Computer Vision

Lecture Notes

Stand: February 24, 2020

Tobias Zumsteg
tzumsteg@ethz.ch

Resources:

- Lecture Slides by Marc Pollefeys, Luc Van Gool, Vittorio Ferrari
- Tutorial Notes by Marc Pollefeys
- Book Computer Vision: Algorithms and Applications by Richard Szeliski

DLT explained (with Gold Standard algorithm):
http://bardsley.org.uk/wp-content/uploads/2007/02/3d-reconstruction-using-the-direct-linear-transform.pdf

Contents

1. Projective Geometry

1.1. Coordinates

1.1.1. Crossproduct

$$\tilde{\boldsymbol{x}}_1 \times \tilde{\boldsymbol{x}}_2 = [\tilde{\boldsymbol{x}}_1]_\times \tilde{\boldsymbol{x}}_2 = \begin{bmatrix} 0 & z & -y \\ -z & 0 & x \\ y & -x & 0 \end{bmatrix} \tilde{\boldsymbol{x}}_2$$

1.1.2. 2D, 3D points

For 2D a small letter is used, a capital letter for 3D, a bold letter stands for a vector, while a normal one is a single entry from a vector

Inhomogeneous coordinates (marked with plain letter):

$$\boldsymbol{x} = (x, y)^\top \qquad \boldsymbol{X} = (x, y, z)^\top$$

Homogeneous coordinates (with a tilde on top):

$$\tilde{\boldsymbol{x}} = (x, y, w)^\top \qquad \tilde{\boldsymbol{X}} = (x, y, z, w)^\top$$

also called " augmented coordinates $\bar{x}$ " when $w = 1$.

Scale is not important for incidence relation.

Ideal point / point at infinity:

$$\tilde{\boldsymbol{x}} = (x, y, 0)^\top \qquad \tilde{\boldsymbol{X}} = (x, y, z, 0)^\top$$

Intersection of two lines:

$$\tilde{\boldsymbol{x}} = \tilde{\boldsymbol{l}}_1 \times \tilde{\boldsymbol{l}}_2$$

3D Point form three planes:

$$\begin{bmatrix} \pi^{(1)\top} \\ \pi^{(2)\top} \\ \pi^{(3)\top} \end{bmatrix} \tilde{\boldsymbol{X}} = 0$$

1.1.3. 2D lines

$$\boldsymbol{l} = (a, b, c)$$

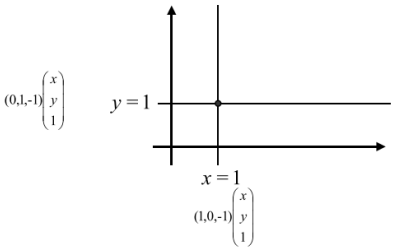
The point x lies on the line l if and only if

$$\boldsymbol{l}^\top \tilde{\boldsymbol{x}} = ax + by + c = 0$$

Line at infinity:

$$\boldsymbol{l}_\infty = (0, 0, 1)^\top$$

Horizontal and vertical lines:



Line joining two points:

$$\boldsymbol{l} = \tilde{\boldsymbol{x}}_1 \times \tilde{\boldsymbol{x}}_2$$

1.1.4. 3D Planes

$$\pi = (\pi_1, \pi_2, \pi_3, \pi_4)^\top$$

The point $\tilde{\boldsymbol{X}}$ lies on the plane π if and only if

$$\pi^T \tilde{\boldsymbol{X}} = \pi_1 x + \pi_2 y + \pi_3 z + \pi_4 w = 0$$

Planes from points

$$\begin{bmatrix} \tilde{\boldsymbol{X}}^{(1)\top} \\ \tilde{\boldsymbol{X}}^{(2)\top} \\ \tilde{\boldsymbol{X}}^{(3)\top} \end{bmatrix} \pi = 0$$

1.1.5. Conic

Inhomogenious:

$$ax^2 + bxy + cy^2 + dx + ey + f = 0$$

5 Degrees of freedom

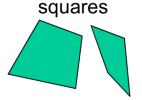
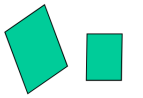
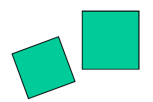
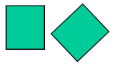
1.2. Transformation

1.2.1. 2D Transformation

- Rotation+translation/Euclidean
- Scaled Rotation/Similarity/Metric - This transformation preserves angles between lines and planes(Orthogonal views of the planar at all times)
- Affine - Parallel lines and planes remain parallel under affine transformations (general camera motion relative to the planar shape, but always keeping sufficient distance from the planar shape)
- Projectivity/Collineation/Projective transformation/Homography (all cameras motions are allowed, also when moving close to the planar shape)

$$\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{pmatrix} x \\ y \\ w \end{pmatrix}$$

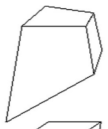
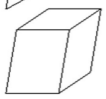
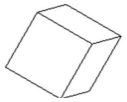
Point transformation: $\tilde{\boldsymbol{x}}' = H\tilde{\boldsymbol{x}}$
Line transformation: $\boldsymbol{l}' = H^{-\top}\boldsymbol{l}$

		transformed squares	invariants
Projective 8dof	$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$		Concurrency, collinearity, order of contact (intersection, tangency, inflection, etc.), cross ratio
Affine 6dof	$\begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		Parallellism, ratio of areas, ratio of lengths on parallel lines (e.g midpoints), linear combinations of vectors (centroids). **The line at infinity l_∞**
Similarity 4dof	$\begin{bmatrix} sr_{11} & sr_{12} & t_x \\ sr_{21} & sr_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		Ratios of lengths, angles. **The circular points I,J**
Euclidean 3dof	$\begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		lengths, areas.

1.2.2. 3D Transformationen

$$\begin{pmatrix} x' \\ y' \\ z' \\ w' \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22} & h_{23} & h_{24} \\ h_{31} & h_{32} & h_{33} & h_{34} \\ h_{41} & h_{42} & h_{43} & h_{44} \end{bmatrix} \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix}$$

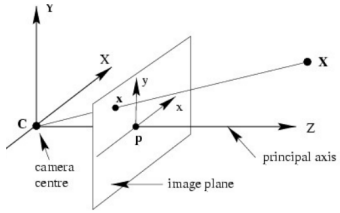
Point transformation: $\tilde{\boldsymbol{X}}' = H\tilde{\boldsymbol{X}}$
Plane transformation: $\boldsymbol{\pi}' = H^{-\top}\boldsymbol{\pi}$

