EE 569 Discussion

Yao Zhu

02/05/2021

Announcements

- HW1
  Due: 02/07/20 11:59pm

- HW2
  Issue: 02/08/2020
  Due: 02/23/20 11:59pm

Announcements

Review for the HW submission policies (should be strictly obeyed!):

- Submit your homework on time. Do NOT wait until the last minute. Otherwise your submission will be marked as late.
- Late submission can be graded, but the score will not be counted into your grades. i.e, you will NOT earn any score if your submission is late.
- Only the last submission BEFORE the deadline will be graded. Will not consider the previous submissions.

Announcements

For your Reports.pdf and Code.txt:

- Do NOT copy any of the sentences from the Homework Instruction in your Reports.

> **(a) Sobel Edge Detector (10%)**
>
> Implement the Sobel edge detector and apply to *Gallery.raw* and *Dogs.raw* images as shown in Fig. 1 (a) and (b). Note that you need to convert RGB images to grayscale image first. The relationship between RGB and grayscale images are provided in the Appendix. Include the following in your results:

- Do NOT copy from Wikipedia, paper, etc.
- Do NOT include the provided source code in your code.txt (including writeraw, readraw etc.)

Announcements

Preview of Discussion Sessions:

- …
- Week 4 (Feb. 05): Edge Detection
- Week 5 (Feb. 12): Digital Half-toning
- Week 6 (Feb. 19): HW3
- Week 7 (Feb. 26): Midterm I Review
- …

Contents

- Overview
- Sobel Edge Detector
- Canny Edge Detector
- Structured Edge (Machine-learning based)
- Evaluation

Edge Detection

Edge in an image:

- Locally: a set of pixels with sharp intensity change
- Globally: the boundaries between objects & background

Edge Detection

Edge Detection Usage:

- Reduce unnecessary information but preserve the structure
- Extract important features: corners, lines, and curves
- Serve for higher level tasks:
  Object recognition, segmentation, saliency detection, …

Edge Detection

What we get is an “edge map”:

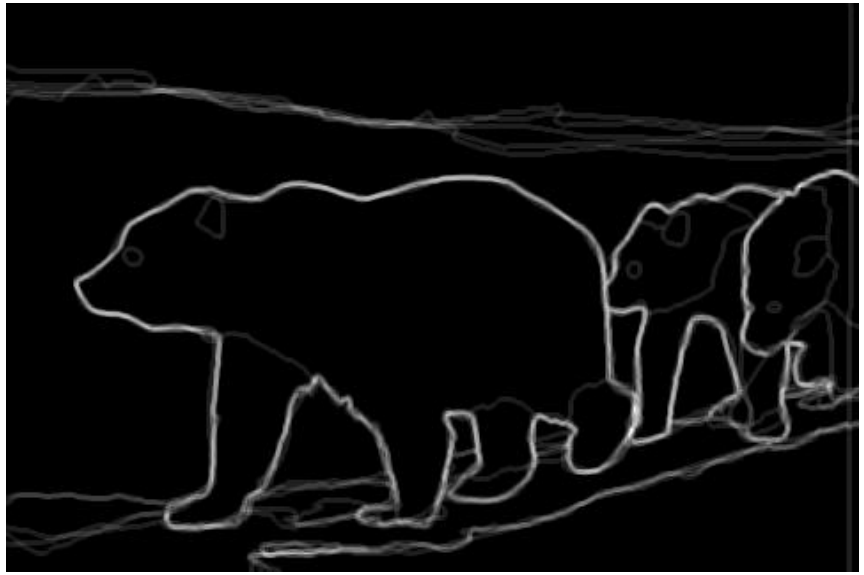
- Edge probability map:

  Values at each pixel represents its “edgeness” score (probability from 0-1)

- Binary edge map:

  “1” – is an edge (white)

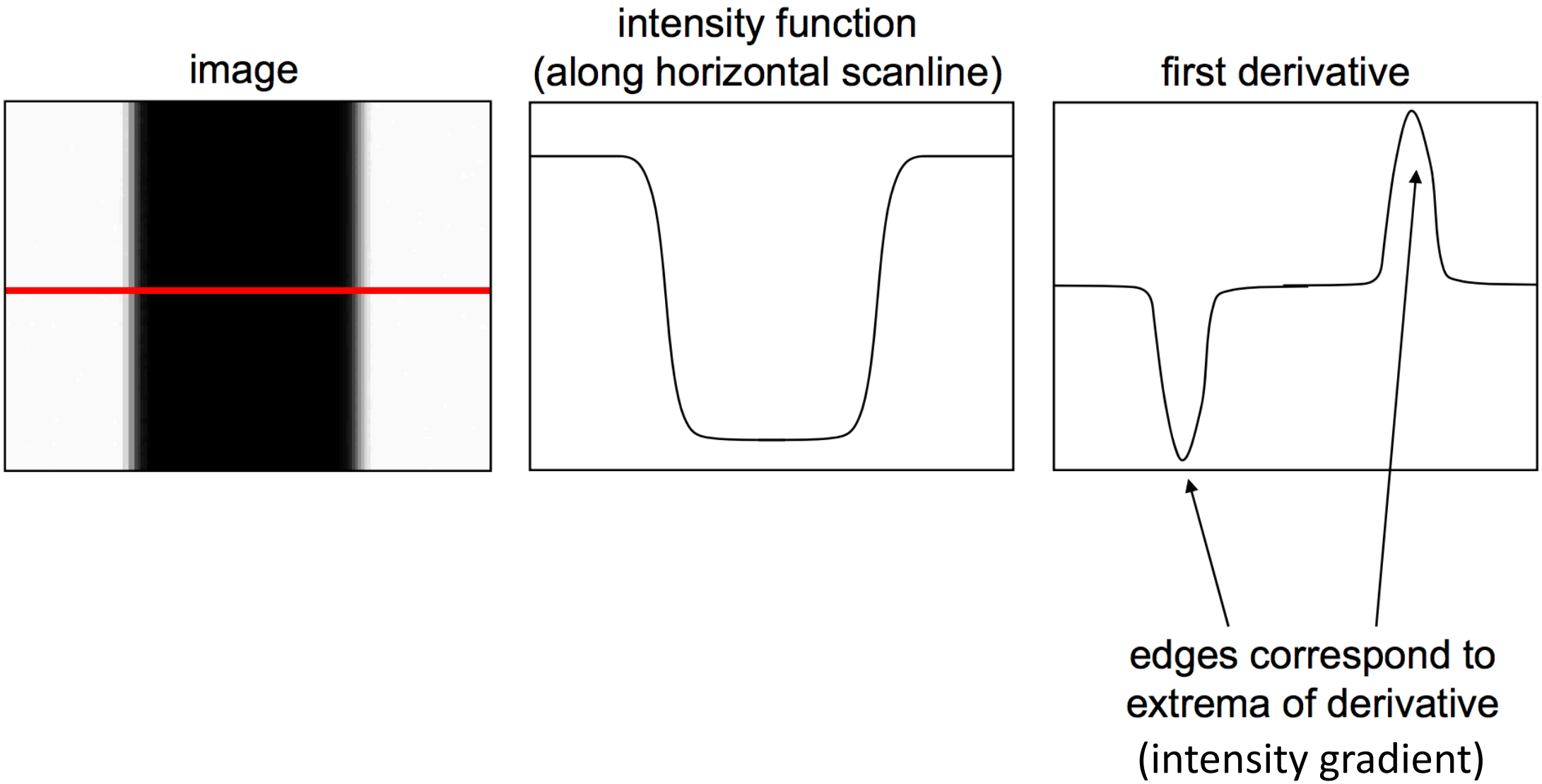
  “0” – is not an edge (black)



