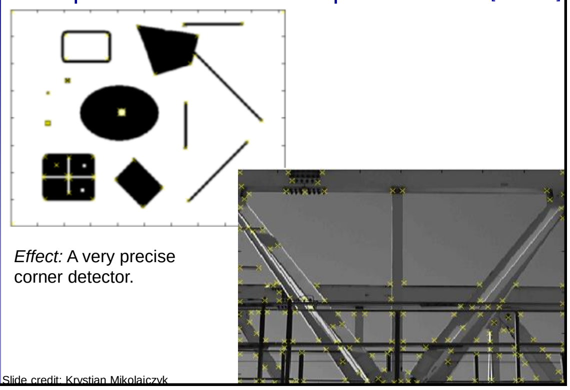


5. Perform non-maximum suppression

Slide credit: Krystian Mikolajczyk

Recap: Harris Detector Responses

[Harris88]



Effect: A very precise corner detector.

Slide credit: Krystian Mikolajczyk

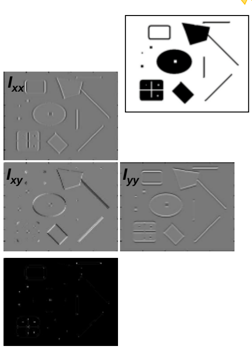
Recap: Hessian Detector

[Beaudet78]

see Exercise 3.2!

- Hessian determinant

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



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

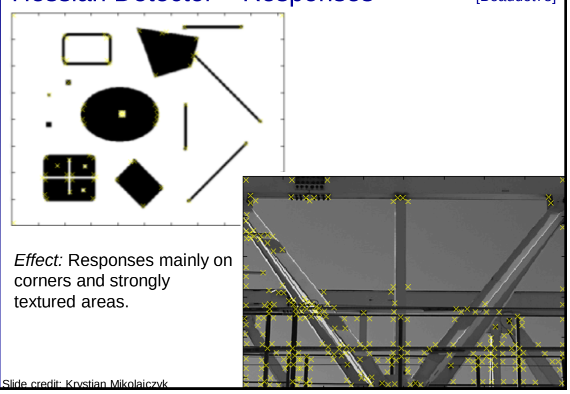
In Matlab:

$$I_{xx} .* I_{yy} - (I_{xy}).\hat{}2$$

Slide credit: Krystian Mikolajczyk

Hessian Detector – Responses

[Beaudet78]

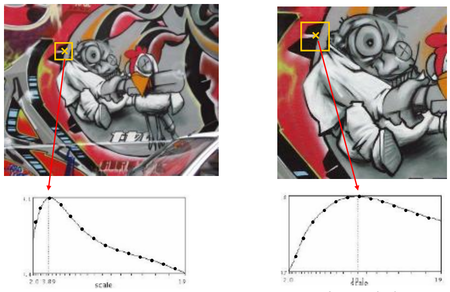


Effect: Responses mainly on corners and strongly textured areas.

Slide credit: Krystian Mikolajczyk

Recap: Automatic Scale Selection

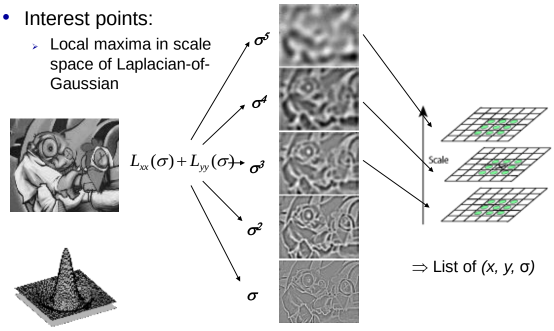
- Function responses for increasing scale (scale signature)



$f(I_{i_1 \ldots i_m}(x,\sigma))$ $f(I_{i_1 \ldots i_m}(x',\sigma'))$

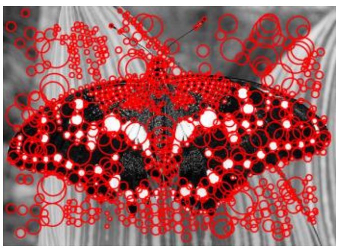
Recap: Laplacian-of-Gaussian (LoG)

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



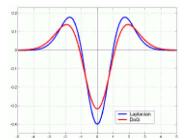
$\Rightarrow$ List of *(x, y, σ)*

Recap: LoG Detector Responses

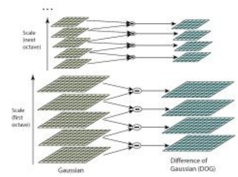


Recap: Key point localization with DoG

- 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



Candidate keypoints: list of (x,y,σ)

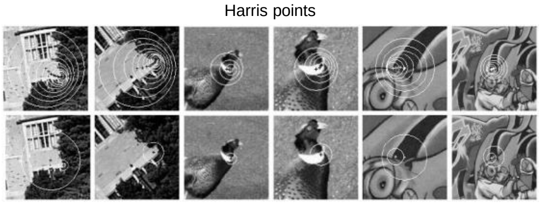
Image source: David Lowe

Recap: Harris-Laplace

[Mikolajczyk '01]

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

Harris points

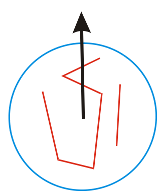


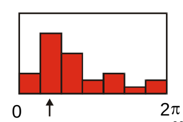
Harris-Laplace points

Recap: Orientation Normalization

[Lowe, SIFT, 1999]

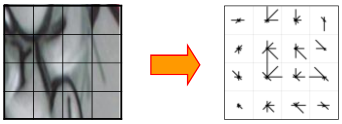
- Compute orientation histogram
- Select dominant orientation
- Normalize: rotate to fixed orientation





Recap: SIFT Feature Descriptor

- Scale Invariant Feature Transform
- Descriptor computation:
  - Divide patch into 4x4 sub-patches: 16 cells
  - Compute histogram of gradient orientations (8 reference angles) for all pixels inside each sub-patch
  - Resulting descriptor: 4x4x8 = 128 dimensions

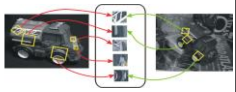


D.G. Lowe. "Distinctive image features from scale-invariant keypoints." *IJCV* 60 (2), pp. 91-110, 2004.

