Computer Vision & Machine Learning for Robots
Where's Waldo?
OpenCV Template Match

30 milliseconds
Visual Pattern Matching

• Household objects; e.g. mug on a table, beer in the fridge
• People, faces, pets
• Landmarks for navigation; e.g. AR markers (fiducials)
• Keypoints for Visual-SLAM
Strategies for Object Detection & Recognition

- Template Matching
- Feature Matching
- Machine Learning (learning from examples)
Detection vs Recognition

• Detection: is there a face, any face, in this picture? E.g., OpenCV Haar face detector

• Recognition: who is that face in the picture? E.g. Eigenfaces
Vision Software

• OpenCV – Linux, Windows, MacOS X, Android
  • C, C++, Python, no GUI :-:
  • Advanced vision algorithms including SIFT, SURF, face detection, machine learning
• RoboRealm – Windows Only
  • C, C++, Python, VBScript, very nice GUI
  • Many, many vision filters
  • Control for many popular robots and cameras
  • None of the patent-protected algorithms
Software Continued...

• PCL (Point Cloud Library)
  • Linux, Windows, MacOS X, Android
  • C++, no Python, no GUI
  • 3D planar segmentation, clustering
  • www.pointclouds.org

• Ecto – Linux only?
  • Combine OpenCV & PCL using Python image processing pipelines
  • Easily extendible
  • ecto.willowgarage.com
Appearances are Everything
Special Challenges for Robotics

- Live video rather than static pictures
- Changes in orientation and position (occlusions, rotations, scale, affine transformations)
- Changes in lighting and background
- Motion of object and/or camera
- Changing appearance (today Joe has on a hat and glasses)
- The world is 3D while pictures are 2D (Kinect/Xtion)
- Pose estimation (e.g. for grasping)
- Speed: at least 10-30 fps
Facing the Challenge
Courtesy of Honda/UCSD Video Database
Haar Face Detector on Video
Template Matching

317x480 pixels, 40 milliseconds
Using `pyrDown()` Twice

`under 10 milliseconds`
Template Matching
Including Scale & Rotation
230 millseconds
Template is Object Specific
Template Matching on Video
From Pixels to Features

- What is a feature? A summary statistic or descriptor computed from pixel values in an image patch, region of interest (ROI), or even the whole image.

- Colors → histograms (e.g. RGB, hue)

- Pixel intensities → gradients, binary comparisons, histograms, interest points (well defined locations)

- Connected regions → contours, segmentation

- Ideally, invariant to scale and rotations
Color Histograms

![Color Histograms](image)
OpenCV CamShift Tracking
Interest Point Detection
Harris Corners, SUSAN, FAST

\[ M = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix} \]

\[ R = \det M - k \cdot \text{Trace}^2 M \]
Interest Points = Keypoints
Another Keypoint Example
Keypoint Descriptors
Descriptors Live in Feature Space
Distance Measures: Euclidean, Manhattan, Hamming
Descriptors Available in OpenCV

- SIFT – Scale Invariant Feature Transform
- SURF – Speed Up Robust Features
- BRIEF – Binary Robust Independent Features
- ORB – Oriented FAST + Rotated BRIEF
OpenCV SURF Keypoints

58 keypoints
10 ms

474 keypoints
89 ms
SURF Keypoint/Descriptor Matching
FLANN + RANSAC, 85 ms
SURF Matching Examples
Upright = False, 180 ms
SURF is (Partially) Object Specific
Partial SURF Match
Finding Waldo with SURF
Keypoint Matching Limitations
Machine Learning & Classification

Input → Classifier → Label

Classifier

Mona Lisa

Classifier

Angelina Jolie
Building a Classifier

• Acquire training samples; e.g. Google Images, research databases (AT&T, Yale), recorded or live video
  • Single class: positive and negative samples
  • Multi-class: samples for each class
• Choose a model (e.g. SVM vs ANN)
• Train the classifier on a subset of the samples
• Test the classifier (cross-validate) on the remaining samples
• Apply classifier to new samples
Machine Learning Software