Contents

- Overview
- Sobel Edge Detector
- Canny Edge Detector
- Structured Edge (Machine-learning based)
- Evaluation

HW2 P1(a) Sobel Edge Detector



HW2 P1(a) Sobel Edge Detector

- Find edge by checking gradient

Gradient

$$\nabla f = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

Magnitude

$$|\nabla f| = \sqrt{g_y^2 + g_x^2}$$

Orientation

$$\theta = \tan^{-1}\left[\frac{g_y}{g_x}\right]$$

- Approach

continuous gradient $\xrightarrow{\text{Analytic Resolution}}$ discrete finite difference

HW2 P1(a) Sobel Edge Detector

- Compute the finite difference at the center pixel by convolution

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

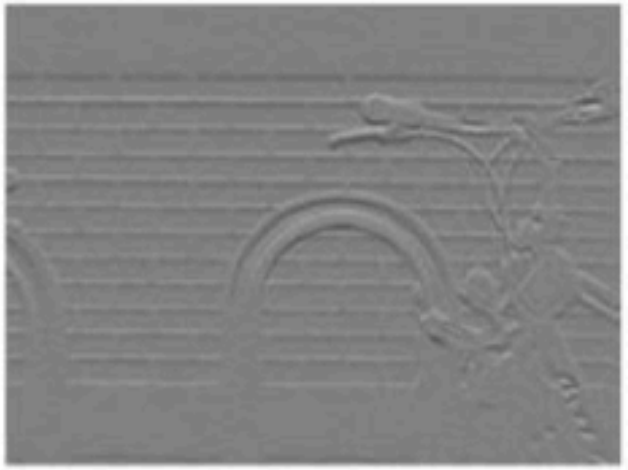
Gy

3-by-3 Sobel filter

Normalize the filter or not does not influence the final normalized gradient map

HW2 P1(a) Sobel Edge Detector

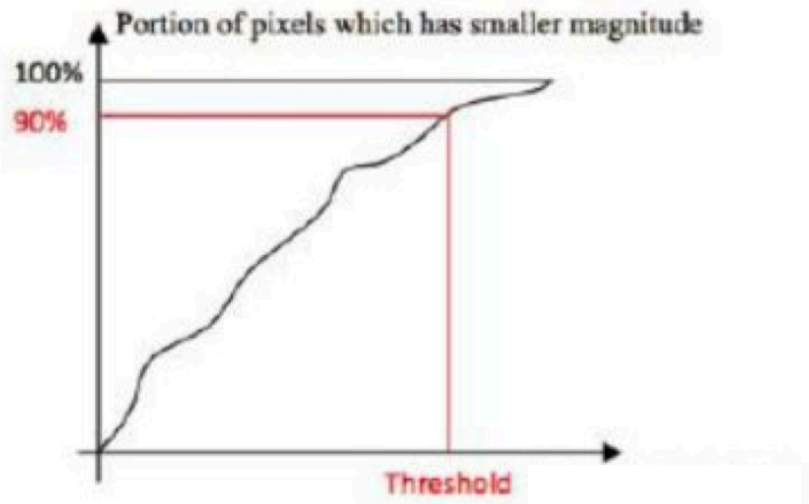
- Normalize results for output (0~255)