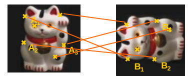
Slide credit: Svetlana Lazebnik

Repetition

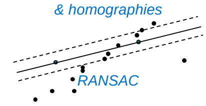
- Image Processing Basics
- Segmentation & Grouping
- Object Recognition
- Local Features & Matching
  - Local Features – Detection and Description
  - Recognition with Local Features
- Deep Learning
- 3D Reconstruction



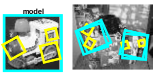
Recognition pipeline



Fitting affine transformations & homographies



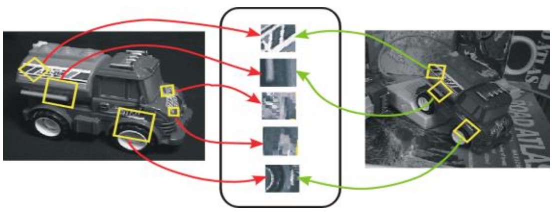
RANSAC



Gen. Hough Transform

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

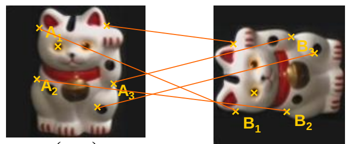


Local Features, e.g. SIFT

Slide credit: David Lowe

Recap: Fitting an Affine Transformation

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



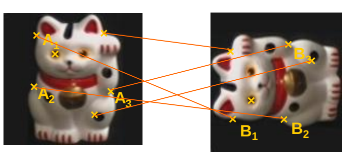
(x_i, y_i) (x_i', y_i')

$$\begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

$$\begin{bmatrix} & & \cdots & & & \\ x_i & y_i & 0 & 0 & 1 & 0 \\ 0 & 0 & x_i & y_i & 0 & 1 \\ & & \cdots & & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} \cdots \\ x_i' \\ y_i' \\ \cdots \end{bmatrix}$$

Recap: Fitting a Homography

- Estimating the transformation



$\mathbf{x}_{A_1} \leftrightarrow \mathbf{x}_{B_1}$
$\mathbf{x}_{A_2} \leftrightarrow \mathbf{x}_{B_2}$
$\mathbf{x}_{A_3} \leftrightarrow \mathbf{x}_{B_3}$
$\vdots$

Homogenous coordinates

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Image coordinates

$$\begin{bmatrix} x'' \\ y'' \\ 1 \end{bmatrix} = \frac{1}{z'} \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix}$$

Matrix notation

$$x' = Hx$$

$$x'' = \frac{1}{z'} x'$$

$$x_{A_1} = \frac{h_{11} x_{B_1} + h_{12} y_{B_1} + h_{13}}{h_{31} x_{B_1} + h_{32} y_{B_1} + 1} \qquad y_{A_1} = \frac{h_{21} x_{B_1} + h_{22} y_{B_1} + h_{23}}{h_{31} x_{B_1} + h_{32} y_{B_1} + 1}$$

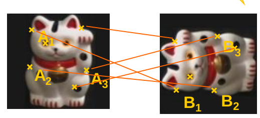
Slide credit: Krystian Mikolajczyk

Recap: Fitting a Homography

see Exercise 4.2!

- Estimating the transformation

$$h_{11} x_{B_1} + h_{12} y_{B_1} + h_{13} - x_{A_1} h_{31} x_{B_1} - x_{A_1} h_{32} y_{B_1} - x_{A_1} = 0$$
$$h_{21} x_{B_1} + h_{22} y_{B_1} + h_{23} - y_{A_1} h_{31} x_{B_1} - y_{A_1} h_{32} y_{B_1} - y_{A_1} = 0$$



$\mathbf{x}_{A_1} \leftrightarrow \mathbf{x}_{B_1}$
$\mathbf{x}_{A_2} \leftrightarrow \mathbf{x}_{B_2}$
$\mathbf{x}_{A_3} \leftrightarrow \mathbf{x}_{B_3}$
$\vdots$

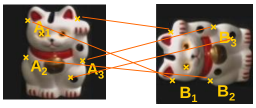
$$\begin{bmatrix} x_{B_1} & y_{B_1} & 1 & 0 & 0 & 0 & -x_{A_1}x_{B_1} & -x_{A_1}y_{B_1} & -x_{A_1} \\ 0 & 0 & 0 & x_{B_1} & y_{B_1} & 1 & -y_{A_1}x_{B_1} & -y_{A_1}y_{B_1} & -y_{A_1} \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \end{bmatrix} \cdot \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ . \\ . \\ . \end{bmatrix}$$

$$Ah = 0$$

Slide credit: Krystian Mikolajczyk

Recap: Fitting a Homography

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



$$\mathbf{x}_{A_1} \leftrightarrow \mathbf{x}_{B_1}$$
$$\mathbf{x}_{A_2} \leftrightarrow \mathbf{x}_{B_2}$$
$$\mathbf{x}_{A_3} \leftrightarrow \mathbf{x}_{B_3}$$
$$\vdots$$

$$Ah = 0$$

SVD

$$\mathbf{A} = \mathbf{UDV}^T = \mathbf{U}\begin{bmatrix} d_{11} & \cdots & d_{19} \\ \vdots & \ddots & \vdots \\ d_{91} & \cdots & d_{99} \end{bmatrix} \begin{bmatrix} v_{11} & \cdots & v_{19} \\ \vdots & \ddots & \vdots \\ v_{91} & \cdots & v_{99} \end{bmatrix}^T$$

$$\mathbf{h} = \frac{[v_{19}, \cdots, v_{99}]}{v_{99}}$$

Minimizes least square error

Recap: RANSAC

see Exercise 5.1!

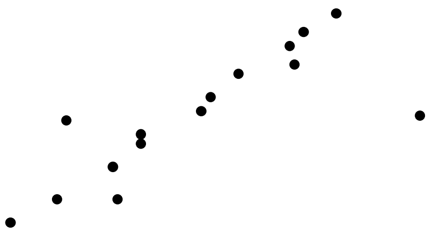
RANSAC loop:

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

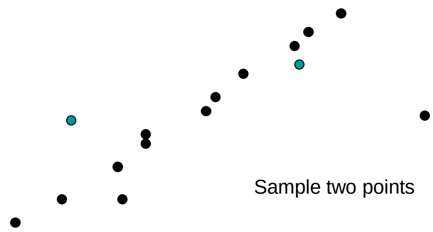
Recap: RANSAC Line Fitting Example

- Task: Estimate the best line



Recap: RANSAC Line Fitting Example

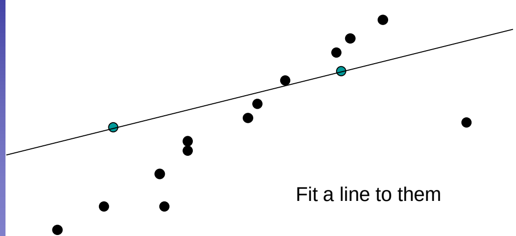
- Task: Estimate the best line



Sample two points

Recap: RANSAC Line Fitting Example

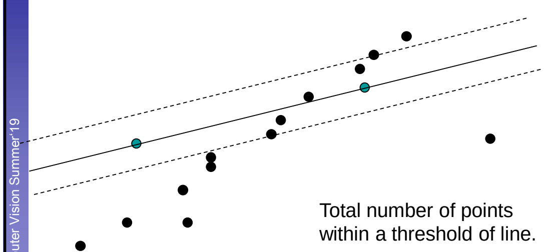
- Task: Estimate the best line



Fit a line to them

Recap: RANSAC Line Fitting Example

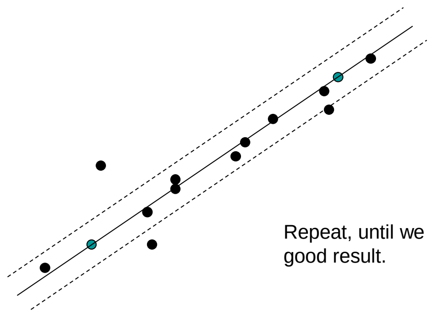
- Task: Estimate the best line



Total number of points within a threshold of line.