- **OpenCV** (http://opencv.willowgarage.com)
  - Linux, Windows, MacOS, Android; C, C++, Python
  - kNN, SVM, Random Trees, Neural Networks, PCA, Normal Bayes, Boosting

- **Orange** (http://orange.biolab.si)
  - Linux, Windows, MacOS X; Python
  - Visual Programming, Cross-Validation, Learner Comparison, Feature Selection, Ensembles
  - (No Neural Networks)
Learning Software Cont'd...

- Scikit-Learn (http://www.scikit-learn.org)
  - Integrated with numpy, scipy, matplotlib
  - Linux, Windows, MacOS X; Python
  - Stochastic Gradient Descent, Gaussian Mixture Models, (No Neural Networks)

- PyBrain (http://www.pybrain.org)
  - Linux; Python
  - Reinforcement Learning, Sequential Learning, Q-Learning, Recurrent Neural Networks, Belief Networks
Most Popular Classifiers

- K Nearest Neighbors (kNN)
- Support Vector Machines (SVM)
- Decision Trees (DT)
- Random Forests (Ensemble of DTs)
- Cascades (e.g. Haar face detector)
- Artificial Neural Networks (ANN)
- Adaptive Boosting (AdaBoost)
k-Nearest Neighbors (kNN) (no training required!)
Support Vector Machines (SVM)
## Decision Tree Data

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
<th>M</th>
<th>N</th>
<th>O</th>
<th>P</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Name</td>
<td>hair</td>
<td>feathers</td>
<td>eggs</td>
<td>milk</td>
<td>airborne</td>
<td>aquatic</td>
<td>predator</td>
<td>toothed</td>
<td>backbone</td>
<td>breathes</td>
<td>venomous</td>
<td>fins</td>
<td>tail</td>
<td>domestic</td>
<td>catsize</td>
<td>type</td>
</tr>
<tr>
<td>2</td>
<td>aardvark</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>mammal</td>
</tr>
<tr>
<td>3</td>
<td>antelope</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>mammal</td>
</tr>
<tr>
<td>4</td>
<td>bass</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>fish</td>
</tr>
<tr>
<td>5</td>
<td>bear</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>mammal</td>
</tr>
<tr>
<td>6</td>
<td>boar</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>mammal</td>
</tr>
<tr>
<td>7</td>
<td>buffalo</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>mammal</td>
</tr>
<tr>
<td>8</td>
<td>calf</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>mammal</td>
</tr>
<tr>
<td>9</td>
<td>carp</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>fish</td>
</tr>
<tr>
<td>10</td>
<td>catfish</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>fish</td>
</tr>
<tr>
<td>11</td>
<td>cavy</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>mammal</td>
</tr>
<tr>
<td>12</td>
<td>cheetah</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>mammal</td>
</tr>
<tr>
<td>13</td>
<td>chicken</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>chup</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>fish</td>
</tr>
<tr>
<td>15</td>
<td>clam</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>invertebrate</td>
</tr>
<tr>
<td>16</td>
<td>crab</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>crayfish</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>crow</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>bird</td>
</tr>
<tr>
<td>19</td>
<td>deer</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>mammal</td>
</tr>
<tr>
<td>20</td>
<td>dogfish</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>fish</td>
</tr>
<tr>
<td>21</td>
<td>dolphin</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>mammal</td>
</tr>
<tr>
<td>22</td>
<td>dove</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>bird</td>
</tr>
<tr>
<td>23</td>
<td>duck</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>bird</td>
</tr>
<tr>
<td>24</td>
<td>elephant</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>mammal</td>
</tr>
</tbody>
</table>
Decision Tree Classifier
Random Forests

\[ P(c|v) = \sum_{t=1}^{T} P_t(c|v) \]
Artificial Neural Networks (ANN)
Prebuilt Classifiers in OpenCV