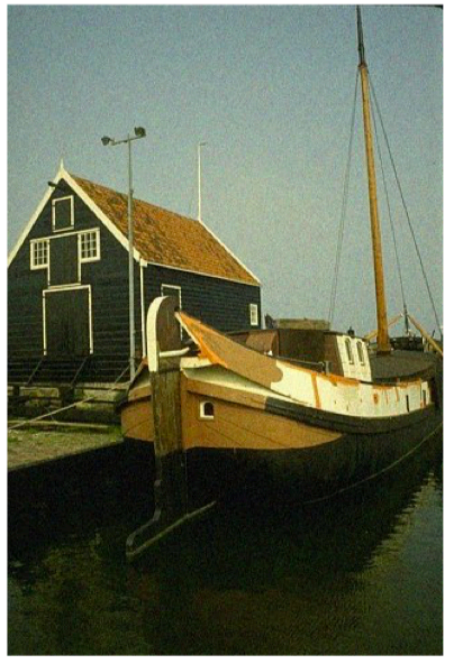
Gradient magnitude

HW2 P1(a) Sobel Edge Detector

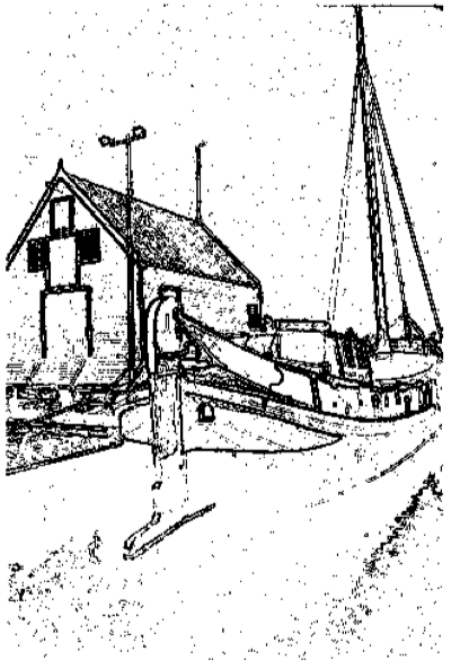
- Thresholding



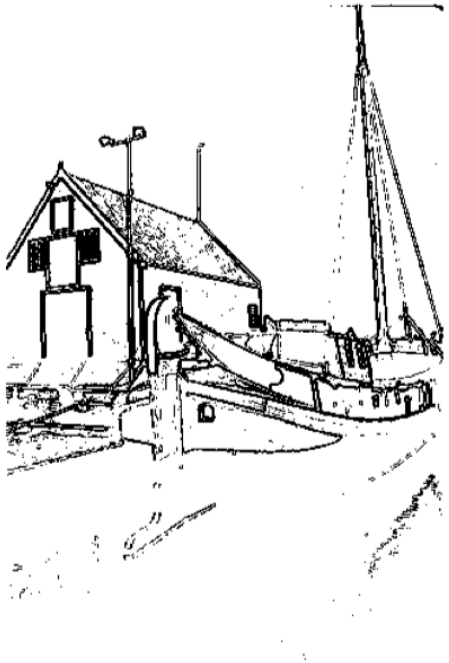
Gradient Magnitude
Cumulative Distribution



Origin



Sobel Edge Map
Threshold = 85%



Sobel Edge Map
Threshold = 90%

Contents

- Overview
- Sobel Edge Detector
- Canny Edge Detector
- Structured Edge (Machine-learning based)
- Evaluation

HW2 P1(b) Canny Edge Detector

Basic Steps:

- Filter image with Gaussian
- Find the intensity gradients of the image
- Non-maximum Suppression (NMS)
- Double Thresholding (Hysteresis Thresholding)

MATLAB:
https://www.mathworks.com/help/images/ref/edge.html

OpenCV
https://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/canny_detector/canny_detector.html

Non-maximum Suppression (NMS)

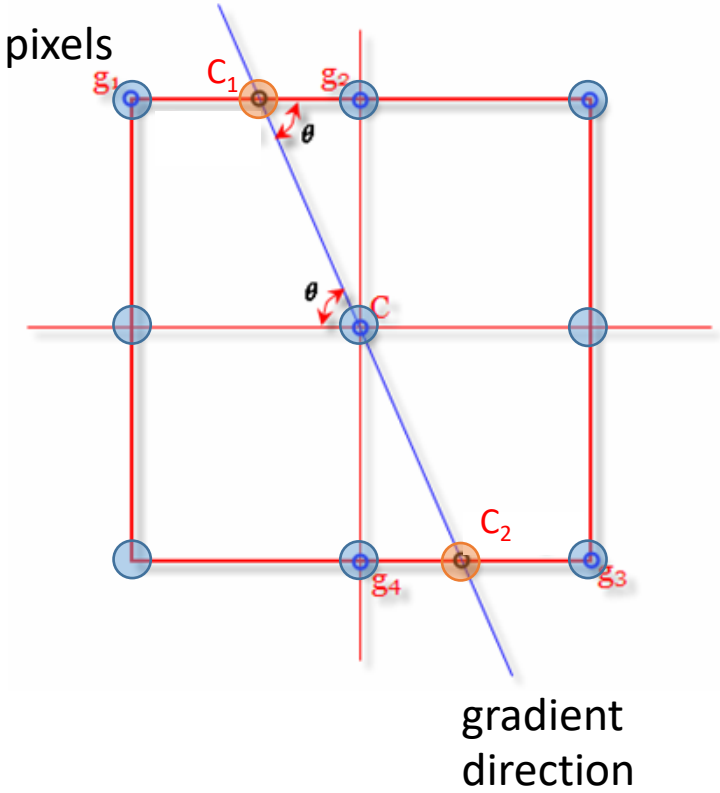
- Motivation

  This removes pixels that are not considered to be part of an edge.

  Hence, it will result in thin edges (candidate edges).

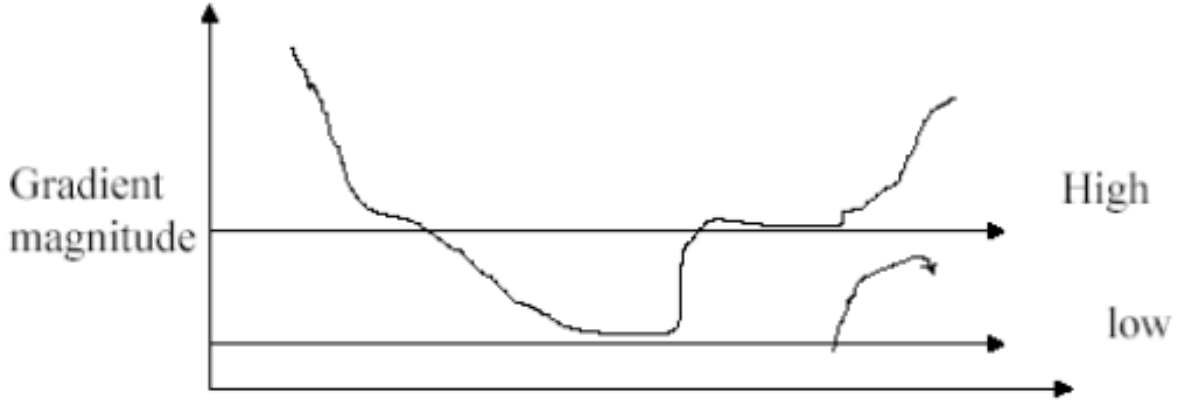
Non-maximum Suppression (NMS)

- Approach
  Suppress all the gradient values to 0 except for the local maxima



Preserve gradient magnitude at pixel C if its value is larger than that of C_1 and C_2 (by interpolation). Otherwise, suppress it to be 0.