Recap: RANSAC Line Fitting Example

- Task: Estimate the best line

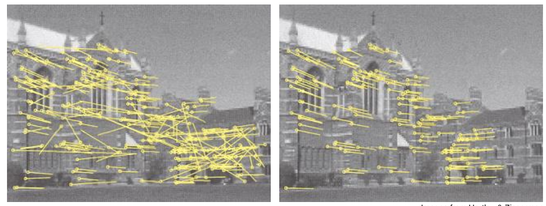


Repeat, until we get a good result.

Recap: Feature Matching Example

- Find best stereo match within a square search window (here 300 pixels^2)
- Global transformation model: epipolar geometry

before RANSAC | after RANSAC

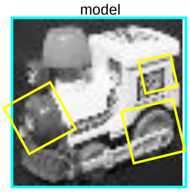


Images from Hartley & Zisserman

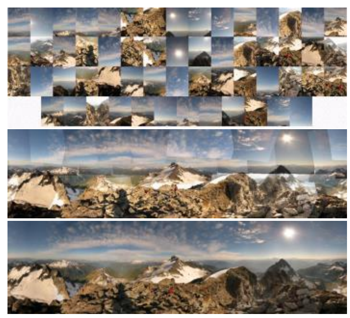
Recap: Generalized Hough Transform

- 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.

model



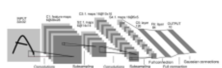
Application: Panorama Stitching



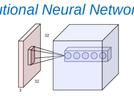
http://www.cs.ubc.ca/~mbrown/autostitch/autostitch.html

Repetition

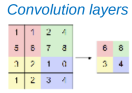
- Image Processing Basics
- Segmentation & Grouping
- Object Recognition
- Local Features & Matching
- Deep Learning
  - Convolutional Neural Networks (CNNs)
  - Deep Learning Background
  - CNNs for Object Detection
  - CNNs for Semantic Segmentation
  - CNNs for Matching & RNNs
- 3D Reconstruction



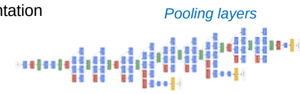
Convolutional Neural Networks



Convolution layers



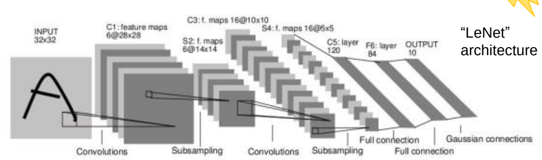
Pooling layers



AlexNet, VGGNet, GoogLeNet, ResNet

Recap: Convolutional Neural Networks

see Exercise 4.1!



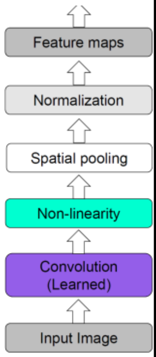
"LeNet" architecture

- 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

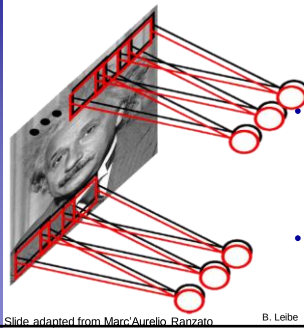
Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86(11): 2278–2324, 1998.

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

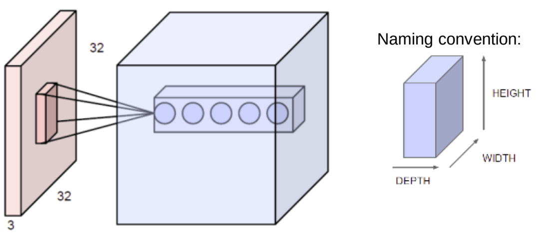


Recap: Intuition of CNNs



- 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!

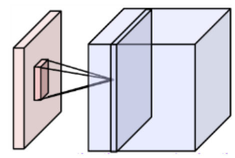
Recap: Convolution Layers



- 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

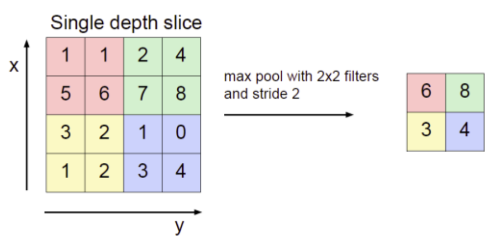
5×5 filters



Activation maps

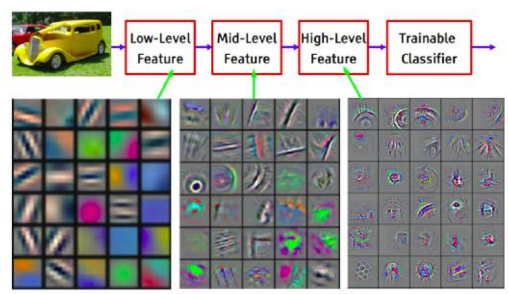
Each activation map is a depth slice through the output volume.

Recap: Pooling Layers



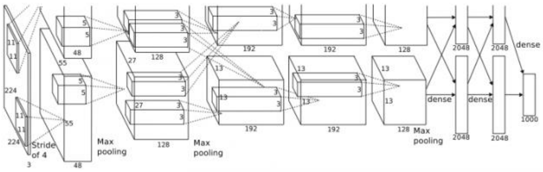
- Effect:
  - Make the representation smaller without losing too much information
  - Achieve robustness to translations

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 (10^6 images instead of 10^3)
  - GPU implementation
  - Better regularization and up-to-date tricks for training (Dropout)

A. Krizhevsky, I. Sutskever, and G. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012.

Image source: A. Krizhevsky, I. Sutskever and G.E. Hinton, NIPS 2012

Recap: VGGNet (2014/15)

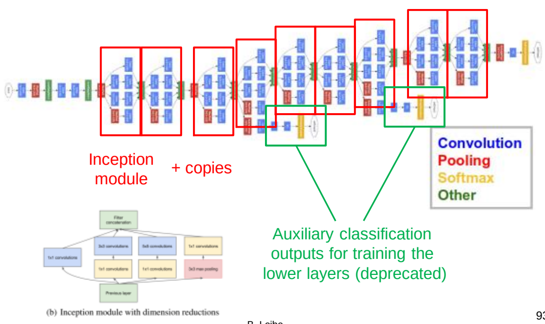
- Main ideas
  - Deeper network
  - Stacked convolutional layers with smaller filters (+ nonlinearity)
  - Detailed evaluation of all components
- Results
  - Improved ILSVRC top-5 error rate to 6.7%.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

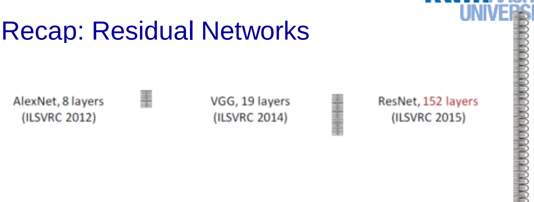
Mainly used

Image source: Simonyan & Zisserman

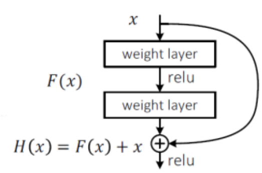
Recap: GoogLeNet (2014)



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

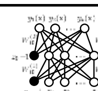
3. If you have a medium sized dataset, "finetune" instead: use the old weights as initialization, train the full network or only some of the higher layers.

