Readings

• Slide Credits: Criminisi and Shotton
  Z. Tu
  R. Cipolla
Common Terminologies

- Decision Forests
- Random Forests
- Randomized Decision Forests
The Abstract Forest Model

Decision Forests

- Regression
- Semi-supervised learning
- Classification
- Density estimation
- Manifold learning
- Active learning
General Idea

Training Data

Multiple Data Sets

Multiple Classifiers

Combined Classifier
The Abstract Forest Model

Recognizing the type of a scene captured in a photograph can be cast as a classification task. The desired output is a discrete label:
- Sky, dog, cat, indoor, outdoor, etc.
Recognizing the type of a scene captured in a photograph can be cast as a classification task. The desired output is a discrete label:

- Sky, dog, cat, indoor, outdoor, etc.

Predicting the price of a house as a function of its distance from a good school may be thought of as a regression problem. In this case the output is a continuous variable.
Recognizing the type of a scene captured in a photograph can be cast as a classification task. The desired output is a **discrete label**:  
- Sky, dog, cat, indoor, outdoor, etc.

Predicting the price of a house as a function of its distance from a good school may be thought of as a **regression problem**. In this case the output is a **continuous variable**.

Interactive image segmentation may be thought of as a semi-supervised problem, where the user’s brush strokes define labeled data and the rest of image pixels provide already available unlabeled data.

Can be applied to both regression and classification problems!
Recognizing the type of a scene captured in a photograph can be cast as a classification task. The desired output is a discrete label: • Sky, dog, cat, indoor, outdoor, etc.

Predicting the price of a house as a function of its distance from a good school may be thought of as a regression problem. In this case the output is a continuous variable.

Interactive image segmentation may be thought of as a semi-supervised problem, where the user’s brush strokes define labeled data and the rest of image pixels provide already available unlabeled data.

Problems related to the automatic or semi-automatic analysis of complex data such as photographs, videos, medical scans, text or genomic data can all be categorized into a relatively small set of prototypical machine learning tasks.
Decision Tree Basics

• DT has been around for a number of years.
Decision Tree Basics

• DT has been around for a number of years.
Decision Tree Basics

• DT has been around for a number of years.
• Their recent revival is mostly due to the discovery that higher accuracy on previously unseen data (i.e., generalization)
A tree is a special type of graph: nodes and edges.
Decision Tree Basics

A tree is a special type of graph: nodes and edges.

Nodes -> internal nodes (split): circle

Terminals (leaf): square
Decision Tree Basics

A tree is a special type of graph: nodes and edges.

**Nodes** -> internal nodes (split): circle  
and  
**Terminals (leaf): square**

- Tree cannot contain loops  
- All nodes have one incoming edge (except root)
Decision Tree Basics

• Solve a complex problem by running simpler tests!
Decision Tree Basics

• Solve a complex problem by running simpler tests!
• Decision Tree: simpler tests are organized in a tree structure!
Decision Tree Basics

• Solve a complex problem by running simpler tests!
• Decision Tree: simpler tests are organized in a tree structure!
Decision Tree Basics

$X$? unknown label

$Z_1 = 1$

$Z_2 = 1$
Claire
$Z_3 = 1$

$Z_2 = 0$

$Z_9 = 1$
Bill
$Z_3 = 1$

$Z_9 = 0$

queries (expensive)

$X = 12$

feature vector $Z = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \end{bmatrix}$
class label female bald brown eyes
Decision Tree – Object Detection