Double Thresholding (Hysteresis Thresholding)



Canny recommended an *upper*:*lower* ratio between 2:1 and 3:1.

- If a pixel gradient is higher than the *upper* threshold, the pixel is accepted as an edge
- If a pixel gradient value is below the *lower* threshold, then it is rejected.
- If the pixel gradient is between the two thresholds, then it will be accepted only if it is connected to a pixel that is above the *upper* threshold.

Contents

- Overview
- Sobel Edge Detector
- Canny Edge Detector
- Structured Edge (Machine-learning based)
- Evaluation

HW2 P1(c) Structured Edge

A Data-driven Edge Detector

- Training (need ground truth annotation) → supervised learning
- Validation
- Testing (get predicted edge maps)

Online open source in Matlab: https://github.com/pdollar/edges

- My system environment: MacOS Big Sur, MATLAB R2017b
- Follow the instructions in the readme.txt

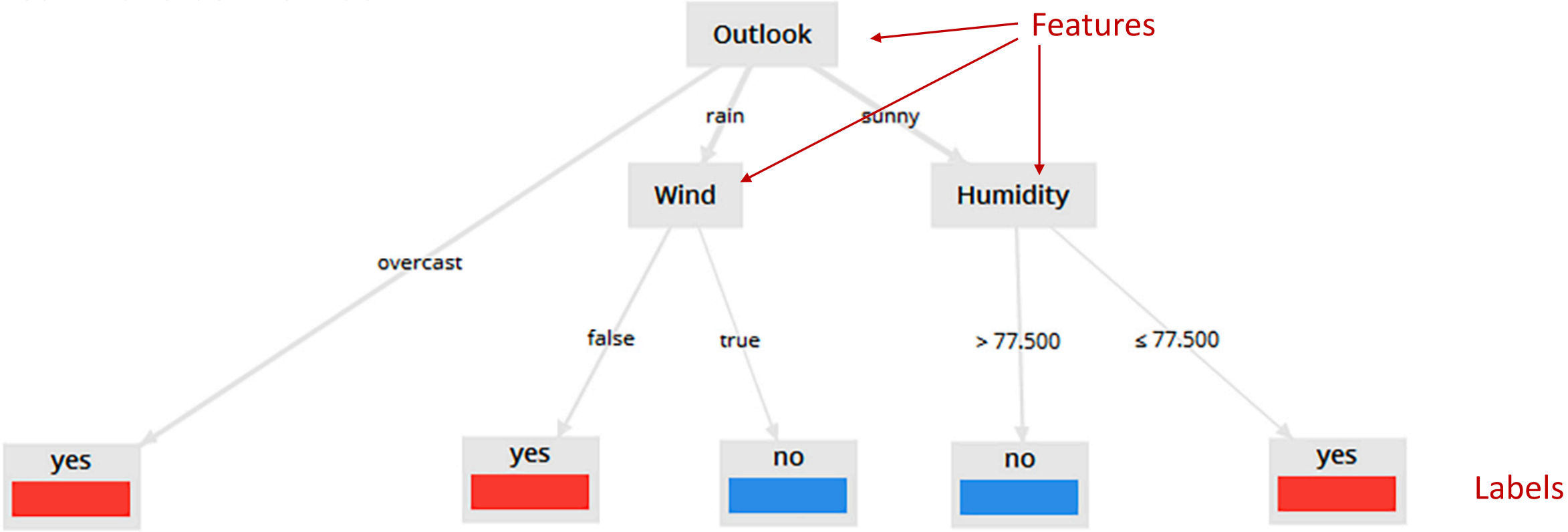
Decision Tree (background knowledge)

Example:

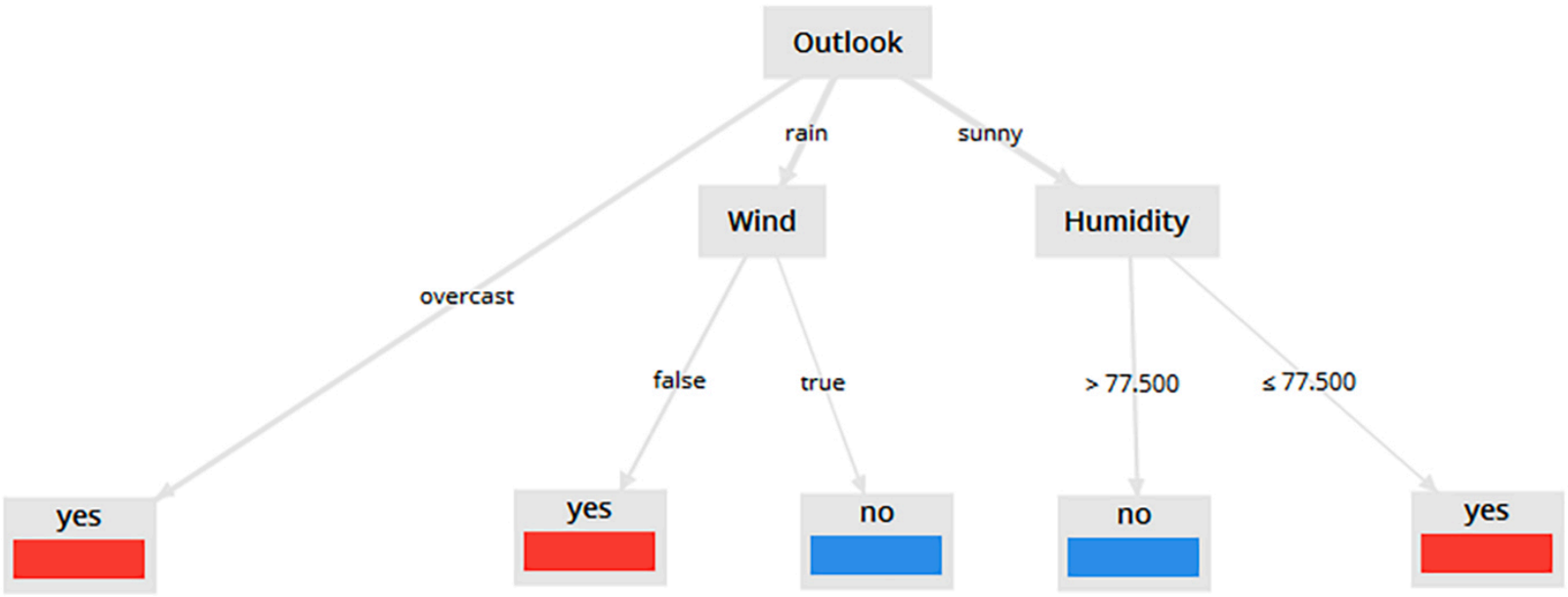
Prediction task is “play tennis or not”

