Computer Vision - Lecture 10

Sliding-Window based Object Detection

4.12.2008

Bastian Leibe
RWTH Aachen
http://www.umic.rwth-aachen.de/multimedia
leibe@umic.rwth-aachen.de

Recap: Subspace Methods

Recap: Obj. Detection by Distance TO Eigenspace

- Scan a window $\omega$ over the image and classify the window as object or non-object as follows:
  - Project window to subspace and reconstruct as earlier.
  - Compute the distance between $\omega$ and the reconstruction (reprojection error).
  - Local minima of distance over all image locations $\Rightarrow$ object locations
  - Repeat at different scales
  - Possibly normalize window intensity such that $|\omega|=1$.

Course Outline

- Image Processing Basics
- Segmentation & Grouping
- Recognition
  - Global Representations
  - Subspace Representations
- Object Categorization I
  - Sliding Window based Object Detection
- Local Features & Matching
- Object Categorization II
- Part based Approaches
- 3D Reconstruction
- Motion and Tracking

Recap: Obj. Identification by Distance IN Eigenspace

- Objects are represented as coordinates in an $n$-dim. eigenspace.
- Example:
  - 3D space with points representing individual objects or a manifold representing parametric eigenspace (e.g., orientation, pose, illumination).
- Estimate parameters by finding the NN in the eigenspace

Applications: Recognition, Pose Estimation

H. Murase and S. Nayar, Visual learning and recognition of 3-d objects from appearance, IJCV 1995
Applications: Visual Inspection

Recap: Eigenfaces

Recap: Restrictions of PCA

Recap: Linear Discriminant Analysis (LDA)

Recap: Fisherfaces

Recap: Robust Estimation of PCA Coeff.


Topics of This Lecture

- Object Categorization
  - Problem Definition
  - Challenges
- Sliding-Window based Object Detection
  - Detection via Classification
  - Global Representations
  - Classifier Construction
- Classification with Boosting
  - Adaboost
  - Viola-Jones Face Detection
- Classification with SVMs
  - Support Vector Machines
  - HOG Detector

Identification vs. Categorization

What Could Be Done With Recognition Algorithms?

There is a wide range of applications, including...

- Autonomous robots
- Navigation, driver safety
- Situated search
- Content-based retrieval and analysis for images and videos
- Medical image analysis

Challenges: Robustness

- Detection in Crowded Scenes
  - Learn object variability
  - Changes in appearance, scale, and articulation
  - Compensate for clutter, overlap, and occlusion
Topics of This Lecture

- **Object Categorization**
  - Problem Definition
  - Challenges
- **Sliding-Window based Object Detection**
  - Detection via Classification
  - Global Representations
  - Classifier Construction
- **Classification with Boosting**
  - AdaBoost
  - Viola Jones Face Detection
- **Classification with SVMs**
  - Support Vector Machines
  - HOG Detector

Detection via Classification: Main Idea

- **Basic component: a binary classifier**

Detection via Classification: Main Idea

- 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.

Detection via Classification: Main Idea

- Consider all subwindows in an image
  - Sample at multiple scales and positions (and orientations)
- Make a decision per window:
  - “Does this contain object category X or not?”

Feature extraction: Global Appearance

- Simple holistic descriptions of image content
  - Grayscale / color histogram
  - Vector of pixel intensities
**Eigenfaces: Global Appearance Description**

This can also be applied in a sliding-window framework...

- Generate low-dimensional representation of appearance with a linear subspace.
- Project new images to "face space".
- Recognition via nearest neighbors in face space.

**Feature Extraction: Global Appearance**

- Pixel-based representations sensitive to small shifts
- Color or grayscale-based appearance description can be sensitive to illumination and intra-class appearance variation

**Gradient-based Representations**

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

**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

**Gradient-based Representations: Histograms of Oriented Gradients (HoG)**

Map each grid cell in the input window to a histogram counting the gradients per orientation.

- ![Image of HoG](attachment:image.png)

**Classifier Construction**

- How to compute a decision for each subwindow?
### Classifier Construction: Many Choices

- Nearest neighbor
- Neural networks
- Support Vector Machines
- Boosting
- Conditional Random Fields

**Boosting**

- Build a strong classifier by combining number of “weak classifiers”, which need only be better than chance
- Sequential learning process: at each iteration, add a weak classifier
- Flexible to choice of weak learner
  - including fast simple classifiers that alone may be inaccurate

We’ll look at Freund & Schapire’s AdaBoost algorithm

- Easy to implement
- Base learning algorithm for Viola-Jones face detector


---

**AdaBoost: Intuition**

Consider a 2D feature space with positive and negative examples.

Each weak classifier splits the training examples with at least 50% accuracy.

Examples misclassified by a previous weak learner are given more emphasis at future rounds.