How do we automatically learn optimal queries to make at each node?
How do we minimise the expected number of queries?
How do we make learning/estimation efficient?
How do we handle continuous features/distributional outputs?
Decision Tree – Image Segmentation

\[ X \]

unknown label

- face 1
- face 2
- shirt 1
- background 1
- hair 1

<table>
<thead>
<tr>
<th>feature vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.9</td>
</tr>
<tr>
<td>0.3</td>
</tr>
<tr>
<td>0.2</td>
</tr>
<tr>
<td>:</td>
</tr>
</tbody>
</table>

\[ X = 12 \]
Decision Tree Basics-Notations

- A data point \( \mathbf{v} = (x_1, x_2, \ldots, x_d) \) where \( x \) denotes feature

\[ \mathbf{v} = (x_1, x_2, \ldots, x_d) \]

in CV applications: \( \mathbf{v} \) may be pixel, and \( x \) may be responses of a chosen filter-bank at that pixel!
• A data point $\mathbf{v} = (x_1, x_2, \ldots, x_d)$ where $x$ denotes feature in CV applications: $\mathbf{v}$ may be pixel, and $x$ may be responses of a chosen filter-bank at that pixel!

In theory, dimensionality of $d$ can be very large! In practice, only some of the features are being used, $d' \ll d$
Decision Tree Basics-Notations

- **Set of tests**: split functions (i.e., weak learner, test function)
Decision Tree Basics-Notations

• **Set of tests:** split functions (i.e., weak learner, test function)
• Formulate a test function at a **split node** $j$ as a function of its binary outputs:

$$h(v, \theta_j) : \mathbb{R}^d \times \mathcal{T} \rightarrow \{0, 1\}$$
**Decision Tree Basics - Notations**

- **Set of tests**: split functions (i.e., weak learner, test function)
- Formulate a test function at a split node $j$ as a function of its binary outputs:
  \[
  h(v, \theta_j) : \mathbb{R}^d \times \mathcal{T} \rightarrow \{0, 1\}
  \]

**Diagram**

- **Split node (test)**
- **$h(v, \theta_j)$**
- **Domain of Split parameters**
- **Split parameters**
**Decision Tree Basics-Notations**

- **Set of tests**: split functions (i.e., weak learner, test function)
- Formulate a test function at a **split node** \( j \) as a function of its binary outputs:

\[
h(v, \theta_j) : \mathbb{R}^d \times \mathcal{T} \rightarrow \{0, 1\}
\]

**Training point/set**: is a data point for which the attributes (we are seeking for) may be known and used to compute tree parameters:

**Example**: a training set would be a set of photos with associated indoor/outdoor labels.
Decision Tree Basics-Notations

Entire training set.
Decision Tree Basics - Notations

\[ \theta_j = \arg\max_{\theta \in \mathcal{T}} I(S_j, \theta) \]

Subset of training points reaching node j: \( S_j \)

Subsets going to the children of node j: \( S_j^L, S_j^R \)
Decision Tree Basics - Notations

\[ \theta_j = \arg \max_{\theta \in \mathcal{T}} I(S_j, \theta) \]

Subset of training points reaching node \( j \): \( S_j \)

Subsets going to the children of node \( j \): \( S_j^L \cup S_j^R \)
Decision Tree Basics - Notations

$$\theta_j = \arg\max_{\theta \in \mathcal{T}} I(S_j, \theta)$$

Subset of training points reaching node $j$: $S_j$

Subsets going to the children of node $j$: $S^L_j, S^R_j$

$$S_j = S^L_j \cup S^R_j$$

$$\emptyset = S^L_j \cap S^R_j$$
\[ \theta_j = \arg \max_{\theta \in \mathcal{T}} I(S_j, \theta) \]

At each node \( j \), we learn the function that “best” splits \( S_j \) into \( S_j^L \) and \( S_j^R \).

Off-line phase: training
On-line phase: testing
At each node $j$, we learn the function that “best” splits $S_j$ into $S_j^L$ and $S_j^R$.

Maximization of objection function $h$. 