Retrain bigger part of the network

Slide credit: Andrej Karpathy

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

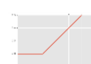


MLPs

Backpropagation Algorithm

$$w_{kj}^{(\tau+1)} = w_{kj}^{(\tau)} - \eta \left. \frac{\partial E(\mathbf{w})}{\partial w_{kj}} \right|_{\mathbf{w}^{(\tau)}}$$

Gradient Descent

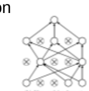


ReLU

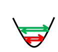
$$\mathrm{Var}(W_i) = \frac{1}{n_{\text{in}}}$$

$$\mathrm{Var}(W) = \frac{2}{n_{\text{in}}}$$

Glorot & He Initialization



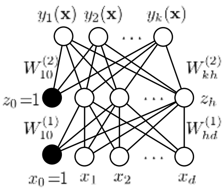
Dropout



Learning Rate

Recap: Multi-Layer Perceptrons

- Deep network = Also learning the feature transformation



Output layer

Hidden layer

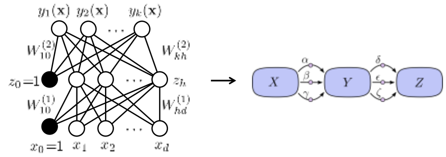
Mapping (learned!)

Input layer

- Output

$$y_k(\mathbf{x}) = g^{(2)}\left(\sum_{i=0}^{h} W_{ki}^{(2)} g^{(1)}\left(\sum_{j=0}^{d} W_{ij}^{(1)} x_j\right)\right)$$

Recap: Backpropagation Algorithm



- General formulation (used in deep learning packages)
  - Convert the network into a computational graph.
  - Perform reverse-mode-differentiation this graph
  - Each new layer/module just needs to specify how it affects the
    - forward pass $\mathbf{y} = \text{module.}\mathbf{fprop}(\mathbf{x})$
    - backward pass $\frac{\partial E}{\partial \mathbf{x}} = \text{module.}\mathbf{bprop}(\frac{\partial E}{\partial \mathbf{y}})$

$\Rightarrow$ Very general framework, *any differentiable layer* can be used.

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

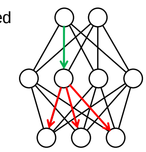
$$w_{kj}^{(\tau+1)} = w_{kj}^{(\tau)} - \eta \left.\frac{\partial E(\mathbf{w})}{\partial w_{kj}}\right|_{\mathbf{w}^{(\tau)}}$$

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

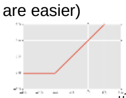
Recap: Practical Considerations

- Vanishing gradients problem
  - In multilayer nets, gradients need to be propagated through many layers
  - The magnitudes of the gradients are often very different for the different layers, especially if the initial weights are small.

  $\Rightarrow$ Gradients can get very small in the early layers of deep nets.



- When designing deep networks, we need to make sure gradients can be propagated throughout the network
  - By restricting the network depth (shallow networks are easier)
  - By very careful implementation (*numerics matter*!)
  - By choosing suitable nonlinearities (e.g., ReLU)
  - By performing proper initialization (Glorot, He)



Recap: Glorot Initialization

[Glorot & Bengio, '10]

- Variance of neuron activations
  - Suppose we have an input X with n components and a linear neuron with random weights W that spits out a number Y.
  - We *want the variance of the input and output of a unit to be the same*, therefore $n\ \text{Var}(W_i)$ should be 1. This means

$$\text{Var}(W_i) = \frac{1}{n} = \frac{1}{n_{\text{in}}}$$

  - Or for the backpropagated gradient

$$\text{Var}(W_i) = \frac{1}{n_{\text{out}}}$$

  - As a compromise, Glorot & Bengio propose to use

$$\text{Var}(W) = \frac{2}{n_{\text{in}} + n_{\text{out}}}$$

$\Rightarrow$ Randomly sample the initial weights with this variance.

Recap: He Initialization

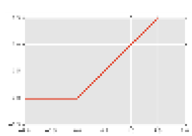
[He et al., '15]

- 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

$$\text{Var}(W) = \frac{2}{n_{\text{in}}}$$

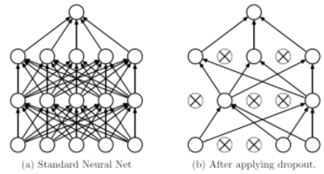
Recap: Batch Normalization

[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]

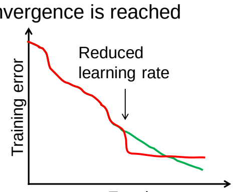


- Idea
  - Randomly switch off units during training.
  - Change network architecture for each data point, effectively training many different variants of the network.
  - When applying the trained network, multiply activations with the probability that the unit was set to zero.

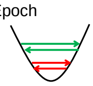
$\Rightarrow$ Improved performance

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.

Slide adapted from Geoff Hinton

Recap: Data Augmentation

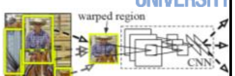
- Effect
  - Much larger training set
  - Robustness against expected variations
- During testing
  - When cropping was used during training, need to again apply crops to get same image size.
  - Beneficial to also apply flipping during test.
  - Applying several ColorPCA variations can bring another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

Augmented training data (from one original image)

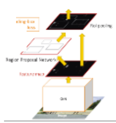
Image source: Lucas Beyer

Repetition

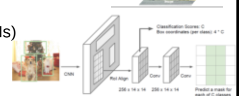
- Image Processing Basics
- Segmentation & Grouping
- Object Recognition
- Local Features & Matching
- Deep Learning
  - Convolutional Neural Networks (CNNs)
  - Deep Learning Background
  - CNNs for Object Detection
  - CNNs for Semantic Segmentation
  - CNNs for Matching & RNNs
- 3D Reconstruction



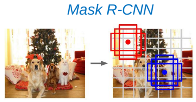
R-CNN



Faster R-CNN

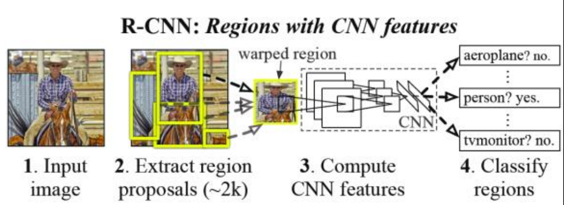


Mask R-CNN



YOLO / SSD

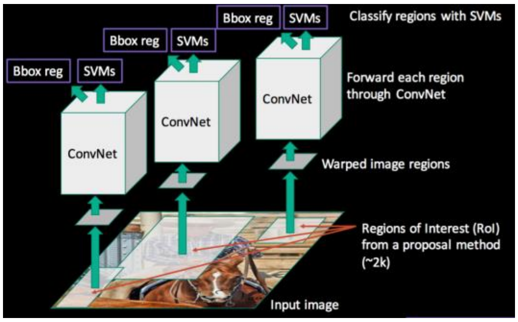
Recap: R-CNN for Object Detection



- 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

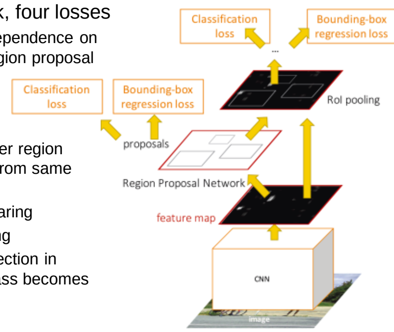
R. Girshick, J. Donahue, T. Darrell, and J. Malik, Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, CVPR 2014

Recap: R-CNN for Object Deteection

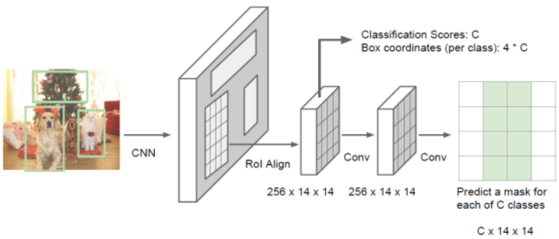


Recap: Faster R-CNN

- 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.

