CAP 4453
Robot Vision
Dr. Gonzalo Vaca-Castaño
gonzalo.vacacastano@ucf.edu
Administrative details

• Issues submitting homework
Credits

• Some slides come directly from:
  • Kristen Grauman
  • A. Zisserman
  • Ross B. Girshick
Short Review from last class
Feature Descriptor
Histogram of Oriented Gradients (HOG)

• Revisiting histogram
Invariance to deformation

- Deformation can also move pixels around
- Again, instead of precise location of each pixel, only want to record rough location
- Divide patch into a grid of cells
- Record counts of each orientation in each cell: *orientation histograms*
Feature detection and description

• Harris corner detection gives:
  • Location of each detected corner
  • Orientation of the corner (given by $x_{\text{max}}$)
  • Scale of the corner (the image scale which gives the maximum response at this location)

• Want feature descriptor that is
  • Invariant to photometric transformations, translation, rotation, scaling
  • Discriminative
Summary of HOG computation

• Step 1: Extract a square window (called “block”) of some size around the pixel location of interest.
• Step 2: Divide block into a square grid of sub-blocks (called “cells”) (2x2 grid in our example, resulting in four cells).
• Step 3: Compute orientation histogram of each cell.
• Step 4: Concatenate the four histograms.
• Step 5: normalize \( v \) using one of the three options:
  - Option 1: Divide \( v \) by its Euclidean norm.
  - Option 2: Divide \( v \) by its L1 norm (the L1 norm is the sum of all absolute values of \( v \)).
  - Option 3:
    - Divide \( v \) by its Euclidean norm.
    - In the resulting vector, clip any value over 0.2
    - Then, renormalize the resulting vector by dividing again by its Euclidean norm
Histogram of Oriented Gradients (HOG)

- Parameters and design options:
  - Angles range from 0 to 180 or from 0 to 360 degrees?
    - In the Dalal & Triggs paper, a range of 0 to 180 degrees is used
  - Number of orientation bins.
    - Usually 9 bins, each bin covering 20 degrees.
  - Cell size.
    - Cells of size 8x8 pixels are often used.
  - Block size.
    - Blocks of size 2x2 cells (16x16 pixels) are often used.
  - Usually a HOG feature has 36 dimensions.
    - 4 cells * 9 orientation bins.
Histogram of Oriented Gradients (HOG)
SIFT descriptor

• Compute on local 16 x 16 window around detection.
• Rotate and scale window according to discovered orientation $\Theta$ and scale $\sigma$ (gain invariance).
• Compute gradients weighted by a Gaussian of variance half the window (for smooth falloff).
SIFT descriptor

• 4x4 array of gradient orientation histograms weighted by gradient magnitude.
• Bin into 8 orientations x 4x4 array = 128 dimensions.
SIFT Descriptor Extraction

Gradient magnitude and orientation

16x16 window

128 dimensional vector

8 bin ‘histogram’ - add magnitude amounts!

Utkarsh Sinha
Reduce effect of illumination

- 128-dim vector normalized to 1
- Threshold gradient magnitudes to avoid excessive influence of high gradients
  - After normalization, clamp gradients > 0.2
  - Renormalize
import numpy as np
import cv2
from matplotlib import pyplot as plt
from google.colab.patches import cv2_imshow

## Create SIFT object
sift = cv2.xfeatures2d.SIFT_create()

## Create flann matcher
FLANN_INDEX_KDTREE = 1  # bug: flann enums are missing
flann_params = dict(algorithm = FLANN_INDEX_KDTREE, trees = 5)
matcher = cv2.FlannBasedMatcher(flann_params, {})

## Detect and compute
img1 = cv2.imread('androis.png')
gray1 = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
kpts1, descs1 = sift.detectAndCompute(gray1, None)

## As up
img2 = cv2.imread('android_small.png')
gray2 = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)
kpts2, descs2 = sift.detectAndCompute(gray2, None)