Off-line phase: training
On-line phase: testing
Weak Learner Models

- The split functions play a crucial role both in training and testing.
Weak Learner Models

• The split functions play a crucial role both in training and testing.
• Let us start with a simple geometric parameterization for weak learner model:

\[ \theta = (\phi, \psi, \tau) \]
Weak Learner Models

- The split functions play a crucial role both in training and testing.
- Let us start with a simple geometric parameterization for weak learner model:

\[
\theta = (\phi, \psi, \tau)
\]

- Where \(\phi\) is filter (selector) function, selecting some features out of the entire vector \(v\)
Weak Learner Models

• The split functions play a crucial role both in training and testing.

• Let us start with a simple geometric parameterization for weak learner model:

\[ \theta = (\phi, \psi, \tau) \]

– Where \( \phi \) is filter (selector) function, selecting some features out of the entire vector \( \mathbf{v} \)

– \( \psi \) defines the geometric primitive used to separate data (axis-aligned hyperplane, ...)
Weak Learner Models

- The split functions play a crucial role both in training and testing.
- Let us start with a simple geometric parameterization for weak learner model:

  \[ \theta = (\phi, \psi, \tau) \]

  - Where \( \phi \) is filter (selector) function, selecting some features out of the entire vector \( \mathbf{v} \)
  - \( \psi \) defines the geometric primitive used to separate data (axis-aligned hyperplane, ...)
  - \( \tau \) captures thresholds for the inequalities used in the binary test.
Ex: Data Separation

\[ h(\mathbf{v}, \theta) = [\tau_1 > \phi(\mathbf{v}).\psi > \tau_2] \]

- where \([.\] indicator function.

\[ h(\mathbf{v}, \theta) = [\tau_1 > \phi(\mathbf{v})^T \psi \phi(\mathbf{v}) > \tau_2] \]
Recap: Classification Tree

- Hierarchical axis-aligned binary partitioning of input space
- Rule for predicting label within each block
Entropy and Information Gain

• The tree training phase is driven by the statistics of the training set.
Entropy and Information Gain

• The tree training phase is driven by the statistics of the training set.

Entropy ($H$) and information gain ($I$) are basic building blocks of the training object function!
Entropy and Information Gain

• The tree training phase is driven by the statistics of the training set.

• Information gain: reduction in uncertainty level by splitting data arriving at the node into multiple child subsets

\[ I = H(S) - \sum_{i \in \{L,R\}} \frac{|S^i|}{|S|} H(S^i) \]
**Decision forest model:** training and information gain

(for categorical, non-parametric distributions)

Information gain

\[ I(S, \theta) = H(S) - \sum_{i \in \{L,R\}} \frac{|S^i|}{|S|} H(S^i) \]

Shannon’s entropy

\[ H(S) = -\sum_{c \in C} p(c) \log(p(c)) \]

Node training

\[ \theta = \arg \max_{\theta \in T_j} I(S_j, \theta) \]
Decision forest model: training and information gain
(for continuous, parametric densities)

Information gain
\[ I(S, \theta) = H(S) - \sum_{i \in \{L,R\}} \frac{|S|^i}{|S|} H(S^i) \]

Differential entropy of Gaussian
\[ H(S) = \frac{1}{2} \log \left( (2\pi e)^d |\Lambda(S)| \right) \]

Node training
\[ \theta = \arg \max_{\theta \in \mathcal{T}_j} I(S_j, \theta) \]
Background: overfitting and underfitting

- **Underfitting**: too little model capacity
- **Overfitting**: too much model capacity

![Graph showing error vs. model capacity](image)

- **Best generalization**
- **Test set error**
- **Training set error**

(model capacity (e.g. tree depth))
Classification forest: analysing generalization

Parameters: $T=200$, $D=13$, $w. l. = \text{conic}$, predictor = prob.

Training points: 4-class spiral
Training pts: 4-class spiral, large gaps
Tr. pts: 4-class spiral, larger gaps

Testing posteriors

(3 videos in this page)
Classification forest: effect of weak learner model and randomness

Parameters: $T=400$, predictor model = prob.

