Class 3: Low-level Representation

Liangliang Cao, Feb 6, 2014
EECS 6890 – Topics in Information Processing
Spring 2014, Columbia University
The Power of Good Feature Detector

Let’s see a few demos...
Photo Tourism: the power of SIFT

Photo Tourism
Exploring photo collections in 3D

Noah Snavely  Steven M. Seitz  Richard Szeliski
University of Washington  Microsoft Research

SIGGRAPH 2006
Photo Tourism

Detect SIFT
Photo Tourism

Detect SIFT
Photo Tourism

Match SIFT
Photo Tourism

From Correspondence To Geometry

minimize $f(R, T, P)$
Demo: The Power of Harr Filter

- **FaceDetection with 15 Lines of Code**

' Using Accord.net, in VB:

```vbnet
Dim detector As HaarObjectDetector
Dim cascade As New FaceHaarCascade
detector = New HaarObjectDetector(cascade, 30)
detector.SearchMode = ObjectDetectorSearchMode.Average
detector.ScalingFactor = 1.5

Dim sw As Stopwatch = Stopwatch.StartNew
Dim faceObjects As Rectangle() = detector.ProcessFrame(PictureBox1.Image)
sw.Stop()

Dim g As Graphics = Graphics.FromImage(PictureBox1.Image)
For Each face In faceObjects
    g.DrawRectangle(Pens.DeepSkyBlue, face)
Next
g.Dispose()
```
Demo: The Power of Harr Filter

- **FaceDetection with 15 Lines of Code**

/*Using OpenCV, in C/C++*/

#include "opencv2/core.hpp"
#include "opencv2/contrib.hpp"
#include "opencv2/highgui.hpp"
#include "opencv2/imgproc.hpp"
#include "opencv2/objdetect.hpp"
using namespace cv;

Mat gray;
cvtColor(original, gray, CV_BGR2GRAY);
vector< Rect_<int> > faces;
haar_cascade.detectMultiScale(gray, faces);
for(int i = 0; i < faces.size(); i++) {
    Rect face_i = faces[i];
    rectangle(original, face_i, CV_RGB(0, 255, 0), 1);
}
imshow("face_recognizer", original);
## Outline

1. **Traditional features**
   - Raw Pixels and Histograms
   - Image Filters

2. **Classic local feature: SIFT**
   - SIFT Descriptor
   - SIFT Detector

3. **More recent local features**
   - SURF
   - Binary Local Features

---

### Hands on Instruction
Outline

Traditional features
- Raw Pixels and Histograms
- Image Filters

Classic local feature: SIFT
- SIFT Descriptor
- SIFT Detector

More recent local features
- SURF
- Binary Local Features

Hands on Instruction
Traditional Low Level Features

Raw Pixels and Histograms
Raw Pixels and Histograms

Concatenating Raw Pixels As 1D Vector

Credit: The Face Research Lab
Raw Pixels and Histograms

Concatenated Raw Pixels

Famous applications (widely used in ML field)

• Face recognition

• Hand written digits

*Pictures courtesy to Sam Roweis*
Raw Pixels and Histograms

Tiny Images

Antonio Torralba et al proposed to resize images to 32x32 color thumbnails, which are called “tiny images”

Related applications

– Scene recognition
– Object recognition

Fast speed with limited accuracy

Torralba et al, MIT CSAIL report, 2007
Raw Pixels and Histograms

Problem of raw-pixel based representation
  – Rely heavily on good alignment
  – Assume the images are of similar scale
  – Suffer from occlusion
  – Recognition from different view point will be difficult

We want more powerful features for real-world problems like the following
Raw Pixels and Histograms

Color Histogram

Each pixel is described by a vector of pixel values

\[
\begin{pmatrix}
    r \\
    g \\
    b
\end{pmatrix}
\]

Distribution of color vectors is described by a histogram

Note: There are different choices for color space: RGB, HSV, Lab, etc. For gray images, we usually use 256 or fewer bins for histogram.

