Computer Vision Systems Programming VO

Specific Object Recognition

Christopher Pramerdorfer

Computer Vision Lab, Vienna University of Technology

Topics

Introduction to object recognition

Specific object recognition

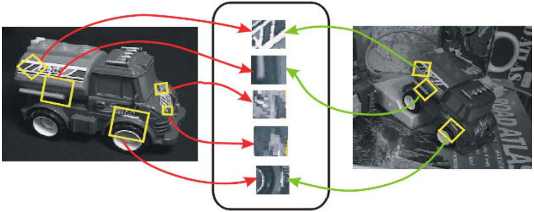


Image from Grauman and Leibe 2011

Object Recognition

Fundamental problem in Computer Vision

Many applications

- Panorama stitching, 3D reconstruction
- HCI and surveillance (face recognition)
- Image understanding (recall Fei-Fei Li's TED talk)

Instance recognition (specific object recognition)

- Recognize a specific, uniquely looking object
- Face of a certain person, the Eiffel tower

Object category recognition

- Recognize objects of a certain category
- Human faces, buildings

Object Recognition
Taxonomy – Instance vs. Category

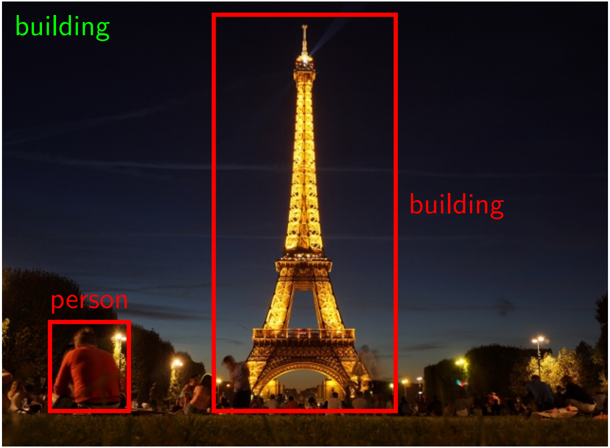
Object classification

- Recognize main object in image
- Location and other objects not relevant

Object detection

- Recognize multiple objects, possibly of different category

Object Recognition
Taxonomy – Classification vs. Detection



Object Recognition
Challenges

Instances of same category can look very differently

- Illumination, pose, viewpoint, occlusions, background

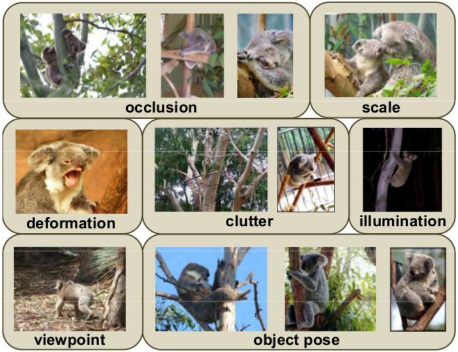


Image from Grauman and Leibe 2011

Planar Rigid Object Detection

We want to detect specific rigid planar objects

- Like markers, books
- Comparatively easy problem

Challenges

- Unknown object pose and scale
- Varying illumination
- Partial occlusions

Planar Rigid Object Detection

Application: Marker-Based AR

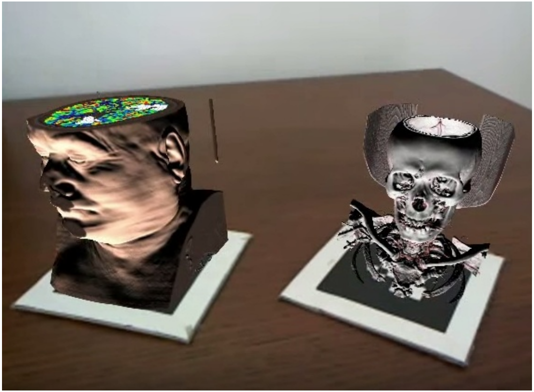


Image from `youtube.com`

Assuming that object is far away (on *plane at infinity*)

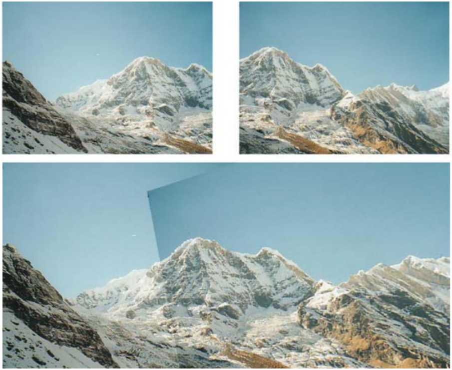


Image adapted from Brown and Lowe 2007

Our problem formulation is

- Given a pixel location in an image
- Predict location on object surface (world or other image)