Randomness: $\rho = 500$

Weak learner: axis aligned
Weak learner: oriented line
Weak learner: conic section
Classification forest: effect of weak learner model and randomness

Testing posteriors

<table>
<thead>
<tr>
<th>Weak learner: axis aligned</th>
<th>Weak learner: oriented line</th>
<th>Weak learner: conic section</th>
</tr>
</thead>
<tbody>
<tr>
<td>D=5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D=13</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Randomness: $\rho = 50$

Parameters: $T=400$ predictor model = prob.
Classification forest: effect of weak learner model and randomness

Testing posteriors

Weak learner: axis aligned | Weak learner: oriented line | Weak learner: conic section

D=5

D=13

Randomness: $\rho = 5$

Parameters: $T=400$ predictor model = prob.
Classification forest: effect of randomness

Testing posteriors

Randomness: $\rho = 1$  Randomness: $\rho = 5$  Randomness: $\rho = 50$

Weak learner: axis aligned
Where is “randomness’ come from?
Decision forest model: the randomness model

1) Bagging (randomizing the training set)

$S_0$  
The full training set

$S^t_0 \subset S_0$  
The randomly sampled subset of training data made available for the tree $t$

Forest training

Efficient training
Decision forest model: the randomness model

2) Randomized node optimization (RNO)

\[ \mathcal{T} \]  
The full set of all possible node test parameters

\[ \mathcal{T}_j \subset \mathcal{T} \]  
For each node the set of randomly sampled features

\[ \rho = |\mathcal{T}_j| \]  
Randomness control parameter.
For \( \rho = |\mathcal{T} | \) no randomness and maximum tree correlation.
For \( \rho = 1 \) max randomness and minimum tree correlation.

The effect of \( \rho \)

Small value of \( \rho \); little tree correlation.

Large value of \( \rho \); large tree correlation.
**Decision forest model:** the ensemble model

An example forest to predict continuous variables

\[ p_{t=1}(y|v) \quad p_{t=2}(y|v) \quad p_{t=3}(y|v) \quad p_{t=4}(y|v) \]

\[ p(y|v) = \frac{1}{T} \sum_{t=1}^{T} p_t(y|v) \]

\[ p(y|v) = \frac{1}{Z} \prod_{t=1}^{T} p_t(y|v) \]
Background: overfitting and underfitting

- **Underfitting**: too little model capacity
- **Overfitting**: too much model capacity

![Diagram showing the relationship between error and model capacity](image)

- **Best generalization**
- **Test set error**
- **Training set error**

Model capacity (e.g. tree depth) vs. error
Classification

- **SVM**: Support Vector Machines
  - Guarantees maximum margin separation for two-class problems

- **Boosting**: building strong classifiers as linear combination of many weak classifiers
Classification

• **SVM:** Support Vector Machines
  – Guarantees maximum margin separation for two-class problems

• **Boosting:** building strong classifiers as linear combination of many weak classifiers

Despite the success of SVMs and boosting, these techniques do not extent naturally to multiple class problems.
Classification

- **SVM**: Support Vector Machines
  - Guarantees maximum margin separation for two-class problems
- **Boosting**: building strong classifiers as linear combination of many weak classifiers

Despite the success of SVMs and boosting, these techniques do not extent naturally to multiple class problems.

- **Classification forests** work unmodified with any number of classes.
Classification

- SVM: Support Vector Machines
  - Guarantees maximum margin separation for two-class problems
- Boosting: building strong classifiers as linear combination of many weak classifiers
- Classification forests work unmodified with any number of classes.

Despite the success of SVMs and boosting, these techniques do not extent naturally to multiple class problems.

- Tracking keypoints in Video
- Human Pose estimation
- Gaze estimation
- Anatomy detection in CT Scans
- Semantic Segmentation of photos and videos
- 3D delineation of brain lesions
- ...

