Computer Vision I - *Multi-View 3D reconstruction and Decision Trees*

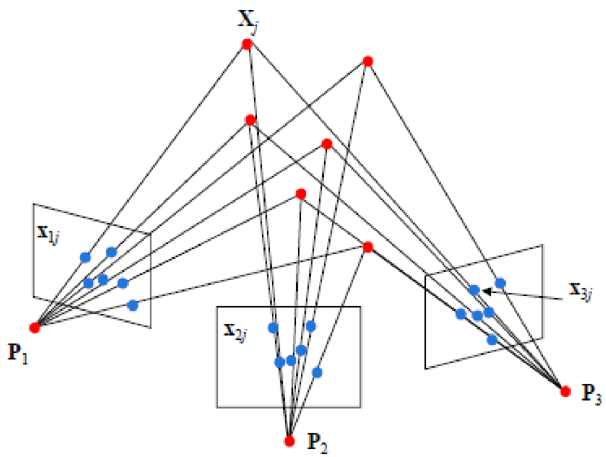
Carsten Rother

03/12/2015

Reminder: Reconstruction Algorithm

Generic Outline (calibrated and un-calibrated cameras)

1) Compute robust F/E-matrix between each pair of neighboring views
2) Compute initial reconstruction of consecutive pair of views
3) Compute an initial full 3D reconstruction
4) Bundle-Adjustment to minimize overall geometric error
5) If cameras are not calibrated then perform auto-calibration (also known as self-calibration)



Reconstruct in step 2): (P_1, P_2); (P_2, P_3); (P_3, P_4) ...

[See page 453 HZ]

Reminder:
Step 2: Compute initial reconstruction of consecutive pair of views

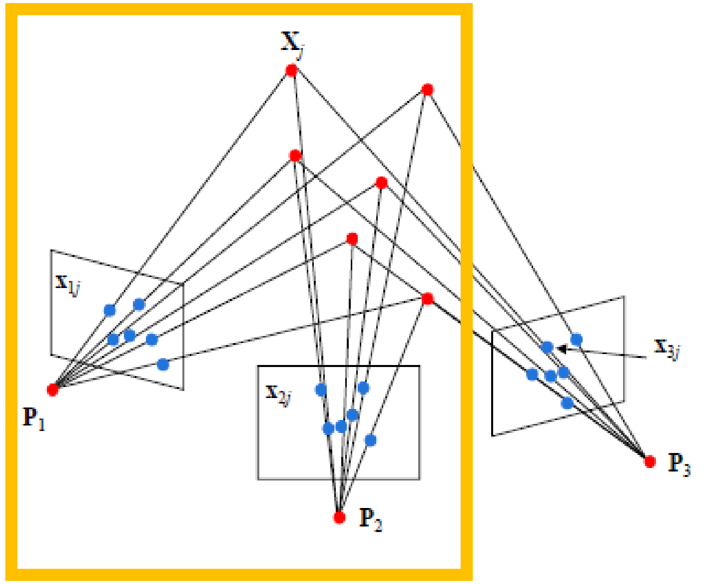
Input:

- Calibrated Cameras: E-matrix, K, K', 5+ matching points (x_i, x'_i)
- Un-calibration Cameras: F-matrices, 7+ matching points (x_i, x'_i)

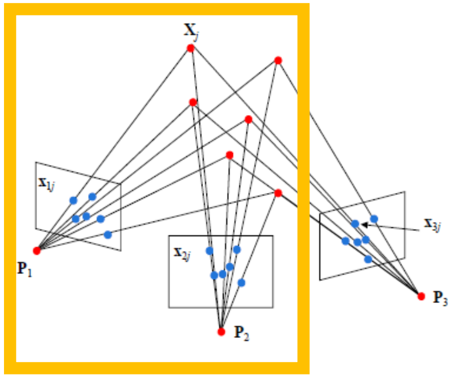
Output: $P, P', X_{i's}$ such that geometric error: $PX_i \ to \ x_i$ and $P'X_i \ to \ x'_i$ is small

2-Step Method:

1. Derive P, P'
2. Compute $X_{i's}$ (called Triangulation)

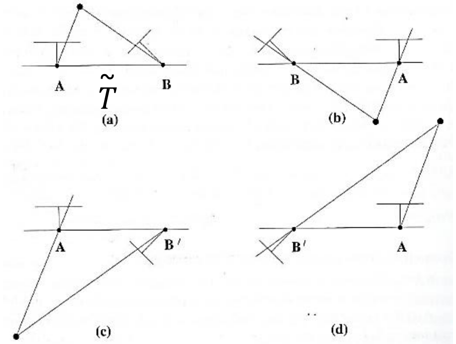


Reminder: Derive P, P': calibrated case



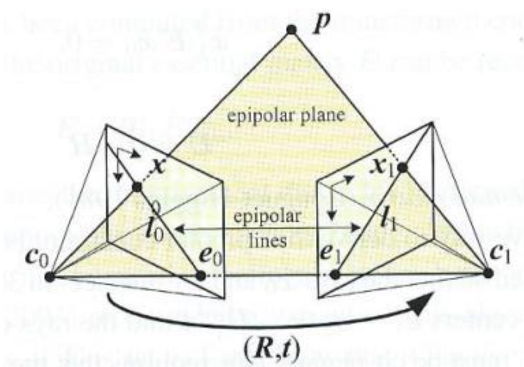
<u>We have done this already</u>:

- We have seen that we can get: $R, \tilde{T}$(up to scale) from E



- We have set in previous lecture the camera matrices to:

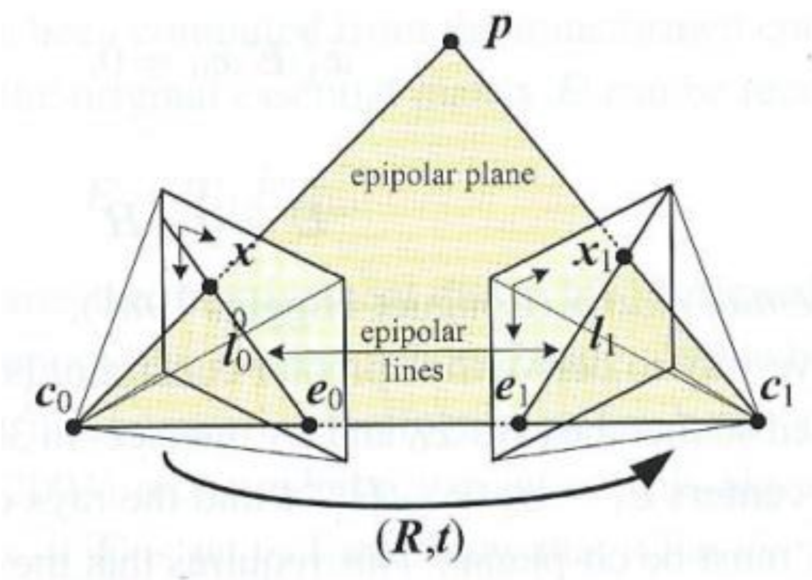
$$x_0 = \underbrace{K_0[I|0]}_{P}\, X \text{ and } x_1 = \underbrace{K_1 R^{-1}[I|-\tilde{T}]}_{P'}\, X$$