[Swain and Ballard 91]
Raw Pixels and Histograms

Benefits of Histogram Representation

No longer sensitive to alignment, scale transform, or even global rotation.

Similar color histograms (after normalization).
Raw Pixels and Histograms

Limitation of Global Histogram

Global histogram has no location information at all

They’re equal in terms of global histogram

Example courtesy to Erik Learned-Miller
Raw Pixels and Histograms

Histogram with Spatial Layout

Concatenated histogram for each region.

Example courtesy to Erik Learned-Miller
IBM IMARS Spatial Gridding

Raw Pixels and Histograms
Raw Pixels and Histograms

Spatial Pyramid Matching

Lazebnik, Schmid and Ponce, CVPR’06
Raw Pixels and Histograms

General Histogram Representation

Color histogram

patch
patch
patch

Extract color descriptor

histogram accumulation

General histogram

patch
patch
patch

Extract other descriptors
- edge histogram
- shape context histogram
- local binary patterns

histogram accumulation
• For each pixel $p$, create an 8-bit number $b_1 b_2 b_3 b_4 b_5 b_6 b_7 b_8$, where $b_i = 0$ if neighbor $i$ has value less than or equal to $p$’s value and 1 otherwise.

• Represent the texture in the image (or a region) by the histogram of these numbers.

Ojala et al, PR’96, PAMI’02
Raw Pixels and Histograms

LBP Histogram

• Divide the examined window to cells (e.g. 16x16 pixels for each cell).
• Compute the histogram, over the cell, of the frequency of each "number" occurring.
• Optionally normalize the histogram.
• Concatenate normalized histograms of all cells.
Image Filters
Image Filters

Haar-like Feature

Haar Wavelet

Haar like features

Given two adjacent rectangular regions, sums up the pixel intensities in each region and calculates the difference between the two sums.

Application: face detection [Viola and Jones]:

Efficiently processing via integral images
Image Filters

Integral Image

In integral image, each location saves the sum of gray scale pixel value of the sub-image to the left top corner.

\[ I_f(x) = \sum \sum I(i, j) \]

\[ S = A - B - C + D \]

Cost four additions operation only
Box filters And Harr Filters

Approximated second order derivatives with box filters (mean/average filter)

Both box filters and following Haar wavelets can be computed with integral images.
**Image Filters**

**Gabor Filtering**

- Gabor filters are defined by a harmonic function multiplied by a Gaussian function.

\[ g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = \exp \left( -\frac{x^2 + \gamma^2 y^2}{2\sigma^2} \right) \cos \left( 2\pi \frac{x'}{\lambda} + \psi \right) \]

- They reflect edges at different scales and spatial frequencies

- J. G. Daugman discovered that simple cells in the visual cortex can be modeled by Gabor functions

*Widely used in fingerprint, iris, OCR, texture and face recognition.*

*Computationally expensive.*
Development of Low Level Features

Classical features
- Raw pixel
- Histogram feature
  - Color Histogram
  - Edge histogram
- Frequency analysis
- Image filters
- Texture features
  - LBP
- Scene features
  - GIST
- Shape descriptors
- Edge detection
- Corner detection

1999

Local Descriptors
- SIFT
- HOG
- SURF
- DAISY
- BRIEF
- ...
- DoG
- Hessian detector
- Laplacian of Harris
- FAST
- ORB
- ...

Visual Recognition And Search 31 Columbia University, Spring 2014
Scale-Invariant Feature Transform (SIFT)

David G. Lowe
- Distinctive image features from scale-invariant keypoints, IJCV 2004
- Object recognition from local scale-invariant features, ICCV 1999
Why Local Features?