Despite the success of SVMs and boosting, these techniques do not extent naturally to multiple class problems.
The Algorithm

1. Given a training set $S$
2. For $i = 1$ to $k$ do:
3. Build subset $S_i$ by sampling with replacement from $S$
4. Learn tree $T_i$ from $S_i$
5. At each node:
6. Choose best split from random subset of $F$ features
7. Each tree grows to the largest extend, and no pruning
8. Make predictions according to majority vote of the set of $k$ trees.
Pseudo-Code

- `double[] ClassifyDT(node, v)`
  - if `node.IsSplitNode` then
  -   - if `node.f(v) >= node.t` then
  -     - return `ClassifyDT(node.right, v)`
  -   - else
  -     - return `ClassifyDT(node.left, v)`
  -   - end
  - else
  -   - return `node.P`
  - end
- end
Code

• Matlab
  – Piotr Dollar's toolbox
  – Andrej Karpathy's toolbox
  – M5PrimeLab by Gints Jekabsons

• R
  – Breiman and Cutler's random forests
  – Hothorn et al.'s party package with cforest function

• C/C++
  – Sherwood library
  – Regression tree package by Pierre Geurts

• Python
  – Scikit-learn

• JavaScript
  – Forestjs

• Go (golang)
  – CloudForest
Example Applications
Recap: What can decision forests do?**

- **Classification forests**
- **Regression forests**
- **Density forests**
- **Manifold forests**
- **Semi-supervised forests**
What can decision forests do? applications

**Classification forests**
- e.g. semantic segmentation

**Regression forests**
- e.g. object localization

**Density forests**
- e.g. novelty detection

**Manifold forests**
- e.g. dimensionality reduction

**Semi-supervised forests**
- e.g. semi-supervised semantic segmentation
Toy Learning Example

- Try several lines, chosen at random
- Keep line that best separates data
  - information gain
- Recurse

- feature vectors are x, y coordinates: \( \mathbf{v} = [x, y]^T \)
- split functions are lines with parameters a, b: \( f_n(\mathbf{v}) = ax + by \)
- threshold determines intercepts: \( t_n \)
- four classes: purple, blue, red, green
Toy Learning Example

- Try several lines, chosen at random
- Keep line that best separates data
  - information gain
- Recurse

- feature vectors are $x, y$ coordinates: $v = [x, y]^T$
- split functions are lines with parameters $a, b$: $f_n(v) = ax + by$
- threshold determines intercepts: $t_n$
- four classes: purple, blue, red, green
Toy Learning Example

- Try several lines, chosen at random
- Keep line that best separates data
  - information gain
- Recurse

- feature vectors are $x, y$ coordinates: $\mathbf{v} = [x, y]^T$
- split functions are lines with parameters $a, b$: $f_n(\mathbf{v}) = ax + by$
- threshold determines intercepts: $t_n$
- four classes: purple, blue, red, green
**Toy Learning Example**

- Try several lines, chosen at random

- Keep line that best separates data
  - information gain

- Recurse

- feature vectors are $x, y$ coordinates: $\mathbf{v} = [x, y]^T$
- split functions are lines with parameters $a, b$: $f_n(\mathbf{v}) = ax + by$
- threshold determines intercepts: $t_n$
- four classes: purple, blue, red, green
A Forest of Trees

- Forest is ensemble of several decision trees

\[
P(c|\mathbf{v}) = \frac{1}{T} \sum_{t=1}^{T} P_t(c|\mathbf{v})
\]

classification is

[Amit & Geman 97]
[Breiman 01]
[Lepetit et al. 06]
DEMO

Face Detection

Viola and Jones 2001: Landmark paper in Computer vision

1. A large number of Haar features.
2. Use of integral images.
3. Cascade of classifiers (Random Forest)
Segmentation by binary-class RFs