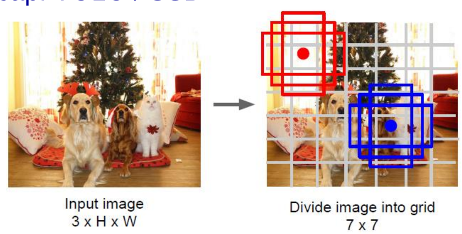


Recap: Mask R-CNN



K. He, G. Gkioxari, P. Dollar, R. Girshick, Mask R-CNN, arXiv 1703.06870.

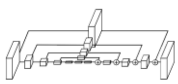
Recap: YOLO / SSD



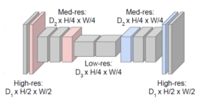
- Idea: Directly go from image to detection scores
- Within each grid cell
  - Start from a set of anchor boxes
  - Regress from each of the B anchor boxes to a final box
  - Predict scores for each of C classes (including background)

Repetition

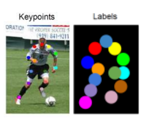
- Image Processing Basics
- Segmentation & Grouping
- Object Recognition
- Local Features & Matching
- Deep Learning
  - Convolutional Neural Networks (CNNs)
  - Deep Learning Background
  - CNNs for Object Detection
  - CNNs for Semantic Segmentation
  - CNNs for Matching & RNNs
- 3D Reconstruction



Fully Convolutional Networks



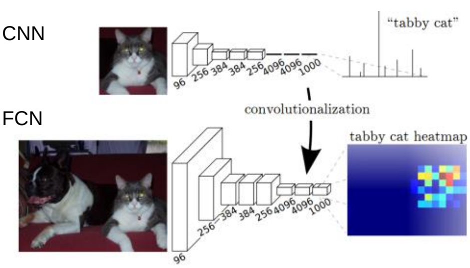
Encoder-Decoder Architecture



Human Pose Estimation

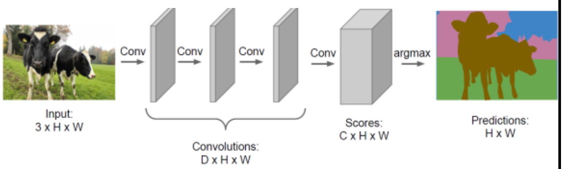
Recap: Fully Convolutional Networks

- CNN
- FCN



- Intuition
  - Think of FCNs as performing a sliding-window classification, producing a heatmap of output scores for each class

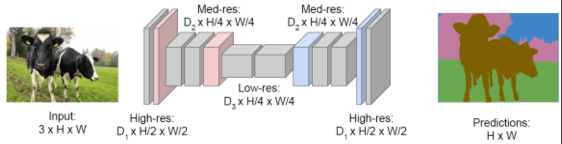
Recap: Fully-Convolutional Networks



- Design a network as a sequence of convolutional layers
  - To make predictions for all pixels at once
  - Fully Convolutional Networks (FCNs)
    - All operations formulated as convolutions
    - Fully-connected layers become 1×1 convolutions
    - Advantage: can process arbitrarily sized images

Slide adapted from FeiFei Li

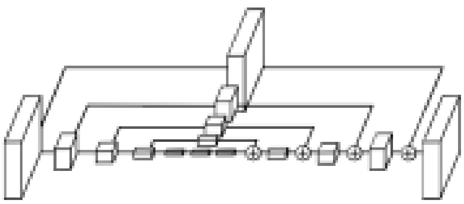
Recap: Encoder-Decorder Architecture



- 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

Slide credit: FeiFei Li

Recap: Skip Connections



- Encoder-Decoder Architecture with skip connections
  - Problem: downsampling loses high-resolution information
  - Use skip connections to preserve this higher-resolution information

Image source: Newell et al.

Recap: FCNs for Human Pose Estimation

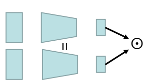
- Input data

- Formulate pose estimation as a segmentation problem
  - Annotate images with keypoints for skeleton joints
  - Define a target disk around each keypoint with radius r
  - Set the ground-truth label to 1 within each such disk
  - Infer heatmaps for the joints as in semantic segmentation

Slide adapted from Georgia Gkioxari

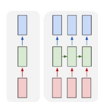
Repetition

- 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



Siamese Networks

Triplet Loss

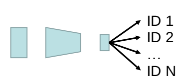


Recurrent Neural Networks

Recap: Types of Models used for Matching Tasks

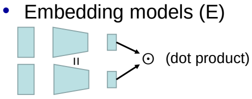
Training

- Identification models (I)



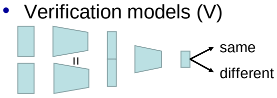
Multi-class classification loss

- Embedding models (E)



Large-margin loss, Triplet loss

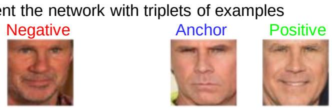
- Verification models (V)



Two-class classification loss

Triplet Loss Networks

- Learning a discriminative embedding
  - Present the network with triplets of examples



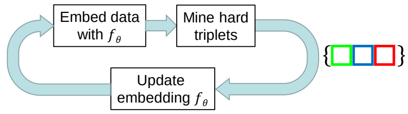
  - Apply triplet loss to learn an embedding $f(\cdot)$ that groups the positive example closer to the anchor than the negative one.

$$\|f(x_i^a) - f(x_i^p)\|_2^2 < \|f(x_i^a) - f(x_i^n)\|_2^2$$

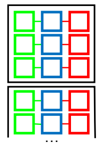
$\Rightarrow$ Used with great success in Google's FaceNet face identification

Offline Hard Triplet Mining

- Considerable effort needed



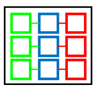
- Using the triplets for learning
  - Minibatch learning



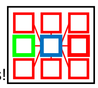
This is a very wasteful design!

Better: Online Hard Triplet Mining

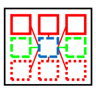
- Core idea
  - The minibatch contains many more potential triplets than the ones that were mined!
  - Why not make use of those also?