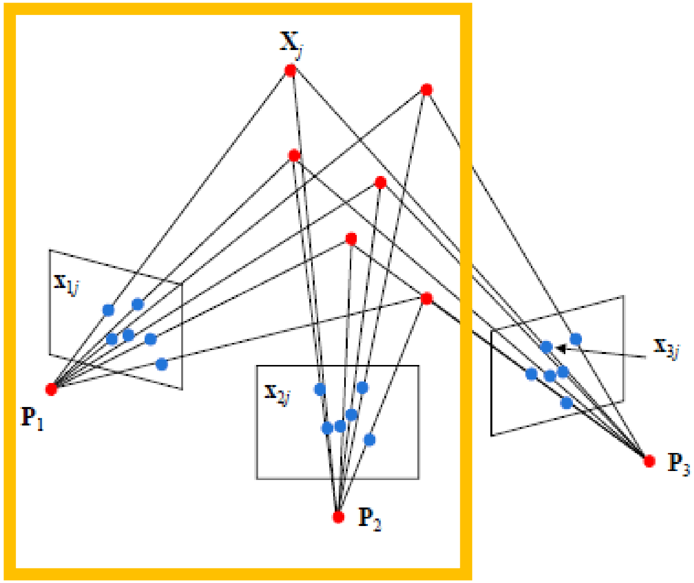


Reminder: Derive P, P': un-calibrated case

- Derivation (blackboard) see HZ page 256

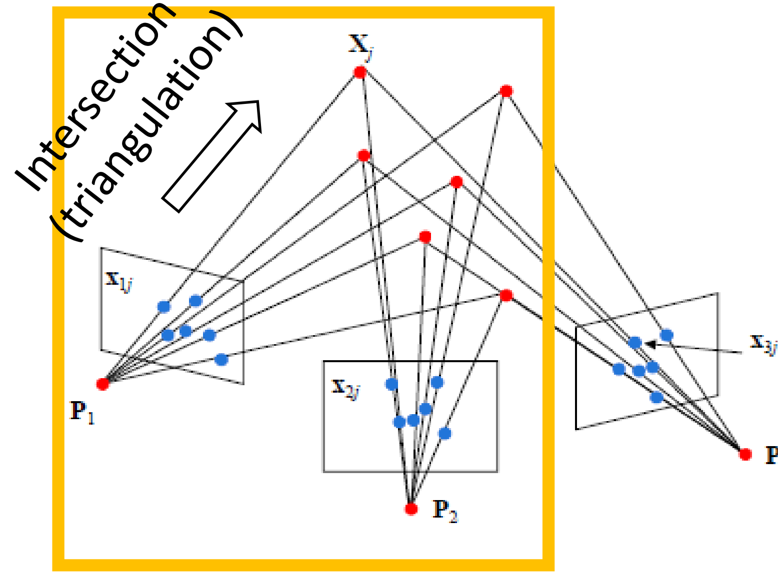
$$P = [I_{3\times 3} \mid 0\,]; \quad P' = [\,[e']_{\times}F \mid e']$$





Reminder: Compute $X_{i's}$ (Triangulation)

- <u>Input:</u> x, x', P, P'
- <u>Output:</u> $X_{i's}$
- Triangulation is also called intersection
- Simple solution for algebraic error:



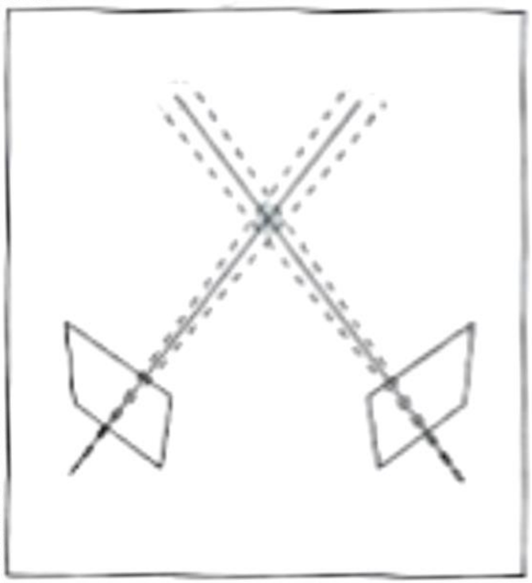
1) $\lambda x = \underset{\text{3x4 matrix}}{P}\, X$ and $\lambda' x' = P'\, X$

2) Eliminate λ by taking ratios. This gives 2x2 linear-independent equations for 4 unknowns: $X = (X_1, X_2, X_3, X_4)$, and we want: $\|X\| = 1$.
(remember X is a homogenous 4D vector, hence scale has to be fixed)

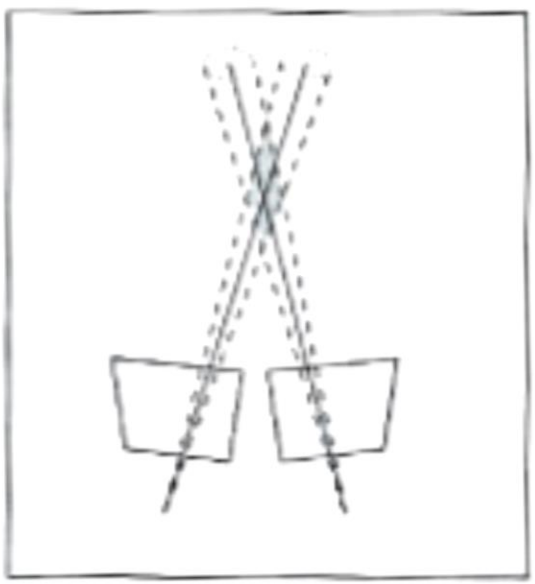
An example ratio is: $\frac{x_1}{x_2} = \frac{p_{11}\, X_1 + p_{12} X_2 + p_{13} X_3 + p_{14} X_4}{p_{21}\, X_1 + p_{22} X_2 + p_{23} X_3 + p_{24} X_4}$