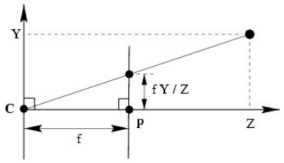
Projective 15dof	$\begin{bmatrix} A & t \\ v^\top & v \end{bmatrix}$		Intersection and tangency
Affine 12dof	$\begin{bmatrix} A & t \\ 0^\top & 1 \end{bmatrix}$		Parallellism of planes, Volume ratios, centroids, **The plane at infinity π_∞**
Similarity 7dof	$\begin{bmatrix} sR & t \\ 0^\top & 1 \end{bmatrix}$		Angles, ratios of length **The absolute conic Ω_∞**
Euclidean 6dof	$\begin{bmatrix} R & t \\ 0^\top & 1 \end{bmatrix}$		Volume

2. Camera Model

2.1. Camera Matrix

2.1.1. Pinhole camera model





Projection on image plane

$$\begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \underbrace{\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{=diag(f,f,1)[I|0]} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

By dividing $(fX, fY, Z)^\top$ by Z we get $(x, y, 1)^\top$

2.1.2. Principal Point offset

Moving the point into the middle of the picture

$$\begin{pmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

By dividing $(fX+Zp_x, fY+Zp_y, Z)^\top$ by Z we get $(x+p_x, y+p_y, 1)^\top$

2.1.3. Kalibration matrix K

Intrinsic camera parameters

$$\underbrace{K = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix}}_{simple{:}a=1 \text{ and } s=0} \qquad \underbrace{K = \begin{bmatrix} f & s & p_x \\ 0 & af & p_y \\ 0 & 0 & 1 \end{bmatrix}}_{general}$$

f: focal length
s: skew between the sensor axes due to the sensor not being mounted perpendicular to the optical axis
a: aspect ratio between f_x & f_y

2.1.4. Camera Rotation + Shift

Extrinsic camera parameters

$$[R|t]$$

R: 3x3 Rotation matrix
t: 3x1 Translation vector

2.1.5. Camera Matrix / Projective Camera

$$P = K[R|t]$$

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = P \begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = K[R|t] \begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix}$$

Full matrix has 11 DOF (5 intrinsic +3 rot +3 trans)
3x4 matrix

$$\tilde{P} = \begin{bmatrix} K & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix}$$

4x4 matrix

$$P = K[R| - RC]$$

2.2. Camera Calibration with Direct Linear Transform (DLT)

The DLT method uses a set of control points whose object space/plane coordinates are already known. The control points are normally fixed to a rigid frame, known as the calibration frame. The problem is essentially to calculate the mapping between the 2D image space coordinates (x,y) and the 3D object space coordinates (X,Y,Z). For this 3D $\leftrightarrow$ 2D correspondence the mapping should take the form of a 3x4 projection matrix (P) such that $x = PX$.

$$x = PX \rightarrow [x]_x PX = 0$$

Denoting the rows of P with P^1, P^2, P^3 we can rewrite above equation to

$$\underbrace{\begin{bmatrix} \mathbf{0}^\top & -w_i\mathbf{X}_i^\top & y_i\mathbf{X}_i^\top \\ w_i\mathbf{X}_i^\top & \mathbf{0}^\top & -x_i\mathbf{X}_i^\top \\ -y_i\mathbf{X}_i^\top & x_i\mathbf{X}_i^\top & \mathbf{0}^\top \end{bmatrix}}_{\text{rank-2 matrix}} \begin{pmatrix} \mathbf{P}^1 \\ \mathbf{P}^2 \\ \mathbf{P}^3 \end{pmatrix} = \mathbf{0}$$

$$\begin{bmatrix} \mathbf{0}^\top & -w_i\mathbf{X}_i^\top & y_i\mathbf{X}_i^\top \\ w_i\mathbf{X}_i^\top & \mathbf{0}^\top & -x_i\mathbf{X}_i^\top \end{bmatrix} \begin{pmatrix} \mathbf{P}^1 \\ \mathbf{P}^2 \\ \mathbf{P}^3 \end{pmatrix} = \mathbf{0}$$

Since the third equation is dependant on the first two, it can be discounted. So for every point X we get two equations.
The bracket is denoted with A (2nx12) and the camera matrix with P (3x4)

$$AP = 0$$

P has 11 DOF $\rightarrow$ 5.5 points are needed for the minimal solution.
For $n \geq 6$ it is over-determined and is solved with SVD.

2.2.1. Gold Standard algorithm

Objective:
Given $n \geq 6$ 2D to 3D point correspondences, determine the Maximum Likelihood Estimation of the camera projection matrix P.

Algorithm:

1. Linear Solution: Compute an initial estimate of P using a linear method.
   a) Normalization $\tilde{X} = UX, \tilde{x} = Tx$
   b) DLT

2. Minimization of geometric error: using the linear estimate as a starting point minimize the geometric error: $\min_P \sum_i d(\tilde{x}_i, \tilde{P}\tilde{X}_i)^2$
3. Denormalization $P = T^{-1}\tilde{P}U$

3. Feature Matching

Feature: measured characteristic of (part of) a pattern / object
Goal: efficient matching
Features shoud overcome large variations in

1. Viewpoint
2. Illumination
3. Background
4. Occlusions
5. Scale change
6. Deformation
7. Perspective Deformation
8. Blur

A feature should capture something **discriminative** about a well **localisable** patch of a surface

3.1. Detector

The detector typically yields image points. Corners are the most prominent example of so-called 'Interest Points', i.e. points that can be well localised in different views of a scene

3.1.1. Uniqueness of a patch

How do the patterns change upon a shift?
flat region: no change in all directions
edge region: no change along edge direction
corner : significant change in all directions

Compare each pixel before and after by summing up the squared differences (SSD) - this defines an error $E(u, v)$. Shift in x-direction: u, in y-direction:y.

$$E(u,v) = \sum_{(x,y)\in W} [I(x+u, y+v) - I(x,y)]^2$$

$$E(u,v) = \sum_{(x,y)\in W} [uv] \underbrace{\begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

Want E(u,v) to be **large** for small shifts in **all** directions - the minimum is given by the smaller eigenvalue of H.

3.1.2. The Harris corner detector

$$R = det(H) - k * trace(H)^2$$

When $|R|$ is small, which happens when λ_1 and λ_2 are small, the region is flat.
When $R < 0$, which happens when λ_1¿¿λ_2 or vice versa, the region is an edge.
When $|R|$ is large, which happens when λ_1 and λ_2 are large and λ_1 λ_2, the region is a corner.