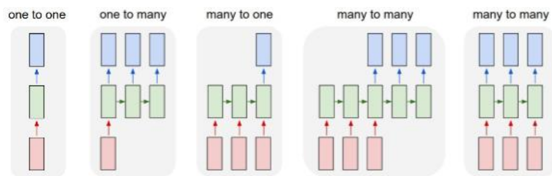
- Possible improvement
  - Each member of another triplet becomes an additional negative candidate
  - But: need both hard negatives *and* hard positives!



- Better design
  - Sample K images from P classes (=people) for each minibatch
  - Triplets are only constructed within the minibatch



Recap: Recurrent Neural Networks



- Up to now
  - Simple neural network structure: 1-to-1 mapping of inputs to outputs
- Recurrent Neural Networks
  - Generalize this to arbitrary mappings

Image source: Andrej Karpathy

Recap: RNNs

- 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.

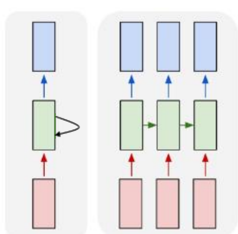
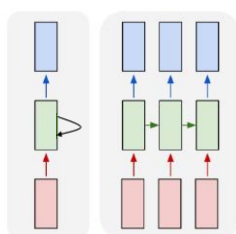


Image source: Andrej Karpathy

Recap: RNNs

- 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.

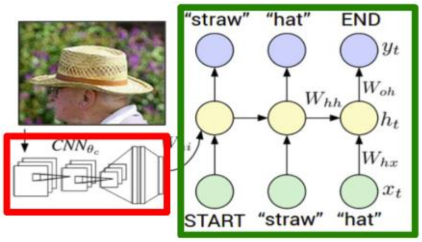


- With enough neurons and time, RNNs can compute anything that can be computed by your computer.
- Training is more challenging (unrolled networks are deep)
  - *See Machine Learning lecture for details…*

Slide credit: Geoff Hinton

Image source: Andrej Karpathy

Recap: Applications – Image Tagging



- Simple combination of CNN and RNN
  - Use CNN to define initial state $\mathbf{h}_0$ of an RNN.
  - Use RNN to produce text description of the image.

Computer Vision Summer'19

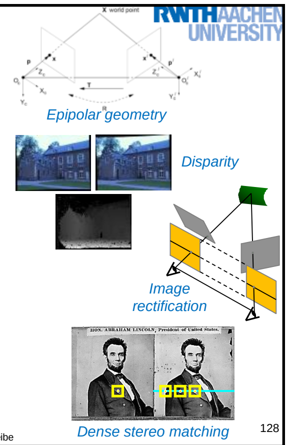
Slide adapted from Andrei Karpathy

B. Leibe

127

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



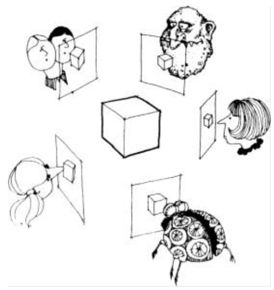
Computer Vision Summer'19

B. Leibe

128

Recap: What Is Stereo Vision?

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



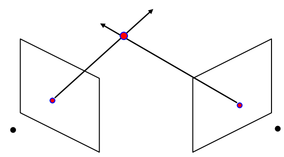
Computer Vision Summer'19

Slide credit: Svetlana Lazebnik, Steve Seitz

B. Leibe

129

Recap: Depth with Stereo – Basic Idea



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

Computer Vision Summer'19

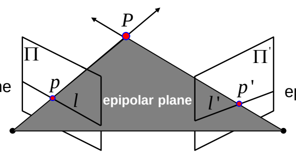
Slide credit: Steve Seitz

B. Leibe

130

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 Π must lie on the epipolar line l' in Π' (and vice versa).
  - Reduces correspondence problem to 1D search along conjugate epipolar lines.

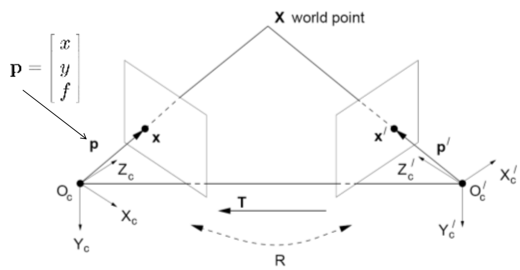
Computer Vision Summer'19

Slide adapted from Steve Seitz

B. Leibe

131

Recap: Stereo Geometry With Calibrated Cameras



- Camera-centered coordinate systems are related by known rotation R and translation T:

$$\mathbf{X}' = \mathbf{R}\mathbf{X} + \mathbf{T}$$

Computer Vision Summer'19

Slide credit: Kristen Grauman

B. Leibe

132

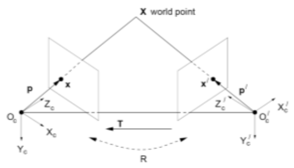
Recap: Essential Matrix

$$\mathbf{X}' \cdot (\mathbf{T} \times \mathbf{R}\mathbf{X}) = 0$$

$$\mathbf{X}' \cdot (\mathbf{T}_x \mathbf{R}\mathbf{X}) = 0$$

Let $\mathbf{E} = \mathbf{T}_x\mathbf{R}$

$$\mathbf{X}'^T \mathbf{E} \mathbf{X} = 0$$



- This holds for the rays p and p' that are parallel to the camera-centered position vectors X and X', so we have: $\mathbf{p}'^T \mathbf{E}\mathbf{p} = 0$
- E is called the essential matrix, which relates corresponding image points [Longuet-Higgins 1981]

Slide credit: Kristen Grauman   B. Leibe   133

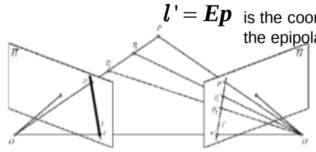
Computer Vision Summer'19

Recap: Essential Matrix and Epipolar Lines

$\mathbf{p}'^T\mathbf{E}\mathbf{p} = 0$ Epipolar constraint: if we observe point p in one image, then its position p' in second image must satisfy this equation.

$l' = Ep$ is the coordinate vector representing the epipolar line for point p

(i.e., the line is given by: $l'^T \mathbf{x} = 0$)



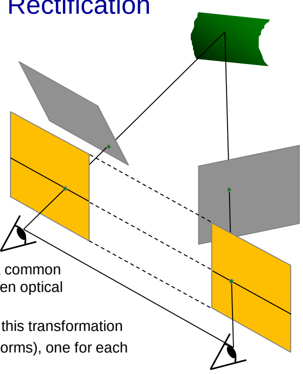
$l = E^T p'$ is the coordinate vector representing the epipolar line for point p'

Slide credit: Kristen Grauman   B. Leibe   134

Computer Vision Summer'19

Recap: Stereo Image Rectification

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

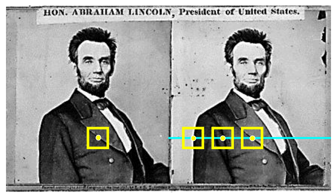


- Algorithm
  - Reproject image planes onto a common plane parallel to the line between optical centers
  - Pixel motion is horizontal after this transformation
  - Two homographies (3x3 transforms), one for each input image reprojection

Slide adapted from Li Zhang   C. Loop & Z. Zhang, Computing Rectifying Homographies for Stereo Vision, CVPR'99   135

Computer Vision Summer'19

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

Adapted from Svetlana Lazebnik, Li Zhang   136

Computer Vision Summer'19

Recap: Effect of Window Size

W = 3   W = 20

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

Slide credit: Kristen Grauman   B. Leibe   Figures from Li Zhang   137

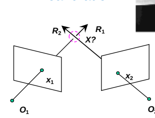
Computer Vision Summer'19

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

$$K = \begin{bmatrix} \alpha_x & s & x_0 \\ & \alpha_y & y_0 \\ & & 1 \end{bmatrix}$$ Camera models

Camera calibration



Triangulation

Essential matrix, Fundamental matrix

$$x^T E x' = 0$$

$$x^T F x' = 0$$

Eight-point algorithm

SVD!