## Ratio test
matches = matcher.knnMatch(descs1, descs2, 2)
matchesMask = [[0,0] for i in range(len(matches))
for i, (m1,m2) in enumerate(matches):
    if m1.distance < 0.7 * m2.distance:
        matchesMask[i] = [1,0]

        ## Notice: How to get the index
pt1 = kpts1[m1.queryIdx].pt
pt2 = kpts2[m1.trainIdx].pt
print(i,pt1,pt2)

if i % 10 ==0:
    ## Draw pairs in purple, to make sure the result is ok
    cv2.circle(img1, (int(pt1[0]),int(pt1[1])), 5, (255,0,255), -1)
    cv2.circle(img2, (int(pt2[0]),int(pt2[1])), 5, (255,0,255), -1)

## Draw match in blue, error in red
draw_params = dict(matchColor = (255, 0, 0), singlePointColor = (0,0,255), matchesMask = matchesMask, flags = 0)
res = cv2.drawMatchesKnn(img1,kpts1,img2,kpts2,matches,None,**draw_params)
cv2_imshow(res)
Robot Vision

13. Object detection I
Outline

• **Overview: What is Object detection?**
• Top methods for object detection
• Object detection with Sliding Window and Feature Extraction (HoG)
  • Sliding Window technique
  • HOG: Gradient based Features
  • Machine Learning
    • Support Vector Machine (SVM)
    • Non-Maxima Suppression (NMS)
• Implementation examples
• Deformable Part-based Model (DPM)
What is object detection

- Classification
- Classification + Localization
- Object Detection
- Instance Segmentation

Single object: CAT, CAT
Multiple objects: CAT, DOG, DUCK, CAT, DOG, DUCK

Object detection

- Multiple outputs
  - Bounding box
  - Label
  - Score
Detection Competitions

Pascal VOC
COCO
ImageNet ILSVRC

VOC: 20 classes

COCO: 200 classes

http://host.robots.ox.ac.uk/pascal/VOC/voc2012/index.html#introduction
Valid detection

- Groundtruth:
  - Bounding box
  - Label

- Possible detection
  - Bounding box
  - Label
  - score

\[
\text{score}_{\text{iou}} = \frac{\text{Intersected Area}}{\text{Union BB area}}
\]

Different criteria to declare detections:

- Pascal criteria
  \[\text{score}_{\text{iou}} > 0.5\]

- All of these:
  \[\text{score}_{\text{iou}} > 0.5\]
  \[\text{score}_{\text{iou}} > 0.55\]
  \[\text{score}_{\text{iou}} > 0.6\]
  \[\text{score}_{\text{iou}} > 0.65\]
  \[\text{score}_{\text{iou}} > 0.7\]
  \[\text{score}_{\text{iou}} > 0.75\]
  \[\text{score}_{\text{iou}} > 0.8\]
  \[\text{score}_{\text{iou}} > 0.9\]
  \[\text{score}_{\text{iou}} > 0.95\]
Terms

Recall
Precision
mAP
IoU
Terms

Recall
Precision
mAP
IoU

Possible detection
Bounding box
Label

score

Average precision (AP): Area under curve
Terms

Recall
Precision
mAP
IoU

mAP is simply all the AP values averaged over different classes/categories.

Box Average Precision (AP@[0.5:0.95]): sums IOUs between 0.5 and 0.95 and divides the sum by the number of the IOU values.
Outline

• Overview: What is Object detection?
• Top methods for object detection
  • Object detection with Sliding Window and Feature Extraction (HoG)
    • Sliding Window technique
    • HOG: Gradient based Features
    • Machine Learning
      • Support Vector Machine (SVM)
      • Non-Maxima Suppression (NMS)
  • Implementation examples
  • Deformable Part-based Model (DPM)
Popular algorithms for object detection

• Pre-DeepLearning
  • HOG + SVM (Dalal, Triggs)
  • Deformable Part-based Model (DPM)