3.2. Descriptor

Descriptors then are a vector of measurements taken around each such point
We need to describe their surrounding image patch such we can discriminate between them, i.e. we need to build a feature vector for the patch & Invariance under geom./phot. change

3.2.1. Deformations under projections (geom. change)

1. similarity 4DOF
2. affinity 6DOF
3. projectivity 8DOF

Complexity of the groups goes up with the generality of the viewing conditions, and so does the complexity of the group's invariants. Fewer invariants are found going from top to bottom

3.2.2. Photometric changes

- Contrasts (intensity differences) let the non-linear offsets cancel; hence gradients are good !
- Moreover the orientation of gradients in the color bands is invariant under their linear changes, as is the intensity gradient orientation in case the scale factors are identical; this is indeed relevant if the illumination changes its intensity, but not its color, which is typically assumed.
- But even under changing color of the illumination, in practice edge orientations tend to remain the same.

3.3. Matching

- Interest points are matched on the basis of their descriptors
- E.g. nearest neighbour, based on some distance like Euclidean or Mahalanobis; good to compare against 2nd nearest neighbour: OK if difference is big; or fuzzy matching w. multiple neighbours
- Speed-ups by using lower-dim. descriptor space (PCA) or through some coarse-to-fine scheme (very fast schemes exist to date!)
- Matching of individual points typically followed by some consistency check, e.g. epipolar geometry, homograpy, or topological

3.4. The Patch

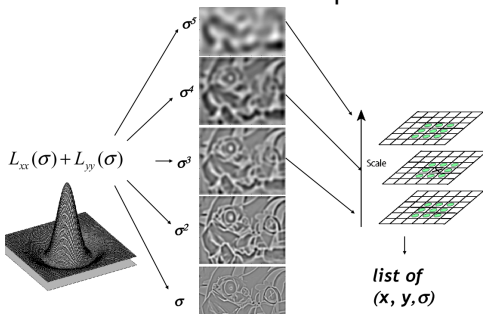
3.4.1. MSER - Maximally stable extremal regions

1. Start with intensity extremum
2. Then move intensity threshold away from its value and watch the super/sub-threshold region grow
3. Take regions at thresholds where the growth is slowest (happens when region is bounded by strong edges)

3.4.2. SIFT - Scale-Invariant Feature Transform

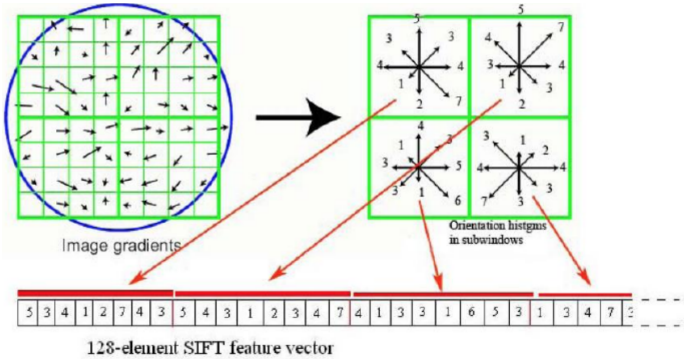
Is a carefully crafted interest point detector + descriptor, based on intensity gradients and invariants under similarities, not affine like so far.

Descriptor is based on blob detection - at several scales - that is local extrema of the Laplacian-of-Gaussian (LoG)



- Thresholded image gradients are sampled over a grid - Dominant orientation selection

  - Compute image gradients of cell
  - Build orientation histogram
  - Find maximum of cell → orientation of cell
- Create array of orientation histograms within blocks
- 8 orientations x 4x4 histogram array (! in picture only 2x2) = 128 dimensions
- Apply weighting with a Gaussian located at the center
- Normalized to unit vector



3.4.3. SURF - efficient alternative to SIFT

no further infos

4. Motion Extraction

Motion can be the only cue for segmentation - camouflage is only visible in motion

Some applications of motion extraction:

- change / shot cut detection in films
- surveillance / traffic monitoring
- autonomous driving
- analyzing game dynamics in sports
- motion capture / gesture analysis
- image stabilisation
- motion compensation (e.g. medical robotics)
- feature tracking for 3D reconstruction (structure from motion)

4.1. Optical Flow

Optical Flow = apparent motion of brightness patterns

Ideally, the optical flow is the projection of the threedimensional motion vectors on the image - Such 2D motion vector is sought at **every** pixel of the image (a motion vector here is a 2D translation vector)

Suppose a point of the scene projects to a certain pixel of the current video frame. Our task is to figure out to which pixel in the next frame it moves… In order to find these corresponding pixels, we need to come up with a reasonable assumption on how we can detect them among the many. We assume these corresponding pixels have the same intensities as the pixels the scene points came from in the previous frame.

Two examples where following brightness patterns is misleading:

1. Untextured, rotating sphere $O.F. = 0$
2. No motion, but changing lighting $O.F. \neq 0$

4.1.1. Mathematical formulation

I (x,y,t) = brightness at (x,y) at time t - one equatio per pixel

$$\frac{DI}{Dt} = \frac{\partial I}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial I}{\partial y}\frac{\partial y}{\partial t} + \frac{\partial I}{\partial t} = I_x u + I_y v + I_t = 0$$

The three derivates can be measured but the velocity u,v are unknown.
something calculus
Horn & Schunck algorithm
something solution of problem

- errors at object boundaries
- example of regularization

4.2. Condensation Filter or condensation tracker

System model (f = system transition function)

$$x_t = f_{t-1}(x_{t-1}, w_{t-1})$$

Measurement model (h = measurement function)

$$z_t = h_t(x_t, v_t)$$

and $Z_t = (z_1, \ldots, z_t)$ is the history of observations

4.2.1. Recursive Bayesian filter (Kalman filter)

Object not as a single state but a probabilistic distribution.

1. prediction

$$p(x_t|Z_{t-1}) = \int p(x_t|x_{t-1})p(x_{t-1}|Z_{t-1})dx_{t-1}$$

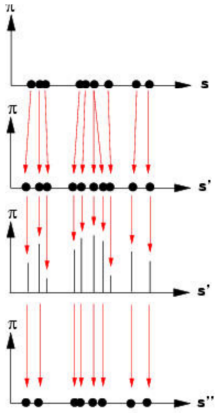
2. measurement update

$$p(x_t|Z_t) = \frac{p(Z_t|x_t)p(x_t|Z_{t-1})}{p(Z_t|Z_{t-1})}$$

Problem integral is computational heavy to compute -¿ condensation tracker (particle filter)

4.2.2. Particle Filter

1. prediction
Start with S_{t-1}, the sample set of the previous step, and apply the system model to each sample, yielding predicted samples s_t.
2. update
Scale each particle by the measurement likelihood. Weight the particles.
3. resample
resample to get N particles with equal weights.



