Computer Vision Systems Programming VO

Object Recognition

Christopher Pramerdorfer

Computer Vision Lab, Vienna University of Technology

Topics

Selection of recognition problems

- Detecting specific rigid objects
- Efficient face detection
- Image classification

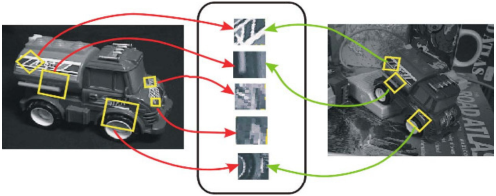


Image from Grauman and Leibe 2011

Taxonomy of Object Recognition

Instance vs. category recognition

- Instance : my face, the Eiffel tower
- Category : faces, buildings, people

Classification vs. detection

- Classification : predict class of main object in image
- Detection : multiple objects, possibly of different class

Taxonomy of Object Recognition
Classification

Blue Mosque (instance), mosque/building (category)

Taxonomy of Object Recognition
Detection

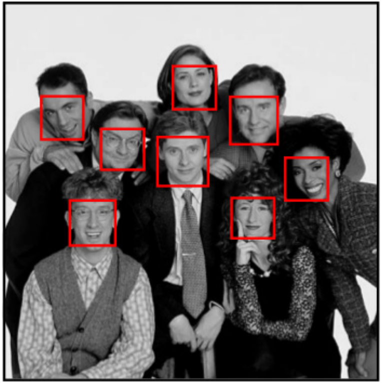


Image from Prince 2012

Challenges

Instances of same category can look very differently

- Illumination, pose, viewpoint, occlusions, background

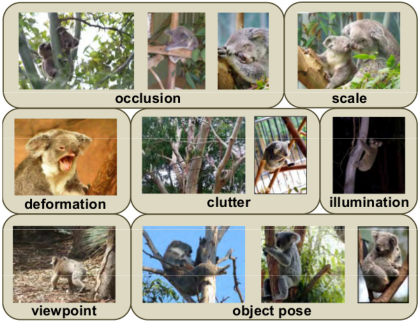


Image from Grauman and Leibe 2011

Detecting Specific Rigid Objects

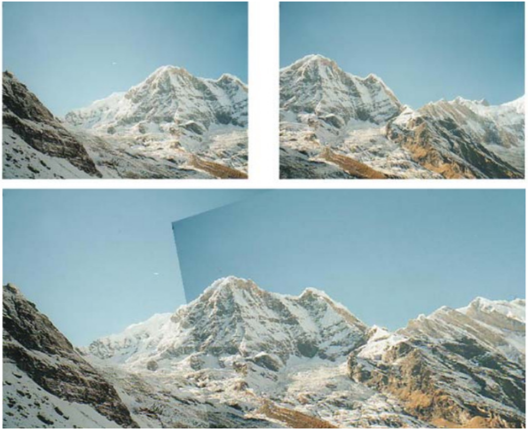


Image adapted from Brown and Lowe 2007

Detecting Specific Rigid Objects

Often accomplished via **local feature matching**

Given an image of the object and a search image

- Compute local features in both images
- Match features between images to find correspondences
- Perform geometric verification

Local features form a sparse object representation

- Features capture characteristic structure
- Allow for efficient matching between images
- Representation robust to occlusion and clutter

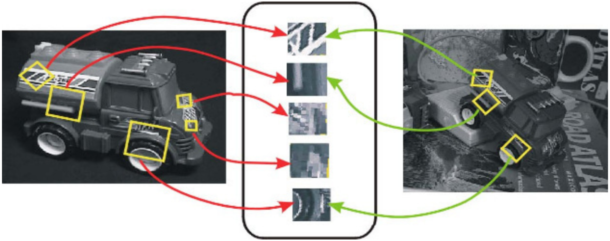


Image from Grauman and Leibe 2011

Many different feature extractors available

- SIFT, SURF, BRISK, FREAK, ...
- OpenCV includes implementations

Guidelines on feature selection

- Features should be invariant to expected transformations
- And only to these transformations
- Extraction and matching speeds differ
- Performance often quite similar, but better test