- Train: build the tree
- Test: traverse the tree

	Outlook	Wind	Humidity	Play tennis?
x1				
x2				
x3				



Decision Tree (background knowledge)



To decide which ”feature” to be chosen as the splitting criterion at a node in training:

- Measure the information gain (purity/entropy…)
- Choose the one that gives the greatest information gain

Structured Decision Tree

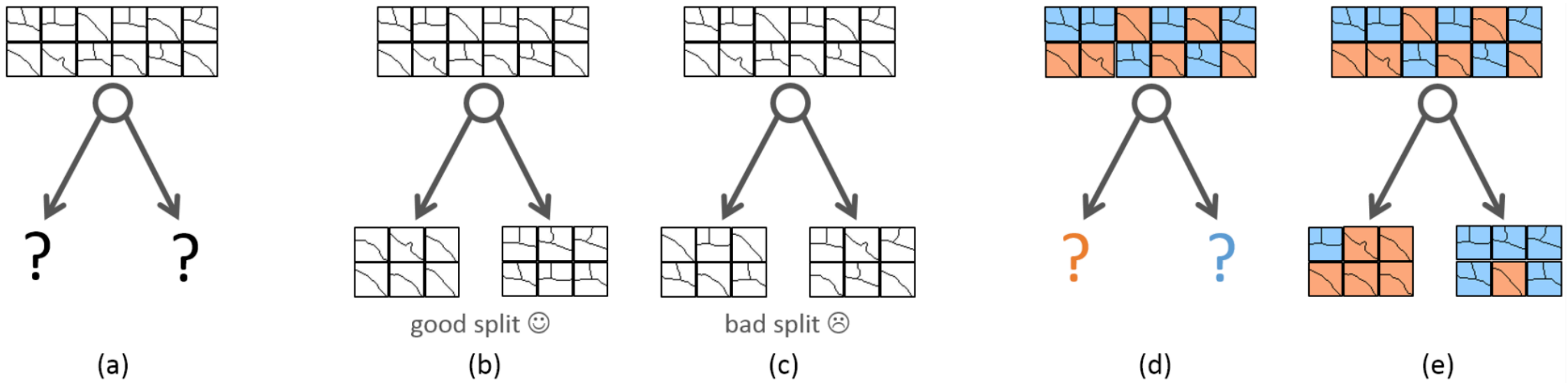
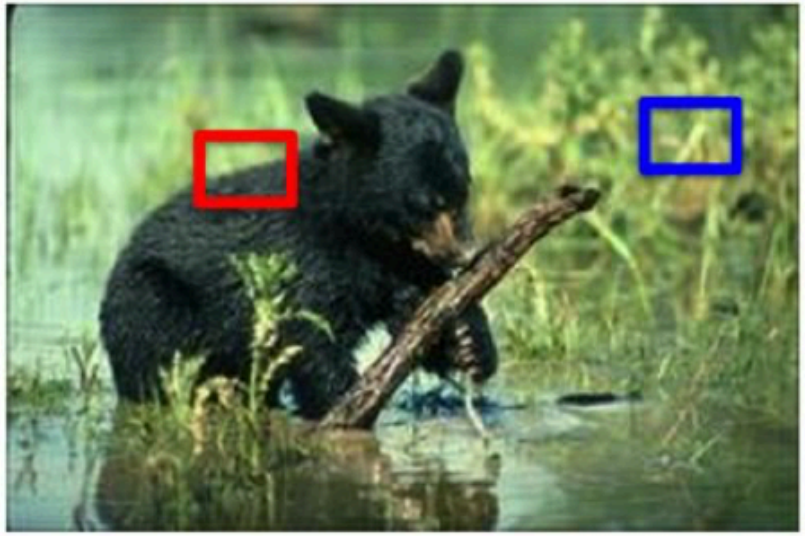


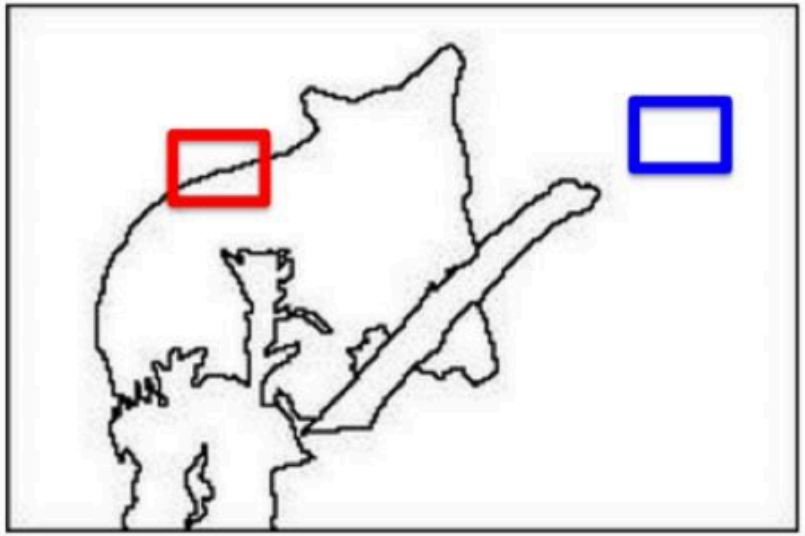
Fig. 2. Illustration of the decision tree node splits: (a) Given a set of structured labels such as segments, a splitting function must be determined. Intuitively a good split (b) groups similar segments, whereas a bad split (c) does not. In practice we cluster the structured labels into two classes (d). Given the class labels, a standard splitting criterion, such as Gini impurity, may be used (e).