B. Leibe   138

Computer Vision Summer'19

Recap: A General Point

- Equations of the form

$$Ax = 0$$

- How do we solve them? (always!)
  - Apply SVD

$$\underset{\text{SVD}}{A = UDV^T} = U \begin{bmatrix} d_{11} & & \\ & \ddots & \\ & & d_{NN} \end{bmatrix} \begin{bmatrix} v_{11} & \cdots & v_{1N} \\ \vdots & \ddots & \vdots \\ v_{N1} & \cdots & v_{NN} \end{bmatrix}^T$$

Singular values   Singular vectors

  - Singular values of A = square roots of the eigenvalues of A^TA.
  - The solution of Ax=0 is the *nullspace* vector of A.
  - This corresponds to the *smallest singular vector* of A.

Recap: Camera Parameters

- Intrinsic parameters
  - Principal point coordinates
  - Focal length
  - Pixel magnification factors
  - *Skew (non-rectangular pixels)*
  - *Radial distortion*

$$K = \begin{bmatrix} m_x & & \\ & m_y & \\ & & 1 \end{bmatrix} \begin{bmatrix} f & s & p_x \\ & f & p_y \\ & & 1 \end{bmatrix} = \begin{bmatrix} \alpha_x & s' & x_0 \\ & \alpha_y & y_0 \\ & & 1 \end{bmatrix}$$

- Extrinsic parameters
  - Rotation R
  - Translation t
    (both relative to world coordinate system)
- Camera projection matrix $\quad P = K[R \mid t]$
  - ⇒ General pinhole camera: 9 DoF
  - ⇒ CCD Camera with square pixels: 10 DoF
  - ⇒ General camera: 11 DoF

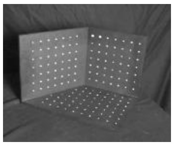
Recap: Calibrating a Camera

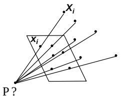
Goal

- 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 $P=P_{int}P_{ext}$





Slide credit: Kristen Grauman

Recap: Camera Calibration (DLT Algorithm)

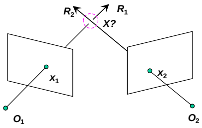
$$\begin{bmatrix} 0^T & X_1^T & -y_1X_1^T \\ X_1^T & 0^T & -x_1X_1^T \\ \cdots & \cdots & \cdots \\ 0^T & X_n^T & -y_nX_n^T \\ X_n^T & 0^T & -x_nX_n^T \end{bmatrix} \begin{pmatrix} P_1 \\ P_2 \\ P_3 \end{pmatrix} = 0 \qquad Ap = 0$$

- 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.

Slide adapted from Svetlana Lazebnik

Recap: Triangulation – Lin. Alg. Approach

see Exercise 6.3!

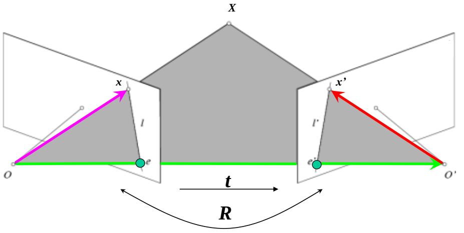


$$\lambda_1 x_1 = P_1X \qquad x_1 \times P_1X = 0 \qquad [x_{1\times}]P_1X = 0$$
$$\lambda_2 x_2 = P_2X \qquad x_2 \times P_2X = 0 \qquad [x_{2\times}]P_2X = 0$$

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

Slide credit: Svetlana Lazebnik

Recap: Epipolar Geometry – Calibrated Case



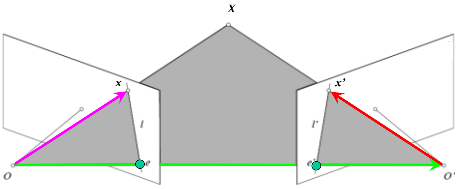
Camera matrix: $[I|0]$
$X = (u, v, w, 1)^T$
$x = (u, v, w)^T$

Camera matrix: $[R^T \mid -R^Tt]$
Vector *x'* in second coord. system has coordinates *Rx'* in the first one.

The vectors *x*, *t*, and *Rx'* are coplanar

Slide credit: Svetlana Lazebnik

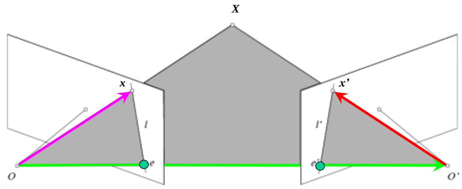
Recap: Epipolar Geometry – Calibrated Case



$x \cdot [t \times (Rx')] = 0 \Rightarrow x^T E x' = 0 \quad \text{with} \quad E = [t_\times]R$

Essential Matrix
(Longuet-Higgins, 1981)

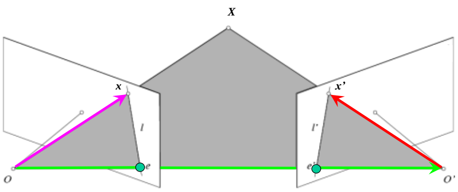
Recap: Epipolar Geometry – Calibrated Case



$x \cdot [t \times (Rx')] = 0 \Rightarrow x^T E x' = 0 \quad \text{with} \quad E = [t_\times]R$

- $E x'$ is the epipolar line associated with x' ($l = E x'$)
- $E^T x$ is the epipolar line associated with x ($l' = E^T x$)
- $E e' = 0$ and $E^T e = 0$
- E is singular (rank two)
- E has five degrees of freedom (up to scale)

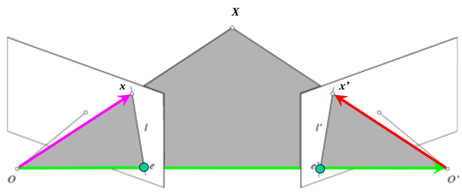
Recap: Epipolar Geometry – Uncalibrated Case



- The calibration matrices K and K' of the two cameras are unknown
- We can write the epipolar constraint in terms of *unknown* normalized coordinates:

$\hat{x}^T E \hat{x}' = 0 \qquad x = K\hat{x}, \quad x' = K'\hat{x}'$

Recap: Epipolar Geometry – Uncalibrated Case



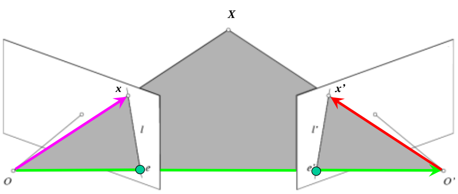
$\hat{x}^T E \hat{x}' = 0 \Rightarrow x^T F x' = 0 \quad \text{with} \quad F = K^{-T} E K'^{-1}$

$x = K\hat{x}$

$x' = K'\hat{x}'$

Fundamental Matrix
(Faugeras and Luong, 1992)

Recap: Epipolar Geometry – Uncalibrated Case



$\hat{x}^T E \hat{x}' = 0 \Rightarrow x^T F x' = 0 \quad \text{with} \quad F = K^{-T} E K'^{-1}$

- $F x'$ is the epipolar line associated with x' ($l = F x'$)
- $F^T x$ is the epipolar line associated with x ($l' = F^T x$)
- $F e' = 0$ and $F^T e = 0$
- F is singular (rank two)
- F has seven degrees of freedom

Recap: The Eight-Point Algorithm

see Exercise 6.1!