**Building**  Non-building  Error  
- Global: 74.50%  
- Average: 79.89%

**Road**  Non-road  Error  
- Global: 88.33%,  
- Average: 87.26%

**Tree**  Non-tree  Error  
- Global: 77.46%,  
- Average: 80.45%

**Car**  Non-car  Error  
- Global: 85.55 %,  
- Average: 85.24 %
Food 101 – Mining Discriminative Components with RF (ETHZ Vision Lab)

- automatically recognizing pictured dishes, and mining discriminative parts with RF

- Foodspotting.com
Food Classification Results

- Beef tartare
- Steak
- Panna cotta
- Cheese cake
- French fries
- Onion rings
- Red velvet cake
- Strawberry shortcake
- Waffles
- French onion soup
- Spring roll
- Prime rib
- Pizza
- Tiramisu
- Swaweed salad
- Clam chowder
- Risotto
- Steak
- Sashimi
- Beef tartare
- Hamburger
- French fries
- Chocolate cake
- Carrot cake
Body Part Classification (BPC) in Kinect

1 million test images, 1 day using a 1000 core cluster
Application: Efficient Human Pose Estimation from Single Depth Images (Shotton et al)

- Fast and reliable estimation of pose of the human body from images has been goal of computer vision for many years (i.e., gaming, human-computer interaction, security, tele-presence, healthcare).

From Microsoft kinect! BPC: body part classification
Application: Efficient Human Pose Estimation from Single Depth Images (Shotton et al)

- Synthetic and real data were used. When synthetic data is generated, then you have ground truth for evaluation!
Application: Efficient Human Pose Estimation from Single Depth Images (Shotton et al)

- 31 body parts were defined/labeled: LU/RU/LW/RW head, neck, L/R shoulder, LU/RU/LW/RW arm, L/R elbow, L/R wrist, L/R torso, ....
Application: Efficient Human Pose Estimation from Single Depth Images (Shotton et al)

- **Image Features**: depth comparison features are extracted for a pixel position \( \mathbf{u} = (u_x, u_y) \):

\[
\mathbf{v}(\mathbf{u}) = (z(q_1), \ldots, z(q_i), \ldots, z(q_d))
\]

with
Application: Efficient Human Pose Estimation from Single Depth Images (Shotton et al)

- **Image Features**: depth comparison features are extracted for a pixel position \( \mathbf{u} = (u_x, u_y) \):

\[
\mathbf{v}(\mathbf{u}) = (z(q_1), \ldots, z(q_i), \ldots, z(q_d))
\]

with

\[
q_i = \mathbf{u} + \frac{\delta_i}{z(\mathbf{u})}
\]

where \( z \) indicates depth, sigma denotes a 2D offset w.r.t the reference position \( \mathbf{u} \).

\( 1/z \) assures that feature response is depth invariant.
Yellow crosses indicate the reference image pixel \( u \) being classified. Red circles indicate the offset pixels.

(a) Two example features give a large depth difference response
(b) The same two features at new image locations give a much smaller response
Application: Efficient Human Pose Estimation from Single Depth Images (Shotton et al)

- Recap: $\theta = (\phi, \psi, \tau)$ weak learners
- Selector function $\phi(v) = (v_i, v_j)$ with $i, j \in \{1, \ldots, d\}$
- Fixed for all split: $\psi = (1, -1)$
- Weak learner function $h(u; \theta_n) = [f(u; \phi_n, \psi) \geq \tau_n]$

\[
f(u; \phi_n, \psi) = \phi(v(u)).\psi
\]

- $n$: denotes any node in the tree
- If $h$ evaluates to 0 -> path branches to the left, otherwise to right. Continue until reaching a leaf node.
**BPC: Body Part Classification**

- BPC predicts a discrete body part label at each pixel as an intermediate step towards predicting joint positions:

1. Store $p_l(c)$ Over body parts $c$ at each leaf $l$.
2. For a given input pixel $u$, the tree is descended to reach leaf $l = l(u)$ and the distribution is retrieved. The distributions are averaged together for all trees in the forest to give the final classification as:

$$p(c|u) = \frac{1}{T} \sum_{l \in \mathcal{L}(u)} p_l(c)$$
BPC: Body Part Classification