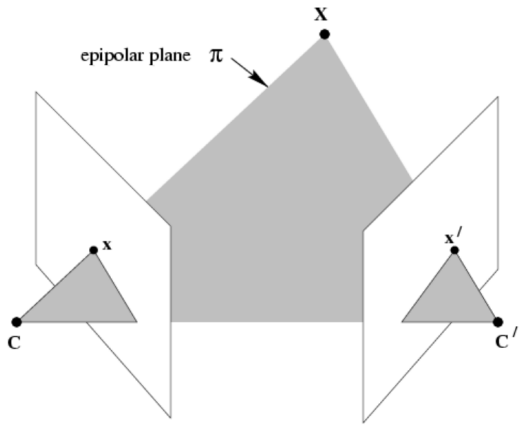
In the limit (large N) equivalent to Bayesian tracker

4.2.3. Comparison Particle Filter - Kalman filter

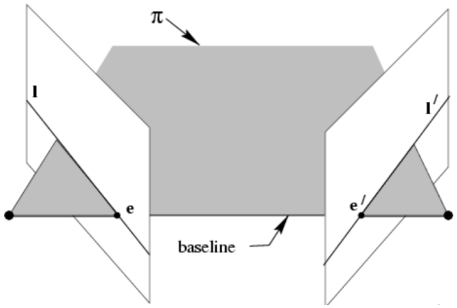
- Unrestricted system models
- Unrestricted noise models
- Multiple hypotheses
- Discretisation error
- Postprocessing for interpret

- Linear system models
- Gaussian noise
- Unimodal
- Exact solution
- Direct interpretation

5. Multiple View Geometry



All points on π project on l and l'



epipoles e,e'

- intersection of baseline with image plane
- projection of projection center in other image
- vanishing point of camera motion direction

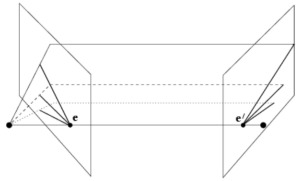
an epipolar plane (π)

- plane containing baseline (1-D family)

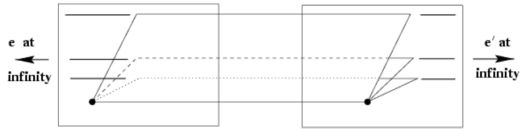
an epipolar line (l)

- intersection of epipolar plane with image (always come in corresponding pairs)

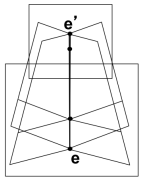
epipolar lines from motion



parallel motion



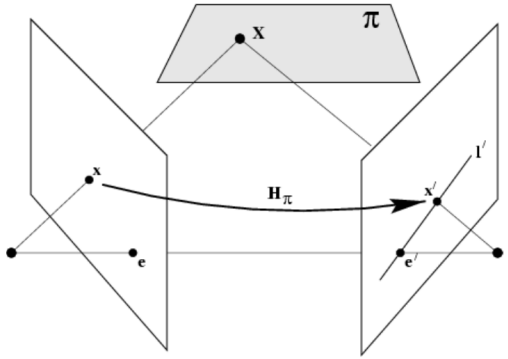
forward motion



5.1. Fundamental matrix

$$l' = e' \times x' = e' \times Hx = Fx$$

The fundamental matrix F is independent of the point.



correspondence condition:

$$x'^T F x = x'^T l' = 0$$

- transpose: if F is fundamental matrix for (P,P'), then transp(F) is fundamental matrix for (P',P)
- Epipolar lines: l'=Fx and l=transp(F)x'
- Epipoles: on all epipolar lines, thus transp(e')Fx=0 for all x therefore transp(e')F = 0, similarly Fe=0
- F has 7 DOF i.e. 3x3-1(homogeneous)-1(rank2)
- F is a correlation, projective mapping from a point x to a line l'=Fx (not a proper correlation, i.e. not invertible)

for pure translation F only has 2 degrees of freedom

5.1.1. Eight-Point Algorithm

Since the fundamental matrix F is a 3x3 matrix determined up to an arbitrary scale factor, 8 linear equations are required to obtain a unique solution. It is possible with 7.

5.2. Essential Matrix

for the calibrated case
5 linear equations are needed.
Same as Fundamental matrix

- Ep' is the epipolar line associated with p'
- transp(E)p is the epipolar line associated with p
- Ee'=0 and transp(E)e=0
- E is singular

New

- E has two equal non-zero singular values

5.3. Multi-view geometry

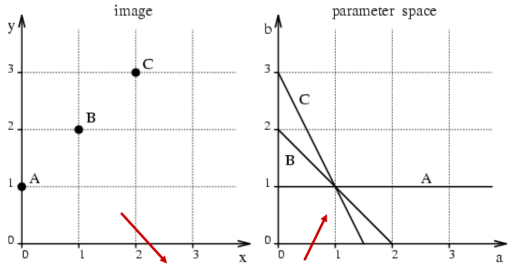
5.4. Dealing with Wide FOV Camera

- Two-step linear approach to compute radial distortion
- estimates distortion polynomial of arbitrary degree

6. Model fitting

6.1. Hough transform

x-y space to a-b parameter space



$$b = (-x) * a + y$$

Alternative: x-y space to θ-ρ (or d) space

$$xcos(\theta) + ysin(\theta) = \rho$$

For each point, render the curve (θ, d) and count the parameter in bins with similar values.
Difficulties: How big should the bins be (too big, and we cannot distinguish between quite different lines; too small, and noise causes lines to be missed)
Hardly ever satisfactory in practice, because problems with noise and bin size defeat it

6.2. Line fitting

6.2.1. Incremental line fitting

```
Put all points on a curve list, in order along the curve
Empty the line point list
Empty the line list
Until there are too few points on the curve
        Transfer first few points on the curve tot the line
             point list
        Fit line to line point list
        While fitted line is good enough
                Transfer the next point on the curve to tthe
                     line point list and refit the line
        end
        Transfer last point(s) back to curve
        Refit line
        Attach line to line list
end
```

6.2.2. K-means

```
Hypothesize k lines (perhabps uniformly at random)
or
Hypothesize an assignment of lines to points and then fit
     lines using this assingment

Until convergence
        Allocate each point to the closest line
        Refit lines
end
```

Problem can be stuck in local optima

6.2.3. Probabilistic

- EM-algorithm (expectation maximum) - M-estimators - RANSAC

6.2.4. RANSAC

- Choose a small subset uniformly at random - Fit to that - Anything that is close to result is signal; all others are noise - Refit - Do this many times and choose the best