- Local Feature = Interest “Point” = Keypoint = Feature “Point” = Distinguished Region = Covariant Region

Local : robust to occlusion/clutter

Invariant : to scale/view/illumination changes

Local feature can be discriminant!
Why Local Features? (2)
How to Get Good Local Features

**Good descriptor**

**Distinctiveness:** is able to distinguish any objects

**Invariance:** find the same object across different viewpoint/illumination change

**Good detector**

**Locality:** features are local, so robust to occlusion and clutter

**Quantity:** many features can be generated for even small objects
Distinctive Descriptor

Histogram of gradient orientation
- Histogram is more robust to position than raw pixels
- Edge gradient is more distinctive than color for local patches

Concatenate histograms in spatial cells for better discriminability

Histogram of gradient orientation → SIFT Descriptor → Concatenated histogram

[D. Lowe, 1999]
• Compute image gradients at each location
• Histogram of the gradient orientation (8 bins)
• Concatenate histograms of 4 x 4 spatial grid
• Feature Dimension:
  8 orientation x (4x4) spatial grid = 128
David Lowe’s excellent performance tuning:

- Good parameters: 4 ori, 4 x 4 grid
- Soft-assignment to spatial bins
- Gaussian weighting over spatial location
**SIFT**

**More Invariant to Illumination Changes**

- Normalize the descriptor to norm one
  - A change in image contrast can change the magnitude but not the orientation

- Reduce the influence of large gradient magnitudes
  - Threshold the values in the unit feature vector to be no larger than 0.2
  - Renormalize the feature vector after thresholding.
Rotation Invariance

• Estimate the dominant orientation of each patch
• Rotate patch in dominant direction
• Each patch is associated with (X, Y, Scale, Orientation)

Note:
- Not every package provides such a procedure in SIFT implementation
- Few existing work makes good use of the dominant orientation
- With rotation invariance, SIFT may be less discriminant in general recognition
How to Get Good Local Features

**Good descriptor**

**Distinctiveness:** is able to distinguish any objects

**Invariance:** find the same object across different viewpoint/illumination change

*Can we do better?*
How to Get Good Local Features

**Good descriptor**

**Distinctiveness:** is able to distinguish any objects

**Invariance:** find the same object across different viewpoint/illumination change

**Can we do better?**

- More computational efficient, esp for mobile application
- More compact storage
- Combining SIFT with traditional features, e.g., LBP
- Improving poor color description:
  - SIFT is robust to illumination change, but color is not
  - SIFT in individual color channels (RGB): more expensive but slightly better
How to Get Good Local Features

**Good descriptor**

*Distinctiveness:* is able to distinguish any objects
*Invariance:* find the same object across different viewpoint/illumination change

**Good detector**

*Locality:* features are local, so robust to occlusion and clutter
*Quantity:* many features can be generated for even small objects
SIFT Detector: A First Impression

SIFT employs

- A variant of Histogram of Gradient Orientation as descriptor
- DoG (Difference of Gaussians) as local region detector

Why?

- DoG has good theoretical support but may not necessarily reflect the requirements in practice
- Trade-off between invariance and number of detection:
  - In many scenarios, combining multiple detectors -> good performance.
Concerns for Detectors

• Detecting Corners
  – Corners are located more robustly than edges

• Selecting Scales
  – Invariant to scale changes

• Finding Affine Covariance
  – Invariant to affine transform; matching ellipse with circles
  – Only useful when there is such transform in your dataset!
SIFT Detector

DoG Detector

- Detecting Corners ✔️
- Selecting Scales ✔️
- Finding Affine Covariance ✗

Figure courtesy to Matas and Mikolajczyk
SIFT Detector

Difference of Gaussians
SIFT Detector

Sampling Frequency in Scale

More points are found as sampling frequency increases, but accuracy of matching decreases after 3 scales/octave
DoG and Scale Space Theory

- Scale space representation is a family of the convolution of the given signal with Gaussian kernel

\[ L(x, y, \sigma) = G(x, y, \sigma) \ast I(x, y) \]
\[ G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2 + y^2}{2\sigma^2}} \]
DoG and Scale Space Theory (2)

Let \( t = \sigma \) which represents a "temperature" of the intensity values of the image as "temperature distribution" in the image plane. Then the scale-space family can be defined as the solution of the diffusion equation

\[
\partial_t L = \frac{1}{2} \nabla^2 L,
\]

where the Gaussian kernel arises as the Green's function of this specific partial differential equation.