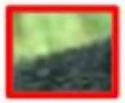
structured labels are 16×16 segmentation mask (from GT) or binary edge map

Training

- Prepare training data and labels





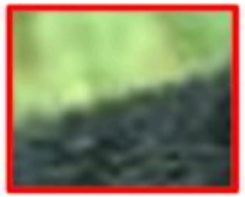
 → edge

→ no edge

} our training data

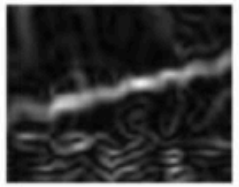
Training

- Extracting features from patches



matrix $\mathbf{P}$

compute gradients
$\rightarrow$



matrix $\mathbf{G}$

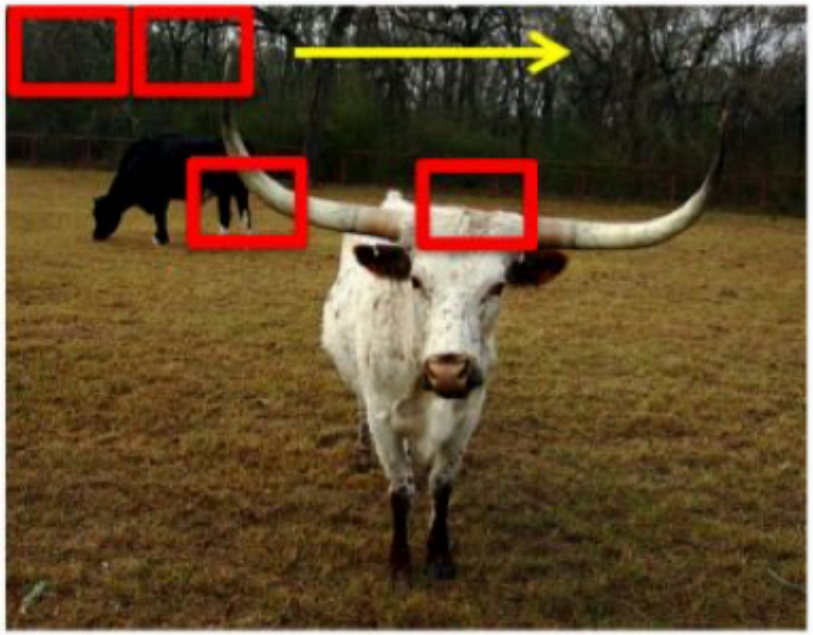
$$\mathbf{x} = \mathbf{G}(:)$$

compute color
$\rightarrow$

- Train the structured random forest

Testing

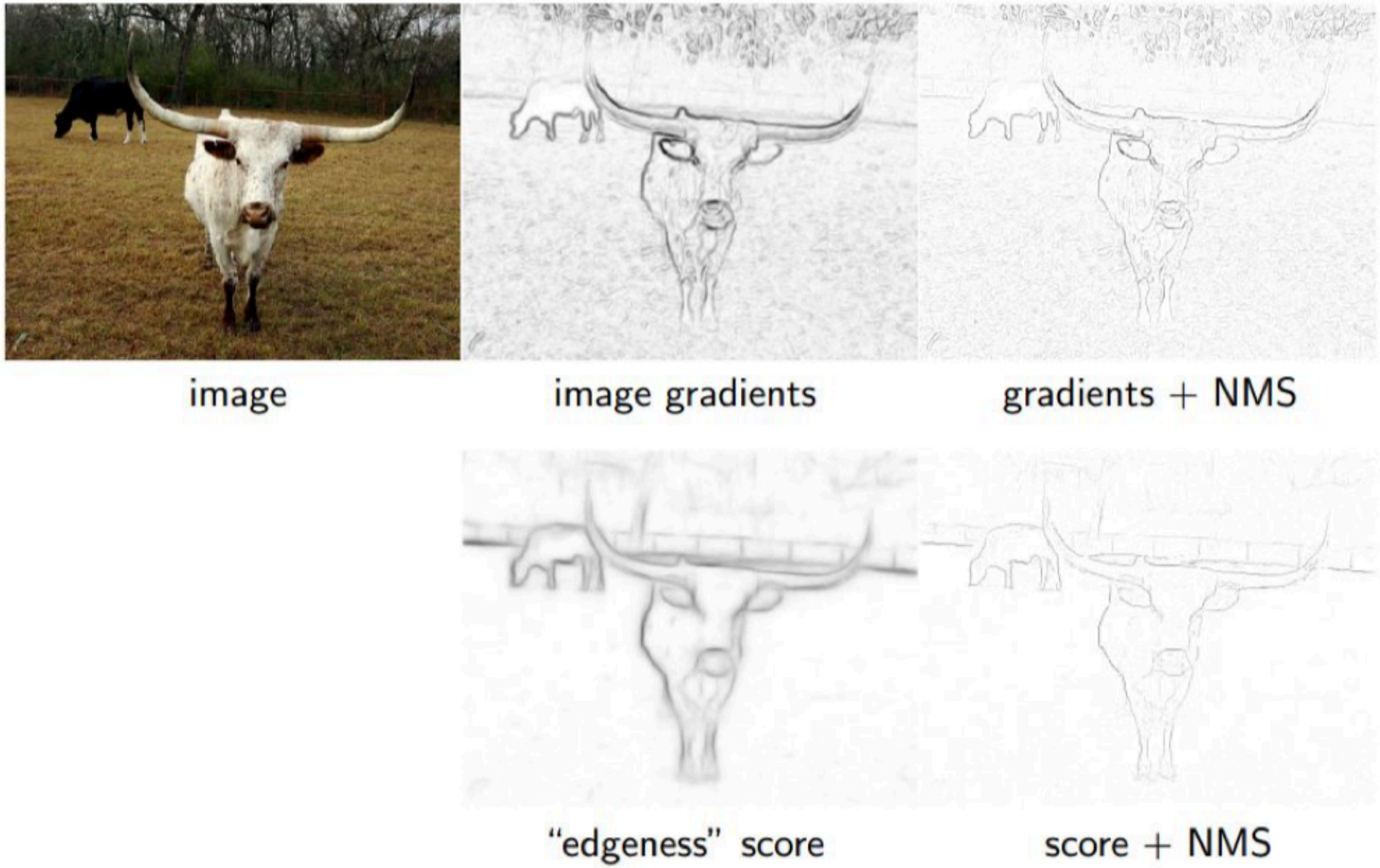
- Extracting patches
- Extract features and use classifier (structured random forest)