Issues

- How many times? Often enough that we are likely to have a good line
- How big a subset? Smallest possible
- What does close mean? Depends on the problem
- What is a good line? One where the number of nearby points is so big it is unlikely to be all outliers

```
Determine:
        n—the smallest number of points required
        k—the number of iterations required
        t—the threshold used to identify a point that fits
             well
        d—the number of nearby points required to assert a
             model fits well
Until k iterations have occured
        Draw a sample of n points from the data uniformly
             and at random
        Fit to that set of n points
        For each data point outside the sample
                Test the distance from the point to the line
                     against t; if the distance from the
                    point to the line is less than t, the
                    point is close
        end
        If there are d or more points close to the line then
              there is a good fit. Refit the line using all
             these points.
end
Use the best fit from this collection, using the fitting
     error as criterium
```

$$k = \frac{log(1-p)}{log(1-w^s)}$$

k - number of iterations
p - probability at least one sample free from outliers
w - fraction of inliers = d/n
s - model dimension (2 for a line, 3 for plane)

6.3. Missing variable problems

In many vision problems, if some variables were known the maximum likelihood inference problem would be easy

- fitting; if we knew which line each token came from, it would be easy to determine line parameters
- segmentation; if we knew the segment each pixel came from, it would be easy to determine the segment parameters
- fundamental matrix estimation; if we knew which feature corresponded to which, it would be easy to determine the fundamental matrix

This sort of thing happens in statistics, too - strategy:

1. estimate missing variables
2. plug these in, now estimate parameters
3. re-estimate appropriate values for missing variables - converges to local extremum (like k-means)

6.4. Cross validation

-Split data set into two pieces, fit to one, and compute negative loglikelihood on the other -Average over multiple different splits -Choose the model with the smallest value of this average -The difference in averages for two different models is an estimate of the difference in KL divergence of the models from the source of the data

7. Image segmentation

Identify groups of pixels that belong together - goal: Delineate objects and background regions & group similar-looking pixels for efficientcy of further processing
Examples of Groupin in vision

- similar appearance
- symmetry
- common fate (direction etc)
- proximity

7.1. Segmentation as clustering

7.1.1. Clustering

Chicken and Egg problem - either know the centers and allocate the points to it OR know the point membership and create center from them
Feature space - depending on that choice we can group pixels in different ways

- pixel intensity
- pixel color
- pixel texture

- pixel intensity + position (proximity)

7.1.2. k-Means

Idea: randomly initialize the k cluster centers, and iterate between the two steps from clustering
Properties: will always converge to some solution (which can be a local maximum)
Pro:

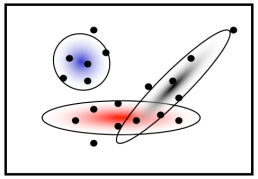
- simple, fast to compute
- converges (to local maximum)

Cons:

- number of cluster has to be defined
- sensitive to initial centers
- sensitive to outliers
- detects spherical clusters only
- assuming means can be computed

7.1.3. Mixture of Gaussians, EM

Idea: instead of treating the data as a bunch of points, assume that they are all generated by sampling a continous function - this function is called a *generative model*, defined by a vector of parameters θ



- **Goal**
  - **Find blob parameters θ that maximize the likelihood function overall all *N* datapoints $X = \{x_1, \ldots, x_N\}$**

$$P(X) = \prod_{x_i \in X} P(x_i|\theta)$$

- **Approach:**
  1. **E-step: given current guess of blobs, compute probabilistic ownership of each point**
  2. **M-step: given ownership probabilities, update blobs to maximize likelihood function**
  3. **Repeat until convergence**

E-Step: Compute probability that point x is in blob b, given current guess of θ

$$P(b|x, \mu_b, V_b) = \frac{\alpha_b P(x|\mu_b, V_b)}{\sum_{i=1}^{K} \alpha_i P(x|\mu_i, V_i)}$$

M-Step: Compute overall probability that blob b is selected (N: data points)

Weight:

$$\alpha_b^{new} = \frac{1}{N} \sum_{i=1}^{N} P(b|x_i, \mu_b, V_b)$$

Mean of blob b:

$$\mu_b^{new} = \frac{\sum_{i=1}^{N} x_i P(b|x_i, \mu_b, V_b)}{\sum_{i=1}^{N} P(b|x_i, \mu_b, V_b)}$$

Covariance of blob b:

$$V_b^{new} = \frac{\sum_{i=1}^{N} (x_i - \mu_b^{new})(x_i - \mu_b^{new})^T P(b|x_i, \mu_b, V_b)}{\sum_{i=1}^{N} x_i P(b|x_i, \mu_b, V_b)}$$

EM Application is useful for all sorts of problems

- Any clustering problem
- Many model estimation problems
- Missing data problems
- Finding outliers
- Segmentation problems
  - based on color
  - based on motion
  - Foreground/background separation

Pro:

- Probabilistic interpretation
- Soft assignments between data points and clusters
- Generative model, can predict novel data points
- Relatively compact storage ($O(Kd^2)$)

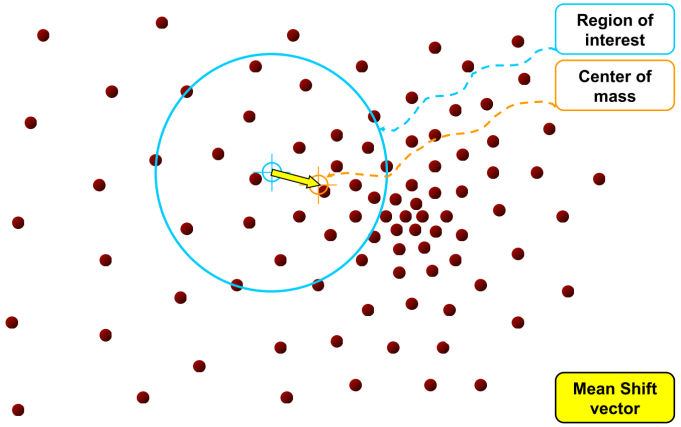
Cons:

- Initialization – often a good idea to start from output of k-means
- Local minima
- Need to know number of components K – Solutions: model selection (AIC, BIC), Dirichlet process mixture
- Need to choose blob generative model (math form of a cluster?)
- Numerical problems are often a nuisance

7.1.4. Mean-Shift Algorithm

Choose features (color, gradients, texture, etc)

1. Initialize random seed center and window size W
2. Calculate center of gravity (the "mean") of W: $\sum_{x \in W} xH(x)$
3. Shift the search window to the mean
4. Repeat steps 2+3 until convergence for all centers
5. merge windows that end up near the same peak/center



Pro:

- General, application-independent tool
- Model-free, does not assume any prior shape (spherical, elliptical, etc.) on data clusters
- Just a single parameter (window size h) – h has a physical meaning (unlike k-means) == scale of clustering
- Finds variable number of modes given the same h
- Robust to outliers

Cons:

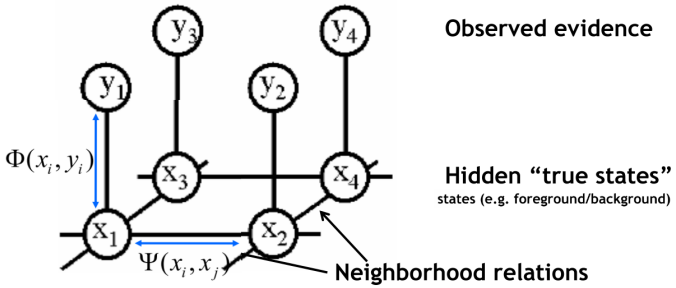
- Output depends on window size h
- Window size (bandwidth) selection is not trivial
- Computationally rather expensive
- Does not scale well with dimension of feature space

7.2. Hough transform

see chapter before

7.3. Interactive Segmentation with GraphCuts

Markov Random Fields



Field Joint Probability

$$P(x,y) = \prod_i \Phi(x_i, y_i) \prod_{i,j} \Psi(x_i, x_j)$$

states; Image; Image-state compatibility function; Local observations; state-state compatibility function; States of neighboring nodes

- **Maximizing the joint probability is the same as minimizing the log**

$$\log P(x,y) = \sum_i \log \Phi(x_i, y_i) + \sum_{i,j} \log \Psi(x_i, x_j)$$

$$E(x,y) = \sum_i \varphi(x_i, y_i) + \sum_{i,j} \psi(x_i, x_j)$$

- **This is similar to free-energy problems in statistical mechanics (spin glass theory). We therefore draw the analogy and call E an *energy function*.**
- **φ and ψ are called *potentials* (unary and pairwise)**

Unary potentials Φ

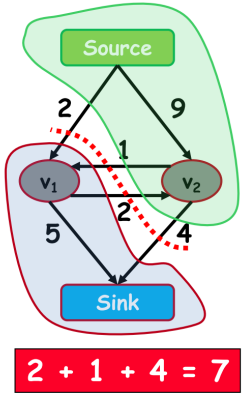
- Encode local information about the given pixel/patch
- How likely is a pixel/patch to be in a certain state (e.g. foreground/background)

Pairwise potentials Ψ

- Ecode neighborhood information
- How different is a pixel/patch's label from that of its neighbor (e.g. here independent of image data, but later based on intensity/color/texture difference)

Goal: minimum cost cut

The s-t-Mincut Problem



What is an st-cut?

An st-cut (S,T) divides the nodes between source and sink.

What is the cost of a st-cut?

Sum of cost of all edges going from S to T

What is the st-mincut?

st-cut with the minimum cost

Maxflow Algorithm

1. Find path from source to sink with positive ! capacity
2. push maximum possible flow through this path (subtract it from all segments)
3. repeat until no path can be found

Pro:

- Powerful technique, based on probabilistic model (MRF).
- Applicable for a wide range of problems.
- Very efficient algorithms available for vision problems.
- Becoming a de-facto standard for many segmentation tasks.

Cons:

- Graph cuts can solve a limited (but useful) class of models
- Submodular energy functions
- Can capture only part of the expressiveness of MRFs
- Only approximate algorithms available for multi-label case (except for special cases with particular topology and/or pairwise potentials)

7.4. Learning-based approaches

Extract the same features used during training & Use the mapping learned before to label

7.4.1. K-nearest neighbor

For every point find the k nearest neighbors (labeled in the training set). The point is then defined through the lables of these neighbors (majority voting of these k neighbors)

Pro:

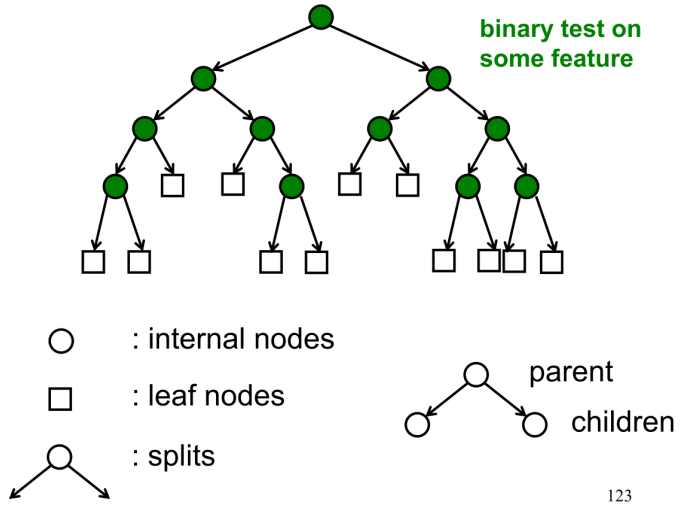
- Very simple to implement
- Very simple to understand
- efficient implementations possible for approx. NNs
- distance definition is flexible

Cons:

- highly depends on the definitions and k
- need to keep the entire data in memory for distance computations
- for high dimensional problems might need many training samples for accuracy
- other methods have better generalization ability

7.4.2. Random forests

Binary decision trees - Random Forests are slow to train, but fast to apply during testing.



TRaining: During training, samples are used for which the true class is known. Having arrived at a node, one randomly generates a set of possible tests to carry out in order to split it up. Among the possibilities one selects the test that maximally reduces the uncertainty. (biggest difference)

Testing: Feed the current sample into every tree. See in

which leaf node it ends -¿ (most probable) class. Combine the results, e.g. which class/label was found most often

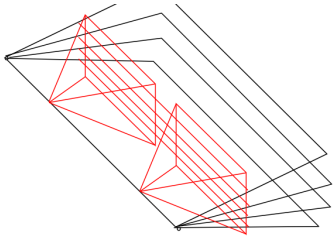
Pro:

- Easy to implement
- very efficient during testing
- can easily use diverse features
- can handle high dimensional spaces

Cons:

- Lots of parametric choices
- needs large number of data
- training can take time

8. Stereo Matching - needs work done



pure translation along X-axis

$$\mathbf{F} = [\mathbf{t}]_\times = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}$$

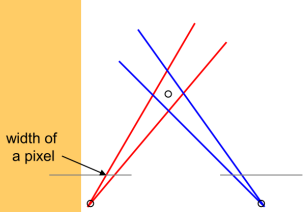
$$\mathbf{F}\mathbf{x} = \begin{bmatrix} 0 \\ 1 \\ -y \end{bmatrix}$$

$$\mathbf{F}^\top \mathbf{x}' = \begin{bmatrix} 0 \\ 1 \\ -y' \end{bmatrix}$$

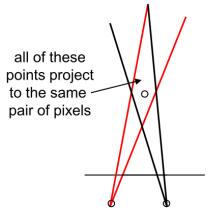
8.1. Occlusion

In an image pair each pixel has at most one corresponding pixel – In general one corresponding pixel – In case of occlusion there is none

Choosing the stereo baseline



Large Baseline



Small Baseline

- What's the optimal baseline?
  - Too small: large depth error
  - Too large: difficult search problem, more occlusions

8.2. Deep Stereo Matching

8.3. 3D convex hull

The 3D convex hull of a set of 3D points is the smallest convex set which contains all 3D points.