(a) Mean average precision vs. total number of training images (in thousands)

(b) Average precision for different body parts

- Red: Body part classification
- Blue: Offset joint regression

- Head
- Neck
- L. Shoulder
- R. Shoulder
- L. Elbow
- R. Elbow
- L. Wrist
- R. Wrist
- L. Hand
- R. Hand
- L. Knee
- R. Knee
- L. Ankle
- R. Ankle
- L. Foot
- R. Foot
- Mean AP
Body tracking in Microsoft Kinect for XBox 360

Input depth image (bg removed)  
Inferred body parts posterior

(2 videos here)
• **Training:** each bounding box includes one organ from CT images, and features are simply defined as absolute position of face of the box (6 faces, hence a vector of length 6)
Regression forests for anatomy localization in CT images

- Each voxel in the image votes for the position of the 6 box sides
- We wish to learn a set of discriminative points (landmarks, clusters) which can predict the kidney position with high confidence.

### Regression forest

<table>
<thead>
<tr>
<th>Input data point</th>
<th>Output (continuous)</th>
<th>Objective function</th>
<th>Node training</th>
<th>Predictor model</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathbf{p} = (x, y, z) )</td>
<td>( \mathbf{b} = (b^L, b^R, b^H, b^F, b^A, b^P) \in \mathbb{R}^6 )</td>
<td>( I(S_j, \theta) = \log</td>
<td>\Lambda(S_j)</td>
<td>- \sum_{i \in {L, R}} \frac{</td>
</tr>
</tbody>
</table>

#### Problem parametrization (for 1 organ only)

- Relative displacement:
  - \( \mathbf{d}(\mathbf{p}) = (d^L, d^R, d^H, d^F, d^A, d^P) \)

- Absolute displacement:
  - \( b^A = v_y - d^A \)

- Gaussian representation:
  - \( p(\mathbf{d}) = \mathcal{N}(\mathbf{d}; \mathbf{d}_\mu, \Sigma) \)

### Leaf model

- Multivariate, probabilistic constant

\[ p(\mathbf{d} | \mathbf{v}) \]
Texton Forests for Image Categorization

- Johnson, Shotton, Cipolla
- Semantic texton forest (STF) are a form of random decision forests that can be employed to produce powerful low-level codewords for computer vision.
Texton Forests for Image Categorization

- Johnson, Shotton, Cipolla
- Semantic texton forest (STF) are a form of random decision forests that can be employed to produce powerful low-level codewords for computer vision.

- (a) Test image with ground truth, (b) a set of semantic textons, (c) a rough per-pixel classification, (d) categories
Texton Forests for Image Categorization

- Johnson, Shotton, Cipolla
- Semantic texton forest (STF) are a form of random decision forests that can be employed to produce powerful low-level codewords for computer vision.

- (a) Test image with ground truth, (b) a set of semantic textons, (c) a rough per-pixel classification, (d) categories

- **Texton**: fundamental microstructures in natural images (representation of small texture patches)
(a) Semantic texton forest features. It simply includes raw image pixels within delta x delta Neighborhood (either raw values, summation of them, average, or absolute difference of pair Of pixels can be used as features).

(b) Visualization of leaf nodes from one tree (distance delta=21 pixels).
Texton Forests for Image Categorization

One texton map per tree

Ground truth categories
Advantages / Disadvantages

**Advantages**
- Improve predictive performance
- Other types of classifiers can be directly included
- Easy to implement
- No too much parameter tuning

**Disadvantages**
- The combined classifier is not so transparent (black box)
- Not a compact representation
References

• Pince, Computer Vision.
• Hasite et al., The Elements of Statistical Learning
• Criminisi and Shotton, Decision Forests, Springer.
• Tae-Kyun Kim, 2009, Univ. of Cambridge