Features are n-dimensional vectors

- Perform nearest neighbor matching in this feature space

Popular matching strategy

- Given feature $\mathbf{x}$ in first image
- Find two nearest neighbors $\mathbf{y}_1, \mathbf{y}_2$ from second image
- $\{\mathbf{x}, \mathbf{y}_1\}$ correspond if $\|\mathbf{x} - \mathbf{y}_1\| / \|\mathbf{x} - \mathbf{y}_2\| < 0.8$

Detecting Specific Rigid Objects
Geometric Verification

Assume that the object in question is planar

- Images of planar objects are related by a homography
- Also applies to local feature locations

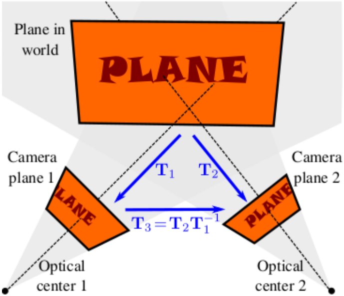


Image from Prince 2012

Relation allows for detecting erroneous correspondences

- Estimate homography T from correspondences
- Discard correspondences for which $\|\mathbf{x} - T(\mathbf{y}_1)\| > t$

Verification also possible for nonplanar scenes

- Epipolar geometry constraints (previous lecture)

Detecting Specific Rigid Objects
OpenCV Code Example

```
// read images (SIFT expects grayscale images)
cv::Mat object = cv::imread("object.jpg", cv::IMREAD_GRAYSCALE);
cv::Mat search = cv::imread("search.jpg", cv::IMREAD_GRAYSCALE);

cv::SIFT sift; // using default arguments here

// compute keypoints
std::vector<cv::KeyPoint> kobject, ksearch;
sift.detect(object, kobject); sift.detect(search, ksearch);

// compute descriptors (our x, y)
cv::Mat dobject, dsearch;
sift.compute(object, kobject, dobject); sift.compute(search, ksearch, dsearch);
```

Detecting Specific Rigid Objects
OpenCV Code Example

```
// find y1,y2 for each x
cv::FlannBasedMatcher matcher; // fast nearest neighbor search
std::vector<std::vector<cv::DMatch> > kMatches;
matcher.knnMatch(dobject, dsearch, kMatches, 2);

// select good matches (see above slide)
std::vector<cv::DMatch> matches;
for(const std::vector<cv::DMatch>& match : kMatches)
    if(match[0].distance < match[1].distance * 0.8)
        matches.push_back(match[0]); // (x,y1)
```

Detecting Specific Rigid Objects

OpenCV Code Example

```
// collect feature locations of correspondences
std::vector<cv::Point2f> pobject, psearch;
for(const cv::DMatch& match : matches) {
    pobject.push_back(kobject.at(match.queryIdx).pt);
    psearch.push_back(ksearch.at(match.trainIdx).pt);
}

// estimate homography using RANSAC for robustness
cv::Mat inliers; // contains indices of valid correspondences
cv::Mat homography = cv::findHomography(pobject, psearch, CV_RANSAC, 3, inliers);
```

Detecting Specific Rigid Objects
Applications – Object Detection

Detection and 2D pose estimation

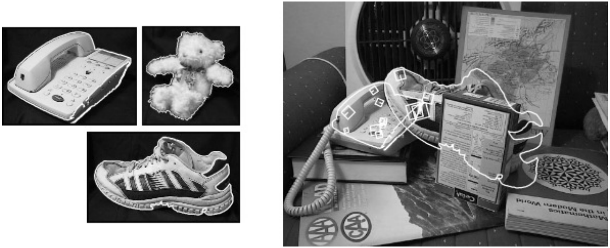
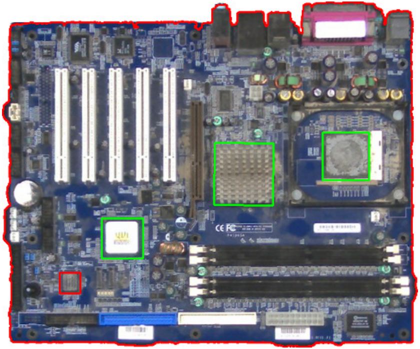