• Deep learning
  • Fast R-CNN
  • Faster R-CNN
  • Region-based Convolutional Neural Networks (R-CNN)
  • Region-based Fully Convolutional Network
  • Single Shot Detector (SSD)
  • YOLO (You Only Look Once)
PASCAL VOC 2005-2012

20 object classes

Classification: person, motorcycle
Detection
Person
Motorcycle

Action: riding bicycle

22,591 images

Segmentation

Object detection progress
PASCAL VOC

Before CNNs

Using CNNs

Source: R. Girshick
Large Scale Visual Recognition Challenge (ILSVRC) 2010-2014

20 object classes
22,591 images

200 object classes
517,840 images

1000 object classes
1,431,167 images

DET

CLS-LOC

http://image-net.org/challenges/LSVRC/
ILSVRC detection in 2014 (Deep learning)

1.9x increase in object detection average precision in one year

~3% due to more data
~18% due to better methods

Microsoft COCO: Common Objects in Context

COCO - Common Objects in Context (cocodataset.org)
State of the art methods
State of the art methods

Do you still need the old methods?

<table>
<thead>
<tr>
<th>Network model name</th>
<th>box AP</th>
<th>AP75</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSD512 [33]</td>
<td>28.8%</td>
<td>30.3%</td>
</tr>
<tr>
<td>RefineDet512(VGG-16) [62]</td>
<td>33.0%</td>
<td>35.5%</td>
</tr>
<tr>
<td>YOLO-v4-608 [63]</td>
<td>43.5%</td>
<td>47.0%</td>
</tr>
<tr>
<td>Faster R-CNN(LIP-ResNet-101-MD w FPN) [64]</td>
<td>43.9%</td>
<td>48.1%</td>
</tr>
<tr>
<td>PP-YOLO [65]</td>
<td>45.2%</td>
<td>49.9%</td>
</tr>
<tr>
<td>Cascade Mask R-CNN(ResNeXt152, multi-scale) [66]</td>
<td>53.3%</td>
<td>58.5%</td>
</tr>
<tr>
<td>SpineNet-190 [67]</td>
<td>54.3%</td>
<td></td>
</tr>
<tr>
<td>DetectoRS(ResNeXt-101-32x4d, multi-scale) [68]</td>
<td>54.7%</td>
<td>60.1%</td>
</tr>
<tr>
<td>EfficientDet-D7x(multi-scale) [69]</td>
<td>55.1%</td>
<td>59.9%</td>
</tr>
<tr>
<td>CSP-p6 + Mish(multi-scale) [70]</td>
<td>55.2%</td>
<td>60.7%</td>
</tr>
<tr>
<td>DetectoRS(ResNeXt-101-64x4d, multi-scale) [68]</td>
<td>55.7%</td>
<td>61.1%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Network model name</th>
<th>mAP</th>
<th>FPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAS-FPNLite MobileNetV2 [71]</td>
<td>25.7%</td>
<td>3</td>
</tr>
<tr>
<td>YOLOv3-608 [31]</td>
<td>33.0%</td>
<td>20</td>
</tr>
<tr>
<td>SSD512-HarDNet-B5 [72]</td>
<td>35.1%</td>
<td>39.0</td>
</tr>
<tr>
<td>Mask R-CNN X-152-32x8d [73]</td>
<td>40.3%</td>
<td>3</td>
</tr>
<tr>
<td>YOLOv4-608 [63]</td>
<td>43.5%</td>
<td>62.0</td>
</tr>
<tr>
<td>CenterNet HarDNet-85 [72]</td>
<td>43.6%</td>
<td>45.0</td>
</tr>
<tr>
<td>SpineNet-49 [74]</td>
<td>45.3%</td>
<td>29.1</td>
</tr>
<tr>
<td>NAS-FPN AmoebaNet [71]</td>
<td>48.3%</td>
<td>3.6</td>
</tr>
<tr>
<td>EfficientDet-D7x(single-scale) [69]</td>
<td>55.1%</td>
<td>6.5</td>
</tr>
</tbody>
</table>
Outline