In scale-space, the corner can be obtained by Laplacian of Gaussian

\[
\sigma^2 \nabla^2 G \sim G(x, y, k\sigma) - G(x, y, \sigma)
\]

where DoG is an efficient approximation of Laplacian of Gaussian.
Accurate Key point localization

- Detect maxima and minima of difference-of-Gaussian in scale space
- Fit a quadratic to surrounding values for sub-pixel and sub-scale interpolation (Brown & Lowe, 2002).
- Taylor expansion around point:

\[ D(x) = D + \frac{\partial D^T}{\partial x} x + \frac{1}{2} x^T \frac{\partial^2 D}{\partial x^2} x \]

- Offset of extremum (use finite differences for derivatives):

\[ \hat{x} = -\frac{\partial^2 D^{-1}}{\partial x^2} \frac{\partial D}{\partial x} \]
Eliminate Unstable Detections

To get stable detection, David Lowe rejects
1. Keypoints with low contrast
2. Keypoints with strong edge but not clear corners.

For 1, we filter out keypoints with small
\[ D = L(x, y, k\sigma) - L(x, y, \sigma). \]

For 2, those keypoints correspond to large principle curvature along the edge but small one in the perpendicular direction. To efficient compute the curvature ratio, we need only compute the trace and determinant of local hessian.
SIFT Detector

Example of SIFT Keypoint Detection

(a) 233x189 image
(b) 832 DOG extrema
(c) 729 left after peak value threshold
(d) 536 left after testing ratio of principle curvatures
How to Do Better than DoG?

- Combine more than one detectors
  - Harris detector
  - Hessian detector

- Dense-sampling instead of sparse detector
  - Grid based sampling + multiple resolutions
  - More computation, more storage, but better accuracy
SIFT Detector

Harris Detector

\[ \mu(\sigma_I, \sigma_D) = g(\sigma_I)^* \begin{bmatrix} I_x^2(\sigma_D) & I_x I_y(\sigma_D) \\ I_x I_y(\sigma_D) & I_y^2(\sigma_D) \end{bmatrix} \]

1. Image derivatives

2. Square of derivatives

3. Gaussian filter \( g(\sigma_I) \)

Cornerness function – both eigenvalues are strong

\[ har = \det[\mu(\sigma_I, \sigma_D)] - \alpha[\text{trace}(\mu(\sigma_I, \sigma_D))] = g(I_x^2)g(I_y^2) - [g(I_x I_y)]^2 - \alpha[g(I_x^2) + g(I_y^2)]^2 \]

Scale selection: Laplacian of Harris detector
SIFT Detector

Hessian Detector

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:

$$I_{xx} \ast I_{yy} - (I_{xy})^2$$
More Local Features Following SIFT

**SURF** (ECCV’06)
- Gradient based on Integral image.
- 3-5 times faster than SIFT, though not as robust

**DAISY** (CVPR’09)
- Binary descriptor (13 bytes)
- Steerable filters + quantization + PCA

**BRIEF** (ECCV’10)
- Binary descriptor (256 bit), super efficient and compact

**ORB** (ICCV’11)
- FAST detector/BRIEF detector
- 100x faster than SIFT

**BRISK** (ICCV’11)
- SIFT-like scale-space detection + BRIEF-like descriptor

*Volunteer needed to present in next class!*
Hands-on Experience

Most of slides in this section are courtesy to Andrea Vedaldi
Existing softwares

• VLFeat
  – http://www.vlfeat.org/

• OpenCV
  – http://opencv.org/

• Amsterdam Color Descriptor
  – http://koen.me/research/colordescriptors/

• Oxford VGG
  – http://www.robots.ox.ac.uk/~vgg/research/affine/
Existing softwares