Image adapted from Lowe 2004

Detecting Specific Rigid Objects
Applications – Object Detection

Industrial applications like PCB recycling



Detecting Specific Rigid Objects
Applications – Panorama Stitching

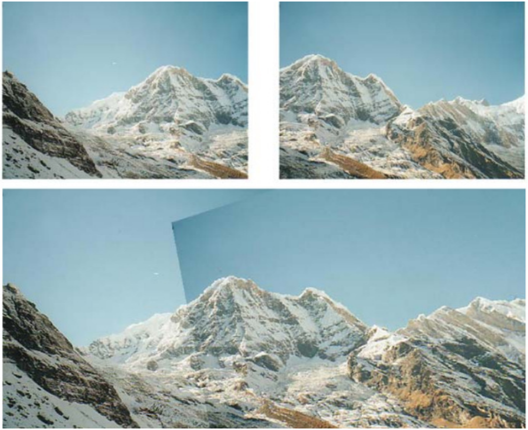


Image adapted from Brown and Lowe 2007

Efficient Face Detection

Image from olympus-europa.com

Efficient Face Detection

Face detection has many applications, such as

- Smart cameras (autofocus on faces)
- Security (preprocessing step to face recognition)
- Augmented reality & gaming

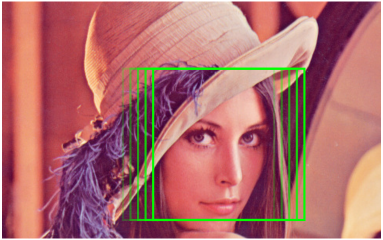
We focus on the popular method from [Viola and Jones 2001]

- Fast enough to run on e.g. cameras

Efficient Face Detection

Sliding window approach

- Slide window over image
- Infer label $w \in \{0, 1\}$ based on measurements $\mathbf{x}$
- Perform non-maximum suppression of confidence scores



Efficient Face Detection

Simple features – difference d in rectangular subwindow of $\mathbf{x}$

- Can be computed in constant time using integral images
- Limited number of such features, $\{f_t\}_{t=1}^{T}$

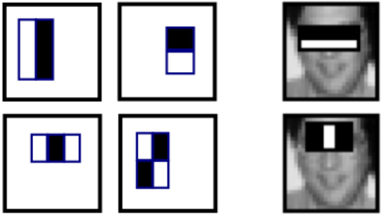


Image adapted from Prince 2012

Efficient Face Detection

Classification using a **boosted cascade**

- Cascade of $K \leq T$ weak but fast classifiers $c_k = f_k > t_k$
- Early rejection of non-face windows for speed
- Final classifier is $C(\mathbf{x}) = \text{sign}(\theta_0 + \sum_k \theta_k c_k)$

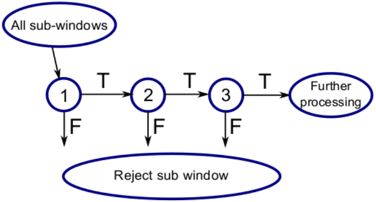


Image from Prince 2012

Efficient Face Detection

Subset of K best classifiers, their order, and $\boldsymbol{\theta}$ are learned

By means of **boosting** : for each $k = 1 \cdots K$

- Find best classifier according to training set, add to C
- Raise weights of samples misclassified by current C

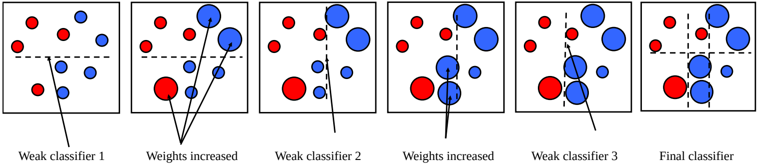


Image from Szeliski 2010

Efficient Face Detection
OpenCV Code Example

Detect faces using a pretrained cascade

OpenCV also supports cascade training

```python
# detect faces using a pretrained cascade
image = cv2.imread('faces.jpg', cv2.IMREAD_GRAYSCALE)
cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
faces = cascade.detectMultiScale(image) # should tune parameters
```