• Overview: What is Object detection?
• Top methods for object detection
  • **Object detection with Sliding Window and Feature Extraction** (HoG)
    • Sliding Window technique
    • HOG: Gradient based Features
    • Machine Learning
      • Support Vector Machine (SVM)
      • Non-Maximum Suppression (NMS)
• Implementation examples
• Deformable Part-based Model (DPM)
Histograms of Oriented Gradients for Human Detection

Navneet Dalal and Bill Triggs
INRIA Rhône-Alps, 655 avenue de l’Europe, Montbonnot 38334, France

Abstract
We study the question of feature sets for robust visual object recognition, adopting linear SVM based human detection as a test case. After reviewing existing edge and gradient based descriptors, we show experimentally that grids of Histograms of Oriented Gradient (HOG) descriptors significantly outperform existing feature sets for human detection. We study the influence of each stage of the computation on performance, concluding that fine-scale gradients, fine orientation binning, relatively coarse spatial binning, and high-quality local contrast normalization in overlapping descriptor blocks are all important for good results. The new approach gives near-perfect separation on the original MIT pedestrian database, so we introduce a more challenging dataset containing over 1800 annotated human images with a large range of pose variations and backgrounds.

2 Previous Work
There is an extensive literature on object detection, but here we mention just a few relevant papers on human detection [18, 17, 22, 16, 20]. See [6] for a survey. Papageorgiou et al [18] describe a pedestrian detector based on a polynomial SVM using rectified Haar wavelets as input descriptors, with a parts (subwindow) based variant in [17]. Depoortere et al give an optimized version of this [2]. Gavrila & Phakomen [8] take a more direct approach, extracting edge images and matching them to a set of learned exemplars using chamfer distance. This has been used in a practical real-time pedestrian detection system [7]. Viola et al [22] build an efficient

• CVPR 2005
Sliding Window Technique

• Classification problem:
  • Score for a category
Sliding Window Technique

• Score every subwindow
  • extract features from the image window
  • classifier decides based on the given features.
• It is a brute-force approach
Window-based detection: strengths

Pros

• Sliding window detection and global appearance descriptors:
  • Simple detection protocol to implement
  • Good feature choices critical
  • Past successes for certain classes

Cons

• High computational complexity
  • For example: 250,000 locations x 30 orientations x 4 scales = 30,000,000 evaluations!
  • If training binary detectors independently, means cost increases linearly with number of classes
  • With so many windows, false positive rate better be low
Cons (continued)

• Not all objects are “box” shaped
Limitations (continued)

- If considering windows in isolation, context is lost

Figure credit: Derek Hoiem
Slide: Kristen Grauman
Outline

• Overview: What is Object detection?
• Top methods for object detection
  • Object detection with Sliding Window and Feature Extraction (HoG)
    • Sliding Window technique
    • HoG: Gradient based Features
    • Machine Learning
      • Support Vector Machine (SVM)
      • Non-Maximum Suppression (NMS)
• Implementation examples
• Deformable Part-based Model (DPM)
Let’s examine possible feature vectors

• Pixel based (as a vector)
  • Sensitive to small shifts

• Color based

  • color-based representations are sensitive to color (illumination)
Gradient-based representations

- summarize the local distribution of gradients with histograms
- invariance to small shifts and rotations
- offers more spatial information compared to a single global histogram
- Includes contrast normalization
  - reduce the impact of variable illumination (color)
Histograms of Oriented Gradients (HOG)

- Step 1: Extract a square window (called “block”) of some size around the pixel location of interest.
- Step 2: Divide block into a square grid of sub-blocks (called “cells”) (2x2 grid in our example, resulting in four cells).
- Step 3: Compute orientation histogram of each cell.
- Step 4: Concatenate the four histograms.
- Step 5: normalize v using one of the three options:
  - Option 1 (L2): Divide v by its Euclidean norm.
  - Option 2 (L1): Divide v by its L1 norm (the L1 norm is the sum of all absolute values of v).
  - Option 3 (L2-Hys):
    - Divide v by its Euclidean norm.
    - In the resulting vector, clip any value over 0.2
    - Then, renormalize the resulting vector by dividing again by its Euclidean norm
Histogram of Oriented Gradients (HOG)

• Angles range from 0 to 180 or from 0 to 360 degrees?
  • In the Dalal & Triggs paper, a range of 0 to 180 degrees is used

• Number of orientation bins.
  • Usually 9 bins, each bin covering 20 degrees.