3) This gives (as usual) a least square optimization problem:
$A\, X = 0$ with $\|X\| = 1$ where A is of size 4×4.
This can be solved in closed-form using SVD.

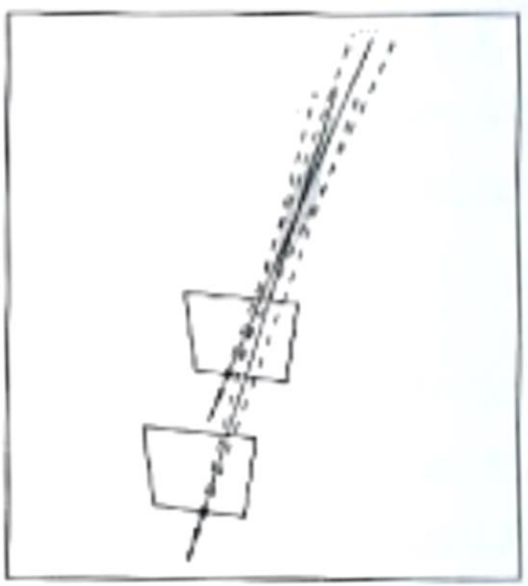
Reminder: Triangulation: Uncertainty



Large baseline
Smaller uncertainty area



Smaller baseline
Larger uncertainty area

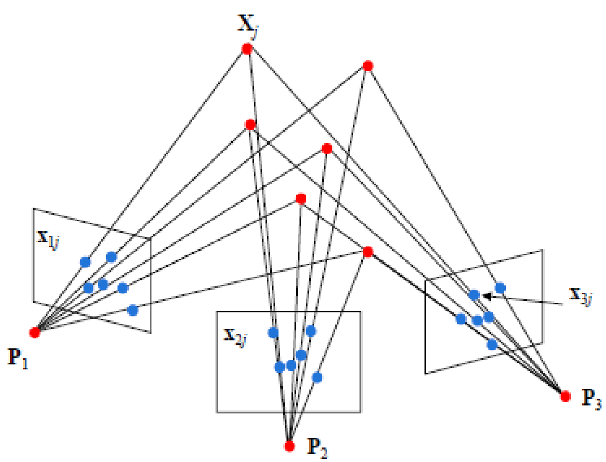


Very small baseline
Very large
uncertainty area

Reconstruction Algorithm

Generic Outline (calibrated and un-calibrated cameras)

1) Compute robust F/E-matrix between each pair of neighboring views
2) Compute initial reconstruction of consecutive pair of views
3) Compute an initial full 3D reconstruction
4) Bundle-Adjustment to minimize overall geometric error
5) If cameras are not calibrated then perform auto-calibration (also known as self-calibration)



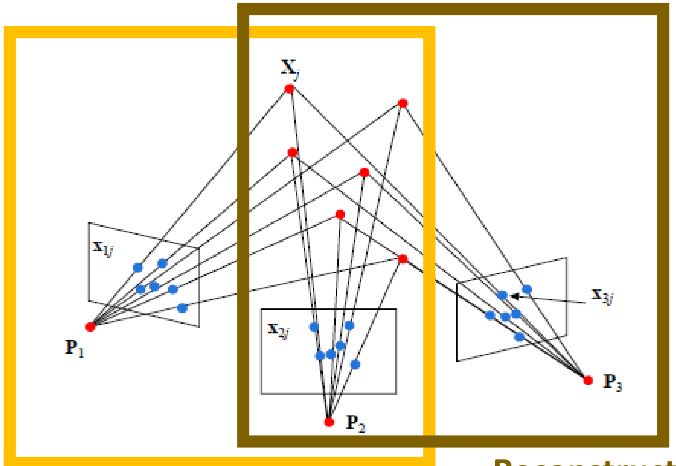
Reconstruct in step 2): (P_1, P_2); (P_2, P_3); (P_3, P_4) ...