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

$$(u, v, 1)\begin{pmatrix} F_{11} & F_{12} & F_{13} \\ F_{21} & F_{22} & F_{23} \\ F_{31} & F_{32} & F_{33} \end{pmatrix}\begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} = 0 \Rightarrow (uu', uv', u, vu', vv', v, u', v', 1)\begin{pmatrix} F_{11} \\ F_{12} \\ F_{13} \\ F_{21} \\ F_{22} \\ F_{23} \\ F_{31} \\ F_{32} \\ F_{33} \end{pmatrix} = 0$$

$$\begin{bmatrix} u_1u_1' & u_1v_1' & u_1 & v_1u_1' & v_1v_1' & v_1 & u_1' & v_1' & 1 \\ u_2u_2' & u_2v_2' & u_2 & v_2u_2' & v_2v_2' & v_2 & u_2' & v_2' & 1 \\ u_3u_3' & u_3v_3' & u_3 & v_3u_3' & v_3v_3' & v_3 & u_3' & v_3' & 1 \\ u_4u_4' & u_4v_4' & u_4 & v_4u_4' & v_4v_4' & v_4 & u_4' & v_4' & 1 \\ u_5u_5' & u_5v_5' & u_5 & v_5u_5' & v_5v_5' & v_5 & u_5' & v_5' & 1 \\ u_6u_6' & u_6v_6' & u_6 & v_6u_6' & v_6v_6' & v_6 & u_6' & v_6' & 1 \\ u_7u_7' & u_7v_7' & u_7 & v_7u_7' & v_7v_7' & v_7 & u_7' & v_7' & 1 \\ u_8u_8' & u_8v_8' & u_8 & v_8u_8' & v_8v_8' & v_8 & u_8' & v_8' & 1 \end{bmatrix}\begin{bmatrix} F_{11} \\ F_{12} \\ F_{13} \\ F_{21} \\ F_{22} \\ F_{23} \\ F_{31} \\ F_{32} \\ F_{33} \end{bmatrix} = 0$$

1.) Solve with SVD. This minimizes

$$\sum_{i=1}^{N} (x_i^T F x_i')^2$$

2.) Enfore rank-2 constraint using SVD

- Problem: poor numerical conditioning

Recap: Normalized Eight-Point Alg.

see Exercise 6.1!

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.

$$F \overset{\text{SVD}}{=} \boldsymbol{UDV}^T = \boldsymbol{U}\begin{bmatrix} d_{11} & & \\ & d_{22} & \\ & & d_{33}\end{bmatrix}\begin{bmatrix} v_{11} & \cdots & v_{13} \\ \vdots & \ddots & \vdots \\ v_{31} & \cdots & v_{33}\end{bmatrix}^T$$

Set d_{33} to zero and reconstruct F

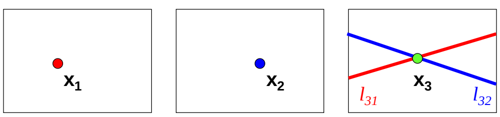
4. Transform fundamental matrix back to original units: if T and T' are the normalizing transformations in the two images, than the fundamental matrix in original coordinates is $T^T F T'$.

Recap: Comparison of Estimation Algorithms

	8-point	Normalized 8-point	Nonlinear least squares
Av. Dist. 1	2.33 pixels	0.92 pixel	0.86 pixel
Av. Dist. 2	2.18 pixels	0.85 pixel	0.80 pixel

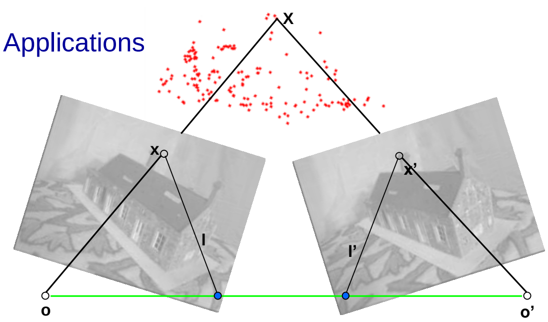
Recap: Epipolar Transfer

- 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?



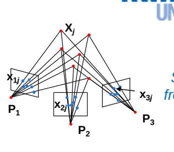
$$l_{31} = F^T{}_{13}\, x_1$$
$$l_{32} = F^T{}_{23}\, x_2$$

Applications



Repetition

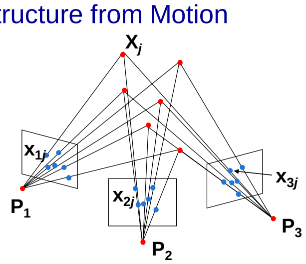
- Image Processing Basics
- Segmentation & Grouping
- Object Recognition
- Local Features & Matching
- Object Categorization
- 3D Reconstruction
  - Epipolar Geometry and Stereo Basics
  - Camera Calibration & Uncalibrated Reconstruction
  - Structure-from-Motion



Structure-from-Motion

Projective ambiguity

Recap: Structure from Motion



- Given: *m* images of *n* fixed 3D points

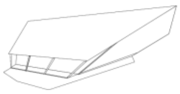
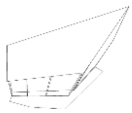
$$\mathrm{x}_{ij} = \mathrm{P}_i \mathrm{X}_j, \quad i = 1, \dots, m, \quad j = 1, \dots, n$$

- Problem: estimate *m* projection matrices P_i and *n* 3D points X_j from the *mn* correspondences x_{ij}

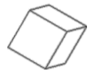
Recap: Structure from Motion Ambiguity

- 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

$$\mathbf{x} = \mathbf{P}\mathbf{X} = (\mathbf{P}\mathbf{Q}^{-1})\mathbf{Q}\mathbf{X}$$

Recap: Hierarchy of 3D Transformations

Projective 15dof	$\begin{bmatrix} A & t \\ v^T & v \end{bmatrix}$		Preserves intersection and tangency
Affine 12dof	$\begin{bmatrix} A & t \\ 0^T & 1 \end{bmatrix}$		Preserves parallellism, volume ratios
Similarity 7dof	$\begin{bmatrix} sR & t \\ 0^T & 1 \end{bmatrix}$		Preserves angles, ratios of length
Euclidean 6dof	$\begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix}$		Preserves angles, lengths

- 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!