image

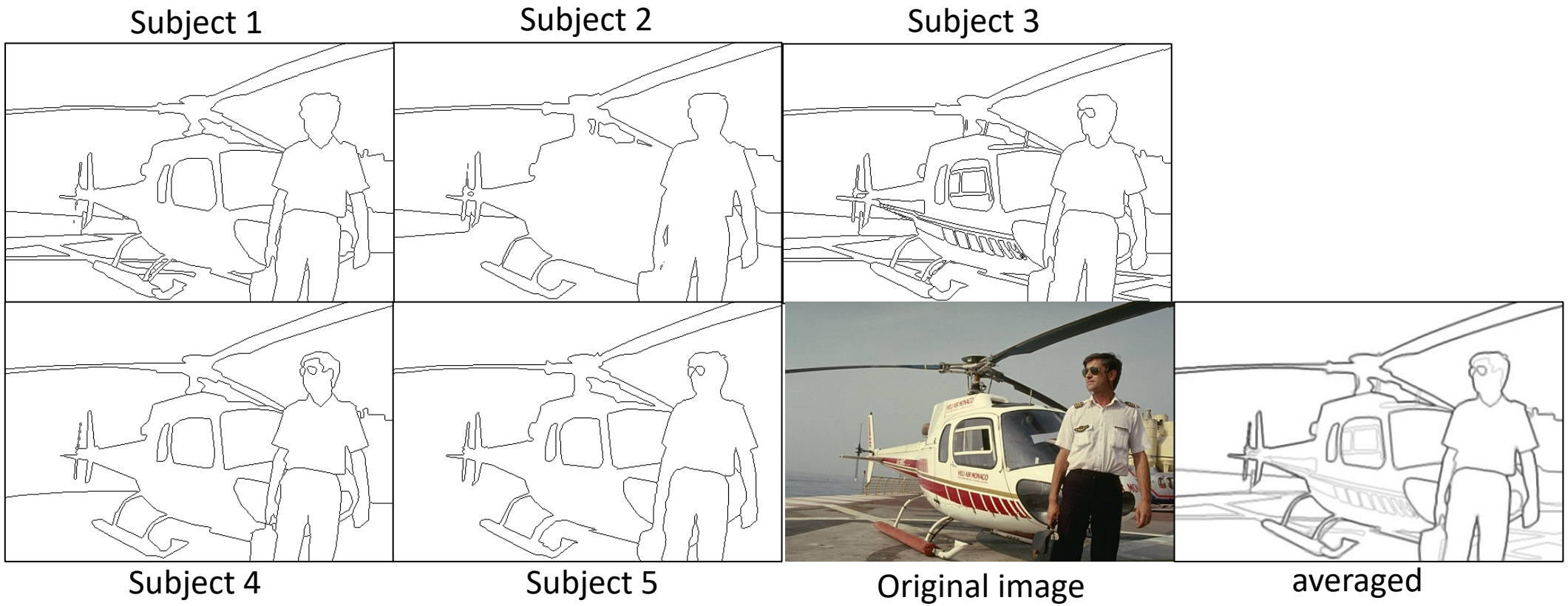
prediction

Performance Comparison



Dataset (BSDS-500)

- Berkeley Segmentation Dataset 500
  - 500 images (200 train + 100 validation + 200 test)
  - Each image was annotated by five subjects on average
  - Served as evaluation of both “Segmentation” and “Contour”



Dataset (BSDS-500)

- GT is provided as *.mat.
- Different annotations are stored using Cell in Matlab

```
elephant = load('elephant_GT.mat');
elephant_GT = elephant.groundTruth;
elephant_GT1_bwmap = elephant_GT{1,1}.Boundaries;
```

dog

1x1 struct with 1 field

Field	Value
groundTruth	1x5 cell

dog_gt

1x5 cell

	1	2	3	4	5
1	1x1 struct	1x1 struct	1x1 struct	1x1 struct	1x1 struct

dog_gt{1, 1}

Field	Value
Segmentation	321x481 uint16
Boundaries	321x481 logical

HW2 P1(c) Structured Edge

edgesDemo.m

```
% Demo for Structured Edge Detector (please see readme.txt first).

%% set opts for training (see edgesTrain.m)
opts=edgesTrain();
opts.modelDir='models/';
opts.modelFnm='modelBsds';
opts.nPos=5e5; opts.nNeg=5e5;
opts.useParfor=0;

%% train edge detector (~20m/8Gb per tree, proportional to nPos/nNeg)
tic, model=edgesTrain(opts); toc; % will load model if already trained

%% set detection parameters (can set after training)
model.opts.multiscale=1;
model.opts.sharpen=2;
model.opts.nTreesEval=4;
model.opts.nThreads=4;
model.opts.nms=1;

%% evaluate edge detector on BSDS500 (see edgesEval.m)
if(0), edgesEval( model, 'show',1, 'name','' ); end

%% detect edge and visualize results
I =
tic, E=edgesDetect(I,model); toc
```