[See page 453 HZ]

Step 3: Compute initial reconstruction

Three views of an un-calibrated or calibrated camera:

X_i, P_1, P_2 X_i', P_2', P_3'



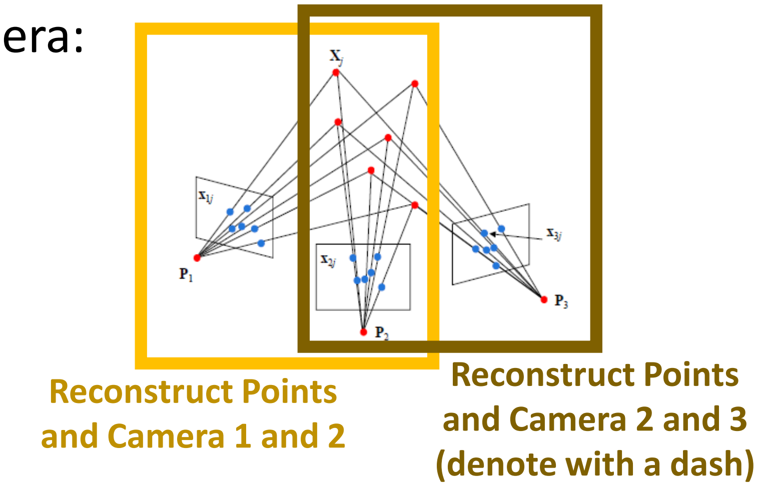
Reconstruct Points and Camera 1 and 2

Reconstruct Points and Camera 2 and 3 (denote with a dash)

- Both reconstructions share: 5+ 3D points and one camera (here P_2, P_2'). (We denote the second reconstruction with a dash)
- Why are $X_\mathrm{i}, X_i{}'$ not the same?
  In general we have the following ambiguity: $x_{ij} = P_j X_i = P_j Q^{-1} Q X_i = P_j' X_i'$
- **Our Goal:** make $X_i = X_i'$ and $P_2 = P_2'$ such that $x_{ij} = P_j X_i$
  (remember all mean "=" mean equal up to scale. All elements, x, X and P are defined up to scale)

Step 3: Compute initial reconstruction

Three views of an un-calibrated or calibrated camera:



Method:

- Compute Q such that $X_{1-5} = QX'_{1-5}$ (up to scale)
- This can be done from 5+ 3D points in usual least-square sense ($\|AQ\|$), since each point gives 3 equations and Q has 15 DoF.

  An example ratio is: $\frac{X^1}{X^2} = \frac{Q_{11}X^{1}\prime + Q_{12}X^{2}\prime + Q_{13}X^{3}\prime + Q_{14}X^{4}\prime}{Q_{21}X^{1}\prime + Q_{22}X^{2}\prime + Q_{23}X^{3}\prime + Q_{24}X^{4}}$

  for $X_1 = (X^1, X^2, X^3, X^4); X'_1 = (X^{1\prime}, X^{2\prime}, X^{3\prime}, X^{4\prime})$

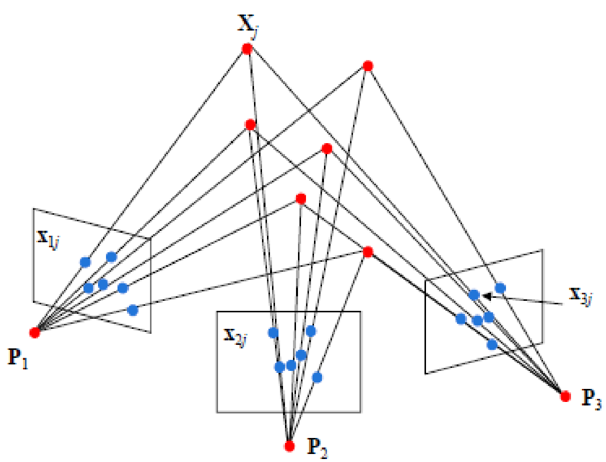
- Convert the second (dashed) reconstruction into the first one:

  $P'_{2,3}(new) = P'_{2,3}Q^{-1}; \quad X'_i(new) = QX'_i$ (note: $x_{ij} = P_jX_i = P_jQ^{-1}QX_i$)
- In this way you can "zip" all reconstructions into a single one, in sequential fashion.