• Cell size.
  • Cells of size 8x8 pixels are often used. (64 → 9)

• Block size.
  • Blocks of size 2x2 cells (16x16 pixels) are often used.

• HOG feature has 36 dimensions.
  • 4 cells * 9 orientation bins.
Calculate HOG Descriptor vector

- The 16×16 window then moves by 8 pixels and a normalized 36×1 vector is calculated over this window and the process is repeated for the image.
- To calculate the final feature vector for the entire image patch, the 36×1 vectors are concatenated into one giant vector.
- Example: an input picture of size 64×64
  - The 16×16 block has 7 positions horizontally and 7 position vertically.
  - In one 16×16 block we have 4 histograms which after normalization concatenate to form a 36×1 vector.
  - This block moves 7 positions horizontally and vertically totalling it to 7×7 = 49 positions.
  - We concatenate them all into one giant vector we obtain a 36×49 = 1764 dimensional vector.
Outline

• Overview: What is Object detection?
• Top methods for object detection
• **Object detection with Sliding Window and Feature Extraction (HoG)**
  • Sliding Window technique
  • HOG: Gradient based Features
  • **Machine Learning**
    • Support Vector Machine (SVM)
    • Non-Maximum Suppression (NMS)
• Implementation examples
• Deformable Part-based Model (DPM)
Support vector machines

[Image source]
Support vector machines

• When the data is linearly separable, there may be more than one separator (hyperplane). Which separator is best?
Linear classifiers

A linear classifier has the form

\[ f(x) = w^\top x + b \]

- In 2D, the discriminant is a line
- \( w \) is the normal to the line, and \( b \) the bias
- \( w \) is known as the weight vector
What is the best $w$?

- **maximum margin** solution: most stable under perturbations of the inputs
Support vector machines

• Find hyperplane that maximizes the *margin* between the positive and negative examples

\[ x_{\text{positive}} (y = 1): \quad x \cdot w + b \geq 1 \]
\[ x_{\text{negative}} (y = -1): \quad x \cdot w + b \leq -1 \]

For support vectors, \( x \cdot w + b = \pm 1 \)

Distance between point and hyperplane:
\[ \frac{|x \cdot w + b|}{||w||} \]

Therefore, the margin is \( \frac{2}{||w||} \)

Finding the maximum margin hyperplane

1. Maximize margin $\frac{2}{\|w\|}$

2. Correctly classify all training data:
   - $x_i$ positive ($y_i = 1$): $x_i \cdot w + b \geq 1$
   - $x_i$ negative ($y_i = -1$): $x_i \cdot w + b \leq -1$

• **Quadratic optimization problem:**

$$\min_{w,b} \frac{1}{2} \|w\|^2 \quad \text{subject to} \quad y_i (w \cdot x_i + b) \geq 1$$

Linear separability again: What is the best $w$?

- The points can be linearly separated but there is a very narrow margin.

- But possibly the large margin solution is better, even though one constraint is violated.