---

**AdaBoost: Intuition**

Final classifier is combination of the weak classifiers

---

**AdaBoost Algorithm**

Start with uniform weights on training examples

- For T rounds
  - Evaluate weighted error for each feature, pick best.
  - Re-weight the examples:
    - Incorrectly classified get more weight
    - Correctly classified get less weight

The final classifier is combination of the “weak” ones, weighted according to the error they had.
Perceptual and Sensory Augmented Computing

Faces: Terminology

- **Detection**: given an image, where is the face?
- **Recognition**: whose face is it?

Example: Face Detection

- Frontal faces are a good example of a class where global appearance models + a sliding window detection approach fit well:
  - Regular 2D structure
  - Center of face almost shaped like a “patch”/window

Now we’ll take AdaBoost and see how the Viola-Jones face detector works

Feature extraction

“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

Outputs of a window

Considering all possible filter parameters: position, scale, and type:

180,000+ possible features associated with each 24 x 24 window

Use AdaBoost both to select the informative features and to form the classifier

Example

AdaBoost for Feature+Classifier Selection

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

Resulting weak classifier:

\[ h_i(x) = \begin{cases} +1 & \text{if } f_i(x) > 0_i \\ -1 & \text{otherwise} \end{cases} \]

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

Large Library of Filters

Use AdaBoost both to select the informative features and to form the classifier
**AdaBoost Algorithm**

Start with uniform weights on training examples

- For \( T \) rounds:
  - Evaluate weighted error for each feature, pick best.
  - Re-weight the examples:
    - Incorrectly classified ⇒⇒⇒⇒ more weight
    - Correctly classified ⇒⇒⇒⇒ less weight
  - Final classifier is combination of the weak ones, weighted according to the error they had.

\[ \text{AdaBoost for Efficient Feature Selection} \]

- Image features ⇒ weak classifiers
- For each round of boosting:
  - Evaluate each rectangle filter on each example
  - Sort examples by filter values
  - Select best threshold for each filter (min error)
  - Sorted list can be quickly scanned for the optimal threshold
  - Select best filter/threshold combination
  - Weight on this features is a simple function of error rate
  - Reweight examples


**Efficiency Considerations**

- Even if the filters are fast to compute, each new image has a lot of possible windows to search.
- How to make the detection more efficient?

**Cascading Classifiers for Detection**

- For efficiency, apply less accurate but faster classifiers first to immediately discard windows that clearly appear to be negative; e.g.,
  - Filter for promising regions with an initial inexpensive classifier
  - Build a chain of classifiers, choosing cheap ones with low false negative rates early in the chain

[Fleuret & Geman, IJCV 2001]
[Rowley et al., PAMI 1998]
[Viola & Jones, CVPR 2001]

**Viola-Jones Face Detector: Summary**

- 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/](http://sourceforge.net/projects/opencvlibrary/)
Viola-Jones Face Detector: Results

- Performance
  - 384 by 288 pixel images detected at 15 fps on a conventional 700 MHz Intel Pentium III in 2001.
  - Training time = weeks

First two features selected

Detecting profile faces?
Detecting profile faces requires training separate detector with profile examples.
You Can Try It At Home...

- The Viola & Jones detector was a huge success
  - First real-time face detector available
  - Many derivative works and improvements

- C++ implementation available in OpenCV [Lienhart, 2002]
  - http://sourceforge.net/projects/opencvlibrary/
- Matlab wrappers for OpenCV code available, e.g. here


Example Application

Frontal faces detected and then tracked, character names inferred with alignment of script and subtitles.

Everingham, M., Sivic, J. and Zisserman, A.
"Hello! My name is... Buffy" - Automatic naming of characters in TV video, BMVC 2006.
http://www.robots.ox.ac.uk/~vgg/research/nface/index.html

Example Application: Faces in Photos

Classifier construction: many choices...

- Nearest neighbor
  - Shakhnarovich, Viola, Darrell 2003
  - Berg, Berg, Malik 2005...

- Neural networks
  - LeCun, Bottou, Bengio, Haffner 1998
  - Rowley, Baluja, Kanade 1998...

- Support Vector Machines
  - Guyon, Vapnik Hebele, Serre, Poggio, 2001...

- Boosting
  - Viola, Jones 2001
  - Torralba et al. 2004
  - Opelt et al. 2006...

- Conditional Random Fields
  - McCallum, Freitag, Pereira 2000
  - Kumar, Hebert 2003...

Linear Classifiers

Let \( w = \begin{bmatrix} a \\ c \end{bmatrix} \) \( x = \begin{bmatrix} x \\ y \end{bmatrix} \)

\[ ax + cy + b = 0 \]
**Lines in $\mathbb{R}^2$**

Let $w = \begin{bmatrix} a \\ c \\ b \end{bmatrix}$ and $x = \begin{bmatrix} x \\ y \end{bmatrix}$.

$$ax + cy + b = 0$$

$w \cdot x + b = 0$

**Linear Classifiers**

- Find linear function to separate positive and negative examples

$\begin{align*}
x, \text{ positive: } & x \cdot w + b \geq 0 \\
x, \text{ negative: } & x \cdot w + b < 0
\end{align*}$

Which line is best?