Training Parameters do not change

Model Parameters can change

Store your image data as WxHx3

Your probability edge map will be in E

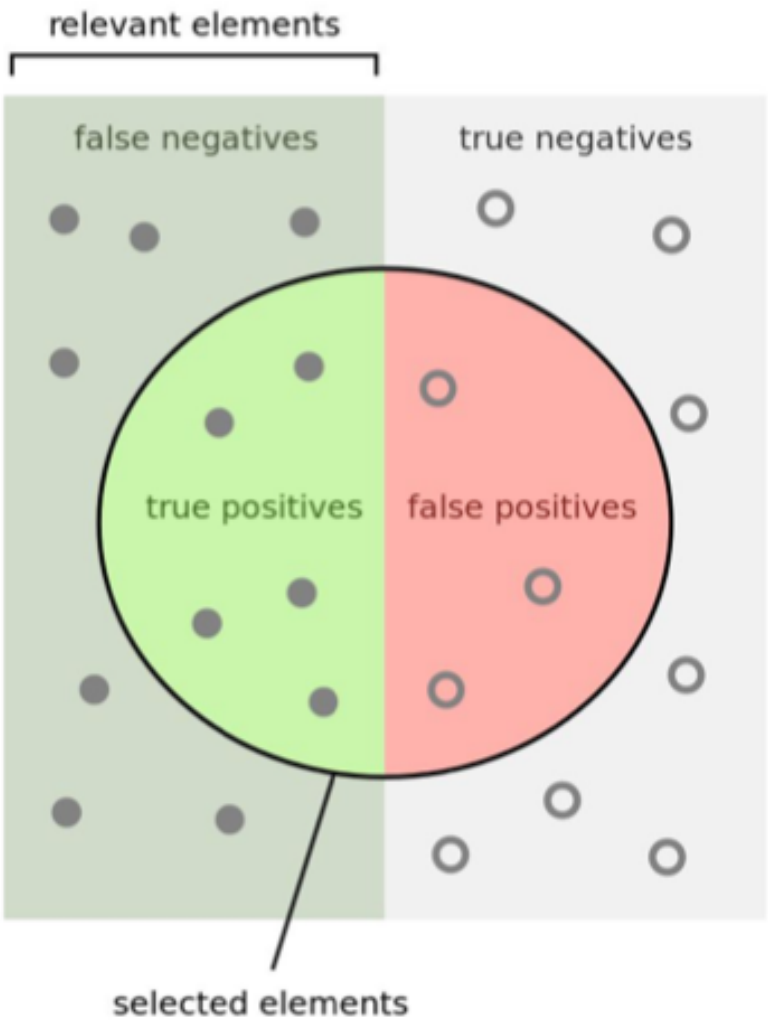
Discuss

Contents

- Overview
- Sobel Edge Detector
- Canny Edge Detector
- Structured Edge (Machine-learning based)
- Evaluation

HW2 P1(d) Performance Evaluation

- True positive: Edge pixels in the edge map coincide with edge pixels in the ground truth.
- True negative: Non-edge pixels in the edge map coincide with non-edge pixels in the ground truth.
- False positive (false alarm): Edge pixels in the edge map correspond to the non-edge pixels in the ground truth.
- False negative (missing): Non-edge pixels in the edge map correspond to the true edge pixels in the ground truth.



HW2 P1(d) Performance Evaluation

- Recall = True Positive / (True Positive + False Negative)
- Precision = True Positive / (True Positive + False Positive)
- Mean recall and mean precision (based on each ground truth)
- F-measure:

$$F_1 = 2 \cdot \frac{1}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}.$$

HW2 P1(d) Performance Evaluation

Recommend using Matlab for this part

- [thrs,cntR,sumR,cntP,sumP,V]=edgesEvalImg(E,Your ground truth file);
- Can I implement my own function instead of using the provided function?
  - **Not recommended**, because when performing edge pixel matching, we will consider the edge points in edge map and ground truth match if they are within a distance range. Exact matching will result in very low precision and recall
  - You can try to set prms.maxDist = 0 to see the result
  - See the Inputs specification of **prms** in edgesEval.m
- You only need to use **cntR (TP), sumR(TP+FN) , cntP (TP), sumP (TP+FP)**
- thrs is the threshold values that you use to binarize the probability edge map
- R = cntR./sumR; gives you the recall values under different threshold values
- P = cntP./sumP; gives you the precision values under different threshold values

HW2 P1(d) Performance Evaluation

Pseudo Code

can view the mean over threshold as an approximation to (rescaled) area under curve, which is a trade-off between precision and recall

```
for each GT g:
    for each threshold t:
        compute Precision(g,t) and Recall(g,t)

for each GT g:
    compute mean_Precision_over_thresholds(g), mean_Recall_over_thresholds(g) #"mean precision, recall for each ground truth"
compute mean_F = 2 * mean(mean_Precision_over_thresholds) * mean(mean_Recall_over_thresholds) /
                    (mean(mean_Precision_over_thresholds) + mean(mean_Recall_over_thresholds))

for each thredhold t:
    compute mean_Precision_over_GTs(t), mean_Recall_over_GTs(t)
    compute F(t) = 2 * mean_Precision_over_GTs(t) * mean_Recall_over_GTs(t) / (mean_Precision_over_GTs(t) + mean_Recall_over_GTs(t))
plot F #x-axis would be threshold values
find the best F-score max(F)
```

HW2 P1 SE Possible Pitfalls

- **Compilation error in MATLAB**
  - Structured Edge package use MATLAB R2013b, compatibility problem with later MATLAB version, OS version, C++ version, etc
- **Helpful links:**
  - https://github.com/pdollar/edges/issues/18
  - https://stackoverflow.com/questions/23333949/the-correct-way-to-define-default-argument-for-a-friend-function-in-c

Q
A

Reference

http://www.cs.utoronto.ca/~fidler/slides/CSC420/lecture5.pdf
https://www.slideshare.net/ishraqabd/edge-detection
http://www.cs.unc.edu/~lazebnik/spring09/lec06_edge.pdf
EE569_discussion3_2020spring by Yijing Yang