In general there is a trade off between the margin and the number of mistakes on the training data.
Introduce “slack” variables

\[ \xi_i \geq 0 \]

- for \( 0 < \xi \leq \frac{1}{||w||} \) point is between margin and correct side of hyperplane. This is a margin violation
- for \( \xi > \frac{1}{||w||} \) point is misclassified

\[ w^T x + b = 1 \]
\[ w^T x + b = 0 \]
\[ w^T x + b = -1 \]
SVM training in general

- Separable data:
  \[
  \min_{w,b} \frac{1}{2} \|w\|^2 \quad \text{subject to} \quad y_i (w \cdot x_i + b) \geq 1
  \]

  Maximize margin
  Classify training data correctly

- Non-separable data:
  \[
  \min_{w,b} \frac{1}{2} \|w\|^2 + C \max_{i=1}^{n} (0, 1 - y_i (w \cdot x_i + b))
  \]

  Maximize margin
  Minimize classification mistakes
SVM training in general

\[
\min_{w,b} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \max(0, 1 - y_i (w \cdot x_i + b))
\]

• Demo: [http://cs.stanford.edu/people/karpathy/svmjs/demo](http://cs.stanford.edu/people/karpathy/svmjs/demo)
Linear separability

- linearly separable
  - data points can be separated by a single line
- not linearly separable
  - data points cannot be separated by a single line
Nonlinear SVMs

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

\[ \Phi: \mathbf{x} \rightarrow \varphi(\mathbf{x}) \]

**Image source**
Nonlinear SVMs

- Linearly separable dataset in 1D:

- Non-separable dataset in 1D:

- We can map the data to a higher-dimensional space:
The kernel trick

• Linear SVM decision function:

\[ \mathbf{w} \cdot \mathbf{x} + b = \sum_i \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} + b \]

The kernel trick

• Linear SVM decision function:

\[ \mathbf{w} \cdot \mathbf{x} + b = \sum_i \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} + b \]

• Kernel SVM decision function:

\[ \sum_i \alpha_i y_i \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}) + b = \sum_i \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b \]

• This gives a nonlinear decision boundary in the original feature space

The kernel trick

• Instead of explicitly computing the lifting transformation $\phi(x)$, define a kernel function $K$ such that

$$K(x, y) = \phi(x) \cdot \phi(y)$$

• (to be valid, the kernel function must satisfy Mercer’s condition)
Polynomial kernel:

\[ K(x, y) = (c + x \cdot y)^d \]
Gaussian kernel

- Also known as the radial basis function (RBF) kernel:

\[ K(x, y) = \exp \left( -\frac{1}{\sigma^2} \|x - y\|^2 \right) \]
Gaussian kernel

- Demo: http://cs.stanford.edu/people/karpathy/svmjs/demo
SVMs: Pros and cons

• Pros
  • Kernel-based framework is very powerful, flexible
  • Training is convex optimization, globally optimal solution can be found
  • Amenable to theoretical analysis
  • SVMs work very well in practice, even with very small training sample sizes

• Cons
  • No “direct” multi-class SVM, must combine two-class SVMs (e.g., with one-vs-others)
  • Computation, memory (esp. for nonlinear SVMs)
Person detection with HoG’s & linear SVM’s (so far)

- Histogram of oriented gradients (HoG): 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.

Dalal & Triggs, CVPR 2005
The Dalal & Triggs detector

Image pyramid
The Dalal & Triggs detector

1. Compute HOG of the whole image at multiple resolutions!
The Dalal & Triggs detector

1. Compute HOG of the whole image at multiple resolutions!

2. Score every window of the feature pyramid

$$\text{score}(I, p) = w \cdot \phi(I, p)$$

FROM TRAINING
The Dalal & Triggs detector

1. Compute HOG of the whole image at multiple resolutions!
2. Score every window of the feature pyramid
   \[ \text{score}(I, p) = w \cdot \phi(I, p) \]
3. Apply non-maximal suppression (NMS)
Outline

• Overview: What is Object detection?
• Top methods for object detection
  • Object detection with Sliding Window and Feature Extraction (HoG)
    • Sliding Window technique
    • HOG: Gradient based Features
    • Machine Learning
      • Support Vector Machine (SVM)
    • Non-Maximum Suppression (NMS)
• Implementation examples
• Deformable Part-based Model (DPM)
Non-Maximum Suppression