Image Classification

Goal is to predict the class of the main object in an image

We consider methods based on visual words

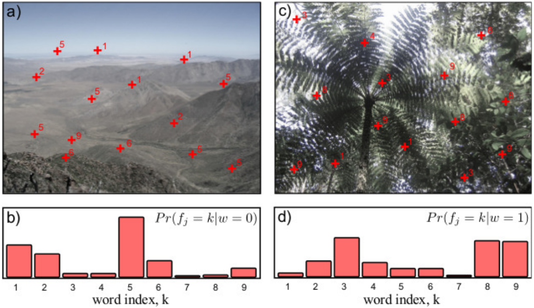


Image from Prince 2012

Image Classification

Idea is to represent an image as a collection of **visual words**

- Images can be compared based on visual word distribution

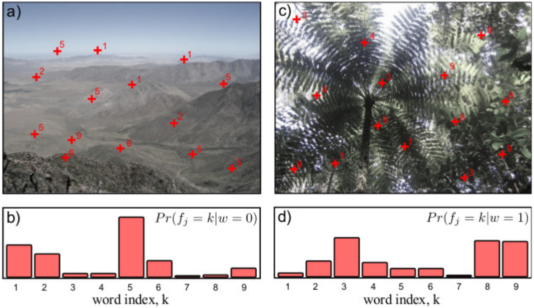


Image from Prince 2012

Image Classification

Visual words are learned from an image collection

- Compute local features for all images
- Cluster these vectors into k clusters using k-means
- k cluster means represent visual words

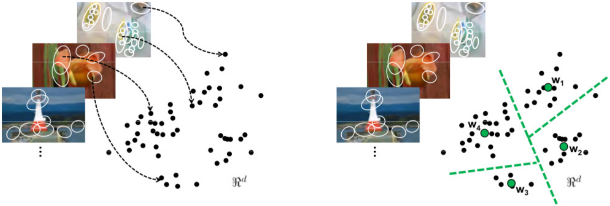
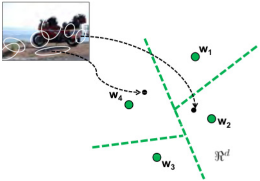


Image from Grauman and Leibe 2011

Image Classification

Visual word distribution $\mathbf{x} \in \mathbb{R}^k$ of image obtained by

- Computing local features for current image
- Assigning each feature to closest visual word
- Summing up the assignment counts for each visual word



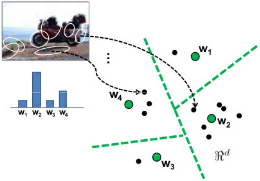


Image from Grauman and Leibe 2011

Image Classification

Prediction of class w from $\mathbf{x}$ using e.g. SVM

- Classifier learned using training samples $\{(\mathbf{x}, w)\}$

Image Classification

The above approach is called **bag of words** model

- Many improvements to this methods exist
- Popular and can work well, but not state of the art

We will look at the current state of the art next time

Summary

Object recognition is key to many successful applications

We have discussed three popular cases

- Detecting specific rigid objects
- Efficient face detection
- Image classification from visual words

Bibliography I

Brown, Matthew and David G Lowe (2007). **Automatic panoramic image stitching using invariant features**.

Grauman, Kristen and Bastian Leibe (2011). **Visual object recognition**. Morgan & Claypool.

Lowe, David G (2004). **Distinctive image features from scale-invariant keypoints**.

Prince, S.J.D. (2012). **Computer Vision: Models Learning and Inference**. Cambridge University Press.

Szeliski, Richard (2010). **Computer vision: algorithms and applications**. Springer.

Bibliography II

Viola, Paul and Michael Jones (2001). **Rapid object detection using a boosted cascade of simple features**.