- VLFeat
  - http://www.vlfeat.org/
- OpenCV
  - http://opencv.org/
- Amsterdam Color Descriptor
  - http://koen.me/research/colordescriptors/
- Oxford VGG
  - http://www.robots.ox.ac.uk/~vgg/research/affine/
VLFeat

SIFT Detector and Descriptor

1. load an image
   ```matlab
   imPath = fullfile('oxford.jpg')
   im = imread(imPath)
   ```

2. convert it to single precision gray scale
   ```matlab
   imgs = im2single(rgb2gray(im))
   ```

3. run SIFT
   ```matlab
   [frames, descrs] = vl_sift(imgs)
   ```

4. visualise keypoints
   ```matlab
   imagesc(im); colormap gray; hold on;
   vl_plotframe(frames)
   ```

5. visualise descriptors
   ```matlab
   vl_plotsiftdescriptor(...
   descrs(:,432), frames(:,432))
   ```
What Does “Frame” Mean?

frames has a feature frame for each column containing a stacked affine transformation (A,T)

\[
\text{frames}(::,i) = \begin{bmatrix}
    t_1 \\
    t_2 \\
    a_{11} \\
    a_{21} \\
    a_{12} \\
    a_{22}
\end{bmatrix}
\]

\[
\begin{bmatrix}
    x \\
    y
\end{bmatrix} = \begin{bmatrix}
    a_{11} & a_{12} \\
    a_{21} & a_{22}
\end{bmatrix} \begin{bmatrix}
    \hat{x} \\
    \hat{y}
\end{bmatrix} + \begin{bmatrix}
    t_1 \\
    t_2
\end{bmatrix}
\]
Other Corner Detectors

```matlab
frames = vl_covdet(imgs, 'method', 'Hessian');
```

- Difference of Gaussian (Laplacian, trace of Hessian, or SIFT)
- Hessian (determinant of)
Other Corner Detectors

frames = \texttt{vl_covdet}(\texttt{imgs, 'method', 'HarrisLaplace'});  

- Harris Laplace (Harris cornerness, Laplacian for scale selection)
- Hessian Laplace
Dominant Orientation

frames = vl_covdet(imgs, 'EstimateOrientations', true) ;
Compute Descriptors

\[
\text{[frames, descrs] = vl_covdet(imgs, 'Descriptor', 'patch');}
\]

Each column of \texttt{descrs} is a stacked normalised image patch.
Custom Frames

frames = vl_covdet(imgs, ...  
'Frames', frames, ...  
'EstimateAffineShape', true, ...  
'EstimateOrientation', true) ;
OpenCV

List of Options

- **OpenCV 2.1:**
  
  Harris, Tomasi&Shi, MSER, DoG, SURF, STAR and FAST, DoG + SIFT, SURF, HARRIS, GoodFeaturesToTrack

- **OpenCV later version:**
  
  ORB, BRIEF, and FREAK
OpenCV

An Example (Extracting SIFT)

```cpp
#include <opencv2/opencv.hpp>
#include <opencv2/highgui.hpp>

int main(int argc, const char* argv[]) {
    const cv::Mat input = cv::imread("input.jpg", 0); // Load as grayscale

    cv::SiftFeatureDetector detector;
    std::vector<cv::KeyPoint> keypoints;
    detector.detect(input, keypoints);

    // Add results to image and save.
    cv::Mat output;
    cv::drawKeypoints(input, keypoints, output);
    cv::imwrite("sift_result.jpg", output);

    return 0;
}

Courtesy to Unapiedra
Next class:

How to Use SIFT for Object Classification

Do you like a class programming contest as today?

- interesting programming related to the class
- 2-person group randomly assigned
- 10-15 mins,
- winner will take the extra scores!