Reconstruction Algorithm

Generic Outline (calibrated and un-calibrated cameras)

1) Compute robust F/E-matrix between each pair of neighboring views
2) Compute initial reconstruction of consecutive pair of views
3) Compute an initial full 3D reconstruction
4) Bundle-Adjustment to minimize overall geometric error
5) If cameras are not calibrated then perform auto-calibration (also known as self-calibration)



Reconstruct in step 2): (P_1, P_2); (P_2, P_3); (P_3, P_4) ...

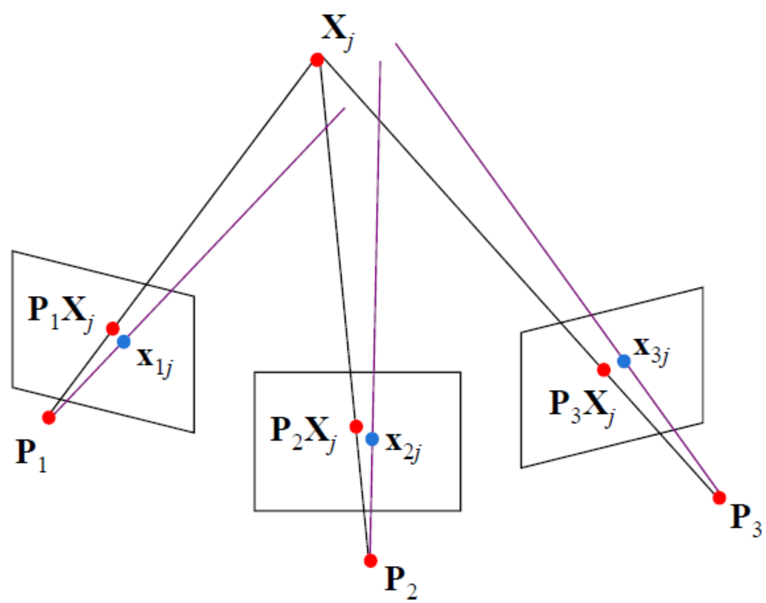
[See page 453 HZ]

Bundle adjustment

- Global refinement of jointly structure (points) and cameras
- Minimize geometric error: $argmin_{\{P_j, X_i\}} \sum_j \sum_i \alpha_{ij}\, d(P_j X_i, x_{ij})$

  here α_{ij} is 1 if X_j visible in view P_j (otherwise 0)

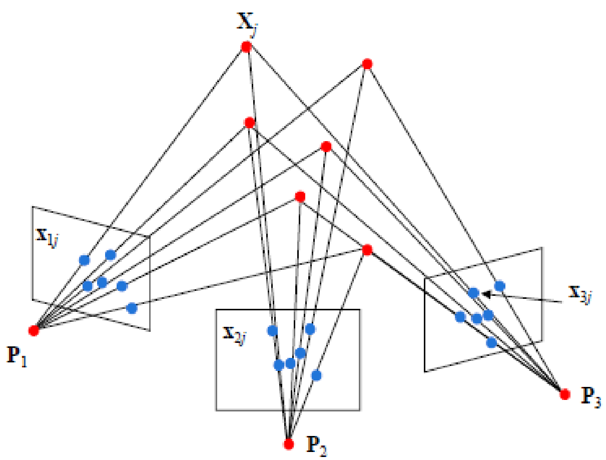
- Non-linear optimization with e.g. Levenberg-Marquard



Reconstruction Algorithm

Generic Outline (calibrated and un-calibrated cameras)

1) Compute robust F/E-matrix between each pair of neighboring views
2) Compute initial reconstruction of consecutive pair of views
3) Compute an initial full 3D reconstruction
4) Bundle-Adjustment to minimize overall geometric error
5) If cameras are not calibrated then perform auto-calibration (also known as self-calibration)



- All is as close to: $x_{ij} = P_j X_i$ for $j = 1 \dots m; i = 1 \dots n$ (algebraic or geometric error)
- But does the reconstruction look already nice?

[See page 453 HZ]

Roadmap for next four lectures