So we know how to select $\mathbf{x}$ and $\mathbf{w}$

- $\mathbf{x} = (x, y)$: pixel location in query image
- $\mathbf{w} = (u, v)$: corresponding location on object surface

As $\mathbf{w} \in \mathbb{R}^2$ this is a regression problem

Images of planar objects are always related by a homography $\mathbf{\Phi}$

- 3×3 matrix mapping between corresponding points

In homogeneous coordinates this means that

$$\lambda \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \mathbf{\Phi} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Planar Rigid Object Detection
Model Selection

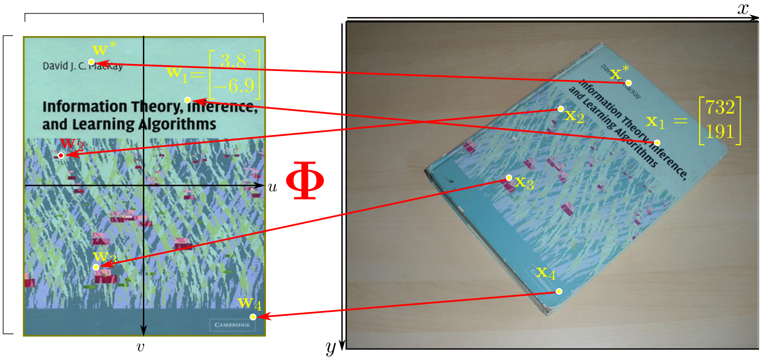


Image adapted from Prince 2012

The model of choice is thus (disregarding noise)

$$\mathbf{w} = \Gamma(\mathbf{x}) = \begin{pmatrix} u \\ v \end{pmatrix} \quad , \quad \lambda \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \mathbf{\Phi} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

We again learn parameters $\boldsymbol{\theta}$ from samples $\{(\mathbf{x}_i, \mathbf{w}_i)\}_{i=1}^n$

- $\boldsymbol{\theta}$ contains 9 parameters comprising $\boldsymbol{\Phi}$

Usually no exact solution because of noisy $\mathbf{x}_i$

- Formulate as a least squares problem instead

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \left[\sum_{i=1}^{n} (\mathbf{w}_i - \Gamma(\mathbf{x}_i))^\top (\mathbf{w}_i - \Gamma(\mathbf{x}_i)) \right]$$

This least squares approach is optimal

- If noise is distributed normally with spherical covariance

Finding $\hat{\boldsymbol{\theta}}$ is a nonlinear optimization problem

- Solvable using any general nonlinear least squares solver
- OpenCV has an own function `findHomography`

$\mathbf{\Phi}$ is a 2D transformation

Where is the object in the world?

- This is called pose estimation
- Required e.g. for marker-based AR like above

This information can be extracted from $\mathbf{\Phi}$

- If we know the intrinsic camera parameters (see below)

How can we compute $\{(\mathbf{x}_i, \mathbf{w}_i)\}_{i=1}^{n}$ automatically?

- We first select $\mathbf{w}_i$, then search corresponding $\mathbf{x}_i$

$\mathbf{w}_i$ can be selected

- Manually (e.g. specific corners on markers)
- Automatically (e.g. SIFT)

We opt for the second approach and use SIFT (or similar)

- Features invariant to rotation, scale, illumination
- Robust to affine transformations

Approach

- Compute keypoints and descriptors in both images
- Match descriptors (e.g. nearest neighbor association)
- Use keypoint locations of matches as $\{(\mathbf{x}_i, \mathbf{w}_i)\}_{i=1}^{n}$

There will likely be incorrect matches

- Would greatly impact the least squares solution
- Hence we use a robust alternative like RANSAC