Before non-max suppression

After non-max suppression

Non-Max Suppression
Non-Maximum Suppression

Algorithm 1: Non-Max Suppression

1: procedure NMS($B, c$)
2: \[ B_{nms} \leftarrow \emptyset \] \hspace{1cm} \text{Initialize empty set}
3: \hspace{1cm} \textbf{for} $b_i \in B \textbf{ do}$ \hspace{0.5cm} \text{Iterate over all the boxes}
4: \hspace{1.5cm} \text{discard} \leftarrow \text{False} \hspace{1.5cm} \text{Take boolean variable and set it as false. This variable indicates whether b(i)
5: \hspace{1.5cm} \textbf{for} $b_j \in B \textbf{ do}$ \hspace{1cm} \text{should be kept or discarded}
6: \hspace{2.5cm} \textbf{if} \ \text{same}(b_i, b_j) > \lambda_{nms} \textbf{ then} \hspace{1cm} \text{Start another loop to compare with b(i)}
7: \hspace{3.5cm} \text{if} \ \text{score}(c, b_j) > \text{score}(c, b_i) \textbf{ then} \hspace{1cm} \text{If both boxes having same IOU}
8: \hspace{4.5cm} \text{discard} \leftarrow \text{True} \hspace{1cm} \text{Compare the scores. If score of b(i) is less than that}
9: \hspace{5.5cm} \textbf{if not} \ \text{discard} \textbf{ then} \hspace{1cm} \text{of b(j), b(i) should be discarded, so set the flag to True.}
10: \hspace{6.5cm} B_{nms} \leftarrow B_{nms} \cup b_i \hspace{1cm} \text{Once b(i) is compared with all other boxes and still the}
11: \hspace{7.5cm} \textbf{return} B_{nms} \hspace{1cm} \text{discarded flag is False, then b(i) should be considered. So add it to the final list.}
12: \hspace{1cm} \textbf{end procedure} \hspace{1cm} \textbf{do the same procedure for remaining boxes and return the final list}
Outline

• Overview: What is Object detection?
• Top methods for object detection
  • **Object detection with Sliding Window and Feature Extraction (HoG)**
    • Sliding Window technique
    • HOG: Gradient based Features
    • Machine Learning
      • Support Vector Machine (SVM)
      • Non-Maximum Suppression (NMS)
• Implementation examples
• Deformable Part-based Model (DPM)
Implementation example (car detector)
Get the data. UIUC Car Database

- 550 positive images
- 500 negatives
Implementation example (car detector)

- Extract features

```python
print("Calculating the descriptors for the positive samples and saving them")
for im_path in glob.glob(os.path.join(pos_im_path, "*.")):
im = imread(im_path)
# print(image shape)
if des_type == "HOG":
    fd = hog(im, orientations, pixels_per_cell, cells_per_block, visualize=visualize, block_norm='L2-Hys')
fd_name = os.path.split(im_path)[1].split(".")[0] + ".feat"
fd_path = os.path.join(pos_feat_ph, fd_name)
joblib.dump(fd, fd_path)
print("Positive features saved in {0}.format(pos_feat_ph)

print("Calculating the descriptors for the negative samples and saving them")
for im_path in glob.glob(os.path.join(neg_im_path, "*.")):
im = imread(im_path)
if des_type == "HOG":
    fd = hog(im, orientations, pixels_per_cell, cells_per_block, visualize=visualize, block_norm='L2-Hys')
fd_name = os.path.split(im_path)[1].split(".")[0] + ".feat"
fd_path = os.path.join(neg_feat_ph, fd_name)
joblib.dump(fd, fd_path)
print("Negative features saved in {0}.format(neg_feat_ph)"
```
Implementation
e
eexample (car detector)

- Train SVM with HOG features
Implementation example (car detector)