**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

**Finding the Maximum Margin Line**

- Solution: $w = \sum \alpha_i y_i x_i$

- Classification function:

$$f(x) = \text{sign}(w \cdot x + b)$$

- Notice that it relies on an inner product between the test point $x$ and the support vectors $x_i$.

- (Solving the optimization problem also involves computing the inner products $x_i \cdot x_j$ between all pairs of training points)

**Support Vector Machines**

- Want line that maximizes the margin.

For support vectors, $x_i \cdot w + b = \pm 1$

**Finding the Maximum Margin Line**

- Solution: $w = \sum \alpha_i y_i x_i$

- Classification function:

$$f(x) = \text{sign}(w \cdot x + b)$$

If $f(x) < 0$, classify as neg., if $f(x) > 0$, classify as pos.
Questions

- What if the features are not 2d?
- What if the data is not linearly separable?
- What if we have more than just two categories?

Non-Linear SVMs

- Datasets that are linearly separable with some noise work out great:
- But what are we going to do if the dataset is just too hard?
- How about... mapping data to a higher-dimensional space?

Another Example

- Non-separable by a hyperplane in 2D
- Separable by a surface in 3D
Non-Linear SVMs: Feature Spaces

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

\[ \Phi: x \rightarrow \Phi(x) \]

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

Nonlinear SVMs

- The kernel trick: instead of explicitly computing the lifting transformation \( \Phi(x) \), define a kernel function \( K \) such that

\[ K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j) \]

- This gives a nonlinear decision boundary in the original feature space:

\[ \sum \alpha_i y_i K(x_i, x) + b \]


Some Often-Used Kernel Functions

- Linear: \( K(x_i, x_j) = x_i^T x_j \)

- Polynomial of power \( p \): \( K(x_i, x_j) = (1 + x_i^T x_j)^p \)

- Gaussian (radial-basis function network):

\[ K(x_i, x_j) = \exp\left(-\frac{||x_i - x_j||^2}{2\sigma^2}\right) \]

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

Questions

- What if the features are not 2d?
  - Generalizes to d-dimensions - replace line with “hyperplane”

- What if the data is not linearly separable?
  - Non-linear SVMs with special kernels

- What if we have more than just two categories?

B. Leibe

Slide credit: Kristen Grauman

Multi-Class SVMs

- Achieve multi-class classifier by combining a number of binary classifiers

  - One vs. all
    - Training: learn an SVM for each class vs. the rest
    - Testing: apply each SVM to test example and assign to it the class of the SVM that returns the highest decision value

  - One vs. one
    - Training: learn an SVM for each pair of classes
    - Testing: each learned SVM “votes” for a class to assign to the test example

B. Leibe

Slide credit: Kristen Grauman

SVMs for Recognition

1. Define your representation for each example.
2. Select a kernel function.
3. Compute pairwise kernel values between labeled examples
4. Given this “kernel matrix” to SVM optimization software to identify support vectors & weights.
5. To classify a new example: compute kernel values between new input and support vectors, apply weights, check sign of output.
Pedestrian Detection

- Detecting upright, walking humans using sliding window’s appearance/texture; e.g.,

  - SVM with Haar wavelets [Papageorgiou & Poggio, ICCV 2000]
  - Space-time rectangle features [Viola, Jones & Snow, ICCV 2003]
  - SVM with HoGs [Dalal & Triggs, CVPR 2005]

- Example: Pedestrian Detection with HoG and SVMs

  - Map each grid cell in the input window to a histogram counting the gradients per orientation.
  - Train a linear SVM using training set of pedestrian vs. non-pedestrian windows.

  Code available: http://pascal.inrialpes.fr/soft/olt/

Pedestrian detection with HoGs & SVMs

- Navneet Dalal, Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR 2005

Summary: Sliding-Windows

- Pros
  - Simple detection protocol to implement
  - Good feature choices critical
  - Past successes for certain classes
  - Good detectors available (Viola & Jones, HOG, etc.)

- Cons/Limitations
  - High computational complexity
    - For example: 250,000 locations x 30 orientations x 4 scales = 30,000,000 evaluations!
    - This puts tight constraints on the classifiers we can use.
    - If training binary detectors independently, this means cost increases linearly with number of classes.
    - With so many windows, false positive rate better be low

Limitations (continued)

- Not all objects are “box” shaped

- Non-rigid, deformable objects not captured well with representations assuming a fixed 2D structure; or must assume fixed viewpoint

- Objects with less-regular textures not captured well with holistic appearance-based descriptions

Limitations (continued)
Limitations (continued)

- If considering windows in isolation, context is lost

Limitations (continued)

- In practice, often entails large, cropped training set (expensive)
- Requiring good match to a global appearance description can lead to sensitivity to partial occlusions

References and Further Reading

- Read the Viola-Jones paper

- Viola-Jones Face Detector
  - C++ implementation available in OpenCV [Lienhart, 2002]
    - http://sourceforge.net/projects/opencvlibrary/
  - Matlab wrappers for OpenCV code available, e.g. here

- HOG Detector
  - Code available: http://pascal.inrialpes.fr/soft/olt/