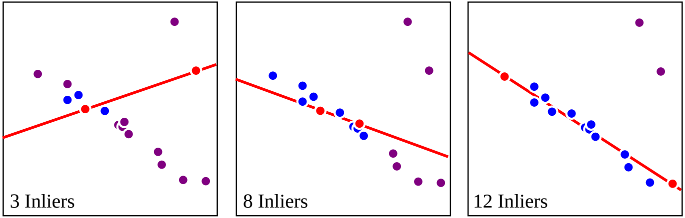


Image adapted from Prince 2012

Planar Rigid Object Detection
Obtaining Point Correspondences Using OpenCV

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

Planar Rigid Object Detection
Obtaining Point Correspondences Using OpenCV

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

// keep match (x,w) if x is clearly more similar than x'
// this is a popular matching strategy
std::vector<cv::DMatch> matches;
for(const std::vector<cv::DMatch>& match : kMatches)
    if(match[0].distance < match[1].distance * 0.8) // x, x'
        matches.push_back(match[0]); // (x,w)
```

```
// collect feature locations of correspondences from before
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, 2, inliers);
```

Nonplanar Rigid Object Detection
Application: Object Detection

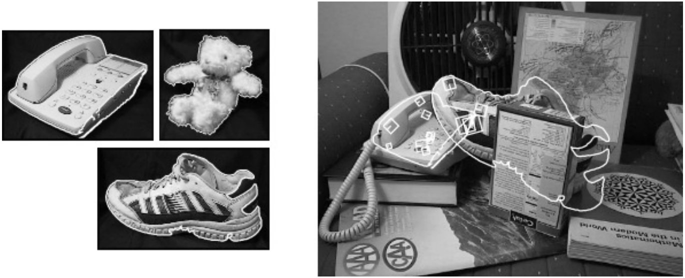
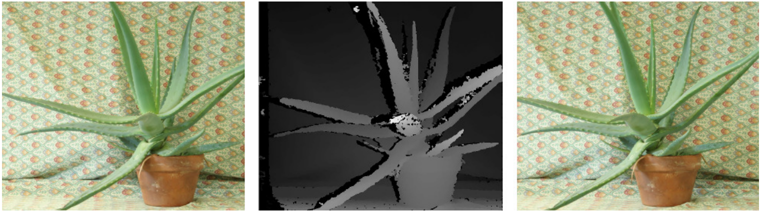


Image adapted from Lowe 2004

Nonplanar Rigid Object Detection
Application: Stereo



Nonplanar Rigid Object Detection
Application: Structure from Motion

Image from https://www.youtube.com/watch?v=sQegEro5Bfo

Nonplanar Rigid Object Detection

Selecting $\mathbf{x}$ and $\mathbf{w}$

We use the same problem formulation as before

- Given a pixel location in an image $\mathbf{x}$
- Predict location on object surface $\mathbf{w}$

As object is no longer planar, we have $\mathbf{w} = (u, v, w)$

- $\mathbf{w}$ is a point in world coordinates
- From $\mathbf{w}$ we can recover corresponding $\mathbf{x}$ in another image

In this case $\mathbf{x}$ and $\mathbf{w}$ are related by the pinhole camera model

- Generalization of the homography model

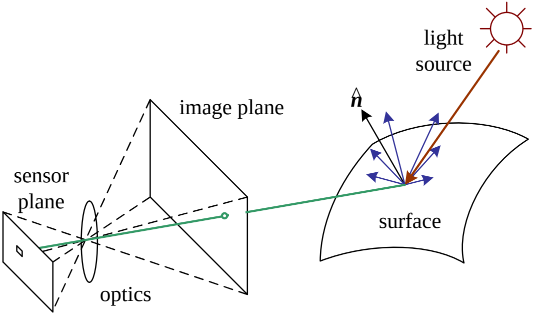


Image from Szeliski 2010

Nonplanar Rigid Object Detection
Model Selection

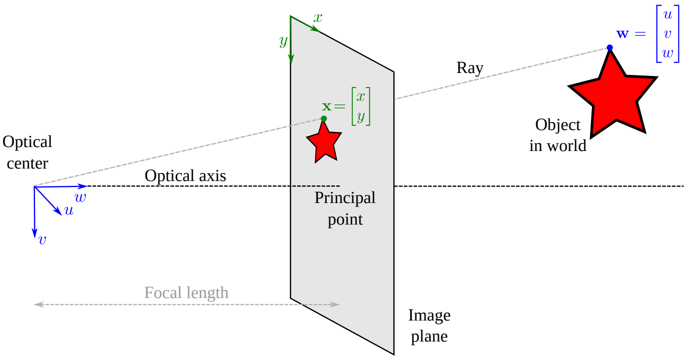


Image adapted from Prince 2012

We see that in homogeneous coordinates

$$\lambda \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} u \\ v \\ w \\ 1 \end{pmatrix}$$

With the intrinsic parameters

- f : focal length in pixels
- p_x, p_y : principal point coordinates

World and camera coordinate systems generally differ

- Transform $\mathbf{w}$ to camera coordinates before projection