Face & People Detectors

- Haar Face Detector
- HOG Person Detector
Automated Color Naming
Hue Histograms
RGB → HSV → Hue
Preparing the Color Data

• Collect color images, say 20 images per color
• Store images in folders named after each color
• Run training script:
  • For each color folder
    • Read image
    • Convert to HSV
    • Compute Hue histogram
    • Add color label and histogram values to tab-delimited data file
Test Multiple Learners with Orange

- **k Nearest Neighbours**
- **Classification Tree**
- **SVM**
- **Random Forest**

**Sampling**
- Cross-validation
  - Number of folds: 5
- Leave-one-out
- Random sampling
  - Repeat train/test: 10
  - Relative training set size: 70%

**Evaluation Results**

<table>
<thead>
<tr>
<th>Method</th>
<th>CA</th>
</tr>
</thead>
<tbody>
<tr>
<td>kNN</td>
<td>0.8732</td>
</tr>
<tr>
<td>SVM</td>
<td>0.9206</td>
</tr>
<tr>
<td>Random Forest</td>
<td>0.8175</td>
</tr>
<tr>
<td>Classification Tree</td>
<td>0.7225</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>0.8412</td>
</tr>
</tbody>
</table>
What's in a Face?

What are face features?

- Eyes, ears, nose, mouth?
- Outline, hair color?
- Geometry?
- Template?
- SURF descriptors?
- Eigenfaces?
- Fisher Faces?
The AT&T / ORL Face Database
Using Eigenfaces for Face Recognition

• Convert each $W \times H$ image to a $(W \times H)$-d vector by concatenating the rows; e.g. 100x100 image $\rightarrow$ 10,000-d vector

• Run a Principle Components Analysis (PCA) on all training vectors $\rightarrow$ returns eigenvectors

• Keep N eigenvectors with largest eigenvalues; e.g. $N = 64$

• Project sample images onto this feature space: converts each 10,000 element vector to a 64 element vector

• Build a classifier using these feature vectors
Eigenfaces
Average and Reconstructed Faces using PCA (64 eigenvectors)
Eigenface Performance on AT&T faces
40 people, 10 images per person
(ANN = 94% correct)
The U Sheffield Faces
20 people, 20-40 images per person

Test Learners

Sampling

- Cross-validation
  - Number of folds: 5
- Leave-one-out
- Random sampling
  - Repeat train/test: 10
  - Relative training set size:

Evaluation Results

<table>
<thead>
<tr>
<th>Method</th>
<th>CA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 kNN</td>
<td>0.9878</td>
</tr>
<tr>
<td>2 SVM</td>
<td>0.9913</td>
</tr>
<tr>
<td>3 Random Forest</td>
<td>0.8557</td>
</tr>
<tr>
<td>4 Classification Tree</td>
<td>0.7826</td>
</tr>
<tr>
<td>5 Naive Bayes</td>
<td>0.9496</td>
</tr>
</tbody>
</table>
Labeled Faces in the Wild
Eigenface Failure...
Alternatives to Eigenfaces

Your Contribution Goes Here :-) 

• PCL: Viewpoint Feature Histograms (VFH)
• SIFT/SURF + PCA
• Biometrics
• 3D Modeling
Real Time Classifier Learning

• Select an object any way you can
  • manual selection
  • existing detector (e.g. Haar face detector)
  • motion detection
• Track keypoints using Optical Flow
• Grab positive and negative samples as you go
• Build a custom classifier from current samples
• Re-detect object using classifier
• Store custom classifier for later detection
Face Tracking using Optical Flow

Video courtesy of Honda/UCSD Video Database

Pi Face Tracker Video
http://youtu.be/Yw_zkLaZNsQ
Predator Algorithm (OpenTLD)
Zdenek Kalal, Jan 2011

Predator Track-Learn-Detect (TLD)
http://youtu.be/1GhNXHCQGzsM
Thanks for Listening!