8.4. Visual hull

The visual hull is the smallest volume which can be obtained by intersecting the generalized cones generated by all possible 2D shape projections ofa 3D object.

In 3D, the visual hull is a subset ofthe convex hull.

8.4.1. voxel-base, marching intersections, exact polyhedral

8.5. Photo hull

The photo hull is the union ofall photo-consistent volumes and the tightest possible bound on the true scene.

It is a subset of the visual hull and - in contrast to the 3D convex hull and the visual hull depends on the texture of the 3D object.

8.5.1. Voxel coloring

8.5.2. Space Carving

Step 1: Initialize V to volume containing true scene
Step 2: For every voxel on surface of V test photo-consistency of voxel if voxel is inconsistent, carve it
Step 3: Repeat Step 2 until all voxels consistent

Convergence: Always converges to a photo-consistent model (when all assumptions are met) Good results on difficult real-world scenes

Properties of Volume Intersection
Pros – Easy to implement, fast – Accelerated via octrees [Szeliski 1993]
Cons – No concavities – Reconstruction is not photo-consistent – Requires identification of silhouettes

9. Structure from Motion

9.1. Factorization

Factorise observations in structure of the scene and motion/-calibration of the camera Use all points in all images at the same time

9.1.1. perspective factorization

$$\lambda m = PM$$

something

9.2. structure from motion

seriously no plan

10. Specific Object Recognition

A specific object = an instance of an object class e.g. "my car" instead of "a car"
Must overcome:

- Illumination

- Object pose
- Clutter
- Occlusions
- Viewpoint

For class recognition also intra-class variation must be detected.

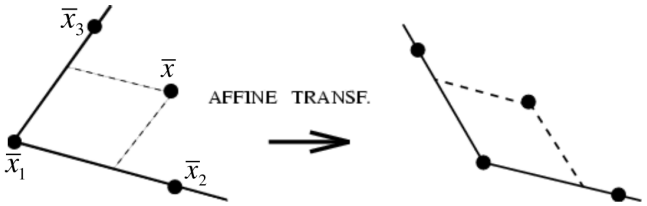
10.1. Model based

comparing image features with features of objects in a database, trying to figure out type + pose
this is SLOW and especially the need to get both object type and pose right to formulate a correct hypothesis is problematic

10.1.1. Invariant-based recognition of planar shapes

The crucial advantage of invariants is that they decouple object type and pose issues
ratios of areas are affine invariant and the following invariants are based on this

$$x_A = \frac{|\bar{x} - \bar{x}_2 \quad \bar{x} - \bar{x}_3|}{|\bar{x}_1 - \bar{x}_3 \quad \bar{x}_2 - \bar{x}_3|} \qquad y_A = \frac{|\bar{x} - \bar{x}_3 \quad \bar{x} - \bar{x}_1|}{|\bar{x}_1 - \bar{x}_3 \quad \bar{x}_2 - \bar{x}_3|}$$



10.2. Image based

PCA represents data in a lower-dimensional space keeping most of the variance It was seen to be powerful for similar patterns like faces, that exhibit a lot of redundancy

10.2.1. Appearance manifold approach

Training:
for every object : - sample the set of viewing conditions (mainly viewpoints in this ex.) - use these images as feature vectors (after manual bounding-box fitting around the object, rescaling, brightness normalization) over all objects: - apply a PCA over all the images of all objects (directly on the images) - keep the dominant PCs (10-20 enough already) - sequence of views for 1 object represent a manifold in the space of projections (fit splines to manifolds + resample if desired)
Recognition stage (aka 'Testing')

Represent the incoming image as a point in the same PC space
Type: what is the nearest manifold to the point ? Pose: what is the closest point on that closest manifold ?

10.3. Model based vs appearance based

Model vs Image

+Compact model vs -Large models +Can deal with clutter vs -Cannot deal with clutter -Slow analysis-by-synthesis vs +Efficient -Models difficult to produce vs +Models easy to produce -For limited object classes vs +For wide classes of objects

10.4. Hybrid techniques

10.4.1. Euclidean invariant feature

Training
1. look for corners (with the Harris corner detector)
2. take circular regions around these points, of multiple radii (cope a bit with scale changes)
3. calculate from the intensities in the circular regions . invariants under planar rotation -¿ feature vectors
4. do this from different viewpoints, where the invariance cuts down on the number of views needed (here no in-plane rotations necessary)
5. put for every object and for each of its viewpoints the . list of corner positions and their invariant feature . vectors (descriptors) in a database

Testing
1. extract corners and their invariant descriptors from the incoming image
2. compare these invariants with those stored in the database -¿ find matches
3. look for consistent placement of candidate matching corner points (e.g. using epipolar geometry)
4. decide which object based on the number of remaining matches (i.e. consistently placed matches) (the best matching image yields the object type+appr.pose)

10.4.2. Recognition using local affine and photometric invariant features

Invariant features = features that are preserved under a specific group of transformations

hyprid approach that aims to deal with large variations in
- Viewpoint
- Illumination
- Background
- Occlusions

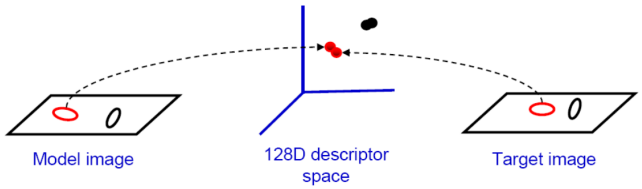
11. Object Class Recognition

Two main tasks: Classification and Detection
Classification: is there a car in this image ? A binary answer is enough
Detection: where is the car ? Need localization

11.1. Bag of words approaches: Visual words

11.1.1. Local features

Local (textured) patches ('regions') • Detection co-variant with translation, scale, and sometimes affine transformations • Cover the same portion of the object in any view
Each patch / region has a descriptor, which is a point in some high-dimensional feature space (e.g., SIFT)
Close points in feature space have similar descriptors, which indicates similar image content.