$$\mathbf{w}' = \begin{pmatrix} u' \\ v' \\ w' \end{pmatrix} = \begin{pmatrix} \omega_{11} & \omega_{12} & \omega_{13} \\ \omega_{21} & \omega_{22} & \omega_{23} \\ \omega_{31} & \omega_{32} & \omega_{33} \end{pmatrix} \begin{pmatrix} u \\ v \\ w \end{pmatrix} + \begin{pmatrix} \tau_u \\ \tau_v \\ \tau_w \end{pmatrix}$$

The extrinsic parameters τ and ω encode translation and rotation

We combine this for the full pinhole camera model

- Standard camera model in Computer Vision

$$\lambda \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \omega_{11} & \omega_{12} & \omega_{13} & \tau_u \\ \omega_{21} & \omega_{22} & \omega_{23} & \tau_v \\ \omega_{31} & \omega_{32} & \omega_{33} & \tau_w \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u \\ v \\ w \\ 1 \end{pmatrix}$$

We again learn the parameters from samples $\{(\mathbf{x}_i, \mathbf{w}_i)\}_{i=1}^{n}$

- Using RANSAC and least squares like before
- OpenCV has own functions `solvePnP`, `solvePnPRansac`

How can we again compute $\{(\mathbf{x}_i, \mathbf{w}_i)\}_{i=1}^{n}$ automatically?

- Assume we have two images of the object
- Let $\mathbf{x}_1, \mathbf{x}_2$ be the projections of $\mathbf{w}$ in these images

Assume we know the camera parameters (Slide 35) but not $\mathbf{w}$

- We can compute $\mathbf{w}$ from $\mathbf{x}_1$ ($\mathbf{x}_2$) only up to scale
- Results in a epipolar line in the other image
- $\mathbf{x}_2$ ($\mathbf{x}_1$) must lie on this line, so we search only there

Nonplanar Rigid Object Detection
Obtaining Point Correspondences

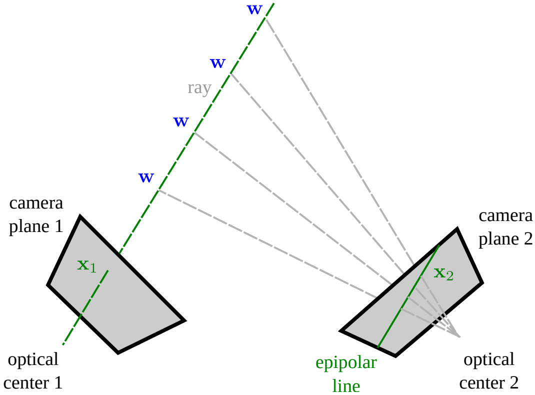


Image adapted from Prince 2012

Nonplanar Rigid Object Detection
Obtaining Point Correspondences

Points on this line fulfill $\tilde{\mathbf{x}}_2^\top \mathbf{E}\, \tilde{\mathbf{x}}_1 = 0$

- Here $\tilde{\mathbf{x}}_i$ is $\mathbf{x}_i$ in homogeneous coordinates
- $\mathbf{E}$ is called essential matrix (encodes extrinsic relationship)

We can now proceed similar to before

- Select points $\{\mathbf{x}_1^i\}_{i=1}^n$ (e.g. every pixel, keypoint locations)
- For each $\mathbf{x}_1^i$ search for $\mathbf{x}_2^i$ on the epipolar line of $\mathbf{x}_1^i$
- If we have $\mathbf{x}_2^i$ we can compute $\mathbf{w}_i$ via triangulation

Nonplanar Rigid Object Detection

Obtaining Point Correspondences

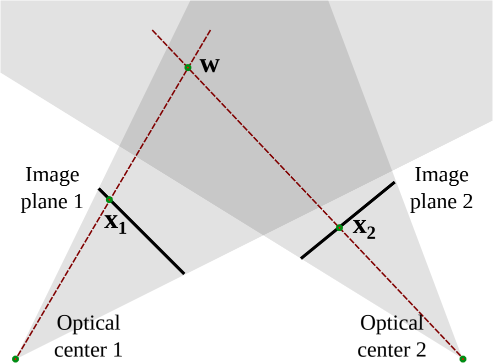


Image adapted from Prince 2012

This is the typical stereo pipeline for 3D reconstruction