Test

- Load image
- Loop over different pyramid images
  - loop the window position
  - Compute HOG for each window
  - Compute score

```python
from skimage.transform import pyramid_gaussian
from skimage.io import imread
from skimage.feature import hog
import joblib
import cv2
import argparse as ap
from nms import nms
from config import *
import numpy as np

# Downscale the image and iterate
for im_scaled in pyramid_gaussian(im, downscale=downscale):
    print(im_scaled.shape)
    # This list contains detections at the current scale
    cd = []
    # If the width or height of the scaled image is less than
    # the width or height of the window, then end the iterations.
    if im_scaled.shape[0] < min_wdw_sz[1] or im_scaled.shape[1] < min_wdw_sz[0]:
        break
    for (x, y, im_window) in sliding_window(im_scaled, min_wdw_sz, step_size):
        print("(x, y, im_window): ", x, y, im_window.shape)
        if im_window.shape[0] != min_wdw_sz[1] or im_window.shape[1] != min_wdw_sz[0]:
            continue
        # Calculate the HOG features
        if visualize:
            (fd, imgVis) = hog(im_window, orientations, pixels_per_cell, cells_per_block, visualize=True, block_norm='l2-Hys')
            cv2.imshow("HOGinput", imgVis)
            cv2.waitKey(30)
        else:
            fd = hog(im_window, orientations, pixels_per_cell, cells_per_block, visualize=False, block_norm='l2-Hys')
        fd = fd[np.newaxis, :]
        pred = clf.predict(fd)
        if pred == 1:
            print("Detection: Location -> (" + str(x) + ", " + str(y) + ")")
            print("Scale -> (" + str(scale) + ") | Confidence Score -> " + str(clf.decision_function(fd)) + ", n")
            detections.append((x, y, clf.decision_function(fd),
                                int(min_wdw_sz[0])*(downscale**scale),
                                int(min_wdw_sz[1])*(downscale**scale)))
```
Implementation example (car detector)

Test
- Load image
- Loop over different pyramid images
  - loop the window position
    - Compute HOG for each window
  - Compute score
- Perform NMS

```python
from skimage.transform import pyramid_gaussian
from skimage.io import imread
from skimage.feature import hog
import joblib
import cv2
import argparse as ap
from nms import nms
from config import *
import numpy as np

# Downscale the image and iterate
for im_scaled in pyramid_gaussian(im, downscale=downscale):
    print(im_scaled.shape)
    # This list contains detections at the current scale
    cd = []
    # If the width or height of the scaled image is less
    # than the width or height of the window, then end the iterations.
    if im_scaled.shape[0] < min_wdw_sz[1] or im_scaled.shape[1] < min_wdw_sz[0]:
        break
    for (x, y, w, h) in sliding_window(im_scaled, min_wdw_sz, step_size):
        print((x, y, w, h))
        if im_window.shape[0] := min_wdw_sz[1] or im_window.shape[1] := min_wdw_sz[0]:
            continue
        # Calculate the HOG features
        if visualize:
            (fd, imgVis) = hog(im_window, orientations, pixels_per_cell, cells_per_block, visualize=True, block_norm='L2-Hys', cv2.imshow('HOG Input', imgVis)
            cv2.waitKey(30)
        else:
            fd = hog(im_window, orientations, pixels_per_cell, cells_per_block, visualize=False, block_norm='L2-Hys')
        fd = fd[np.newaxis, :]
        print(fd.shape)
        pred = clf.predict(fd)
        if pred == 1:
            print("Detection: Location -> (" + str(x) + "," + str(y) + ")")
            print("Scale -> " + str(scale) + ": " + str(confirm_score) + ": " + clf.decision_function(fd) + "\n")
            print("Scale -> " + str(scale) + ": " + str(confirm_score) + ": " + clf.decision_function(fd))
            detections.append((x, y, clf.decision_function(fd),
            int(min_wdw_sz[0] * (downscale ** scale)),
            int(min_wdw_sz[1] * (downscale ** scale))))
        cd.append(detections[-1])
```

Testing (different pyramid levels)
NMS

Before NMS

After NMS
Questions?