11.1.2. Visual Words

1) Extract some local features from a number of images
2) Map high-dimensional descriptors to words by quantizing the feature space (Quantize via clustering, let cluster centers be the prototype "words")
3) Map high-dimensional descriptors to words by quantizing the feature space. Determine which word to assign to each new image region by finding the closest cluster center
Building Visual Vocabularies Issues: • Sampling strategy: where to extract features? -¿ For object classes dense sampling works best • Clustering / quantization algorithm -¿ Many options, often k-means works well enough • What corpus provides features (universal vocabulary?) -¿ Typically as close as possible to your application • Vocabulary size, number of words

11.2. Bag of Words

Summarize entire image based on its distribution (histogram) of word occurrences. (Analogous to bag of words representation commonly used for documents.) Quantize feature space to make discrete set of visual words
Bag of words enable to describe the unordered feature set with a single vector of fixed dimensionality
Works pretty well for whole-image classification
Classifier Choises

- Nearest Neighbor
- Neural Networks
- Support Vector Machines
- Boosting

Nearest Neighbor - Assign label of nearest training data point to each test data point. Works very fast when training data is huge. Support Vector Machines - Maximize the margin between the positive and negative training examples
The bag of words removes spatial layout. This is both a strength and a weakness

11.2.1. Discussion: Bag-of-Words

Pros:

- Flexible to geometry / deformations / viewpoint
- Compact summary of image content
- Provides vector representation for sets (bags to be precise)
- Empirically good recognition results in practice

Cons:

- Basic model ignores geometry – can verify afterwards, or embed within feature descriptors
- Background and foreground mixed when bag covers whole image - Interest points or sampling: no guarantee to capture object-level parts
- Optimal vocabulary formation remains unclear

11.3. Classification: convolutional neural networks

Neural network with specialized connectivity structure.
Typical building blocks of CNNs for classification in vision

- Feed forward
- Most layers (often)
  - Convolve input (filters)
  - Non-linearity (rectified linear)
  - Pooling (local max)
- Last few layers (often)
  - fully connected (linear combinations)
  - Non-linearity (rectified linear)
- output: softmax distribution over classes

supervise, train convolutional filters by back-propagatin classification error

11.4. Detection: Sliding-window approaches

If object may be in a cluttered scene, slide a window around looking for it. - A brute force approach with many local decisions

11.4.1. Gradient based Representations

Summarize local distribution of gradients with histogram

11.5. Boosting

Intuition Consider a 2D feature space with positive and negative examples. Each weak classifier splits the training examples with at least 50Examples misclassified by a previous weak learner are given more emphasis at future rounds. Final classifier is combination of the weak classifiers

11.6. Discussion: Sliding-Windows

Pros

- Simple detection protocol to implement
- Good feature choices critical
- Past successes for certain classes
- Good detectors available (Viola&Jones, HOG, etc.)

Cons/Limitations

- High computational complexity
  - – For example: 250,000 locations x 30 orientations x 4 scales = 30,000,000 evaluations!
  - This puts tight constraints on the classifiers we can use.
  - If training binary detectors independently, this means cost increases linearly with number of classes.
- With so many windows, false positive rate better be low
- Typically need fully supervised training data (= bounding-boxes)
- Some object do not fit a box well (diagonal bottle) Sensitive to partial occlusion (unless in training data)

11.7. Detection: proposal-based approaches

11.7.1. Object proposals

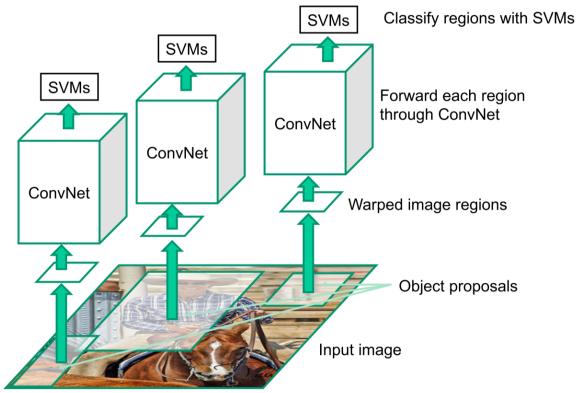
Every object has at least one of these properties • Well-defined, closed boundary in space • Different appearance than its surroundings • Might be unique within the image (salient)

Objectness(w) = probability that w covers an object

Objectness cues

- Density of salient pixels
- color contrast
- superpixel straddling

11.7.2. R-CNN



Pros

- Accurate!
- Any deep architecture can immediately be plugged in

Cons

- Ad hoc training objectives
  - Fine-tuning network with softmax classifier (log loss)
  - Train post-hoc linear SVMs (hinge loss)
  - Train post-hoc bounding-box regressions (least squares)
- Training is slow (84h), takes a lot of disk space
- Inference (detection) is slow

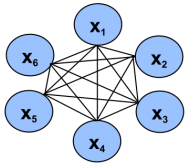
11.7.3. Fast R-CNN

Full Image first through ConvNet and afterwards Region selection

11.8. Detection: a star model

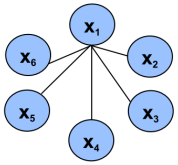
Recognition of object categories

Fully connected



- e.g. Constellation Model
- Parts fully connected
- Recognition complexity: $O(N^P)$
- Method: Exhaustive search

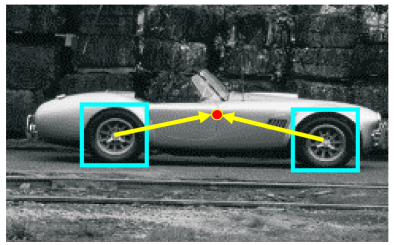
'Star' model



- e.g. Implicit Shape Model (ISM)
- Parts mutually independent
- Recognition complexity: O(NP)
- Method: Gen. Hough Transform

11.8.1. Implicit Shape Model

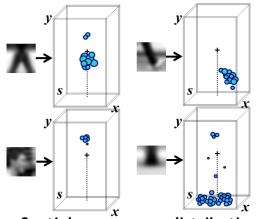
Visual vocabulary is used to index votes for object position [a visual word = a "part"]. Objects are detected as consistent configurations of the observed parts (visual words).



Training image

Visual codeword with displacement vectors

- Learn appearance codebook
  - Extract local features at interest points
  - Agglomerative clustering ⇒ codebook
- Learn spatial distributions
  - Match codebook to training images
  - Record matching positions on object



Spatial occurrence distributions

Pros:

- Works well for many different object categories – Both rigid and articulated objects
- Flexible geometric model – Can recombine parts seen on different training examples
- Learning from relatively few (50-100) training examples
- Optimized for detection, good localization properties

Cons:

- Needs supervised training data – Object bounding boxes for detection & Segmentations for top-down segmentation
- Only weak geometric constraints – Result segmentations may contain superfluous body parts.
- Purely representative model – No discriminative learning

A. Glossary

Mode = local maximum of a given distribution

k-means vs knn KNN represents a supervised classification algorithm that will give new data points accordingly to the k number or the closest data points, while k-means clustering is an unsupervised clustering algorithm that gathers and groups data into k number of clusters.