```
# dense stereo in OpenCV (Python), left and right are rectified
imgL = cv2.pyrDown(cv2.imread('left.jpg'))
imgR = cv2.pyrDown(cv2.imread('right.jpg'))
stereo = cv2.StereoSGBM(...) # args depend on images
disparity = stereo.compute(imgL, imgR)
```

But what if we do not know the camera parameters?

- Why we wanted to compute $\{(\mathbf{x}_i, \mathbf{w}_i)\}_{i=1}^{n}$ in the first place

If we know only the intrinsics, we can estimate the extrinsics

- Estimate $\mathbf{E}$ from correspondences, derive extrinsics

If not, we can estimate the fundamental matrix $\mathbf{F}$

- Corresponding points must fulfill $\tilde{\mathbf{x}}_2^\top \mathbf{F}\, \tilde{\mathbf{x}}_1 = 0$
- Metric reconstruction of $\mathbf{w}$ no longer possible

Nonplanar Rigid Object Detection

Remarks

With 3 or more images we can estimate all parameters and $\mathbf{w}$

- See above structure from motion example

Camera parameters and $\mathbf{w}$ influence each other

- We want to jointly optimize all parameters and all $\mathbf{w}$
- This technique is called bundle adjustment

Nonplanar Rigid Object Detection

Remarks

We have treated 3D reconstruction as an object detection problem

- Somewhat unorthodox but fits in nicely

Lecture 183.129 has more 3D vision

This approach to object detection is powerful but has limitations

- Slow, less robust if images are very dissimilar (Slide 27)
- Alternatives more popular if "coarse" detection suffices

Nonplanar Nonrigid Object Detection

What about *general* specific object recognition?

- Living things are neither planar nor rigid

Image from attackofthecute.com

Classic CV problem : recognize depicted person

ID: Christopher Pramerdorfer

Constellation models can be used for this

- Describe object as set of parts and their spatial relations

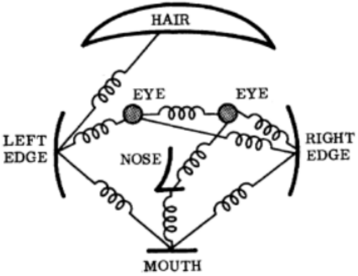


Image from Fischler and Elschlager 1973

We describe an object by M parts

- So $\mathbf{w} = (\mathbf{w}_1, \ldots, \mathbf{w}_M)$ with $\mathbf{w}_m = (u_m, v_m)$
- $\mathbf{w}_m$ is (relative) location of part m in input image

We extract a feature vector $\mathbf{x}_m$ suitable for describing part m

- Gradient histograms often work well (SIFT, HOG)
- Features can vary between parts

We then specify individual models

- One for each part, $\Pr(\mathbf{w}_m|\mathbf{x}_m)$
- One for each pair of related parts, $\Pr(\mathbf{w}_j|\mathbf{w}_k)$

And combine this to

$$\Pr(\mathbf{w}_1,\ldots,\mathbf{w}_M|\mathbf{x}_1,\ldots,\mathbf{x}_M) = \prod_{m=1}^{M}\Pr(\mathbf{w}_m|\mathbf{x}_m)\prod_{j,k}\Pr(\mathbf{w}_j|\mathbf{w}_k)$$

We again learn the model parameters $\boldsymbol{\theta}$ from training data

- E.g. using maximum likelihood like before
- Parameters and learning algorithms depend on chosen models

Often we care only about relative position between parts

- "Fix" $\mathbf{w}_m$, use relative positions $\mathbf{w}_j, \mathbf{w}_k$

We seek the $\mathbf{w}$ that maximizes $\Pr(\mathbf{w}_1, \ldots, \mathbf{w}_M | \mathbf{x}_1, \ldots, \mathbf{x}_M)$

Doing so is NP-hard, so we draw samples instead

- For each part m find $\{\mathbf{w}_m\}$ for which $\Pr(\mathbf{w}_m | \mathbf{x}_m) > t_m$
- Evaluate model for all combinations $\{\mathbf{w}_m\}$ between parts
- Pick the most likely part configuration $\mathbf{w}$

Approximation : might miss the most likely configuration

More powerful alternatives, especially if we have lots of data

- Next lecture

Bibliography I

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

Fischler, Martin A and Robert A Elschlager (1973). *The representation and matching of pictorial structures*. IEEE Transactions on Computers.

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.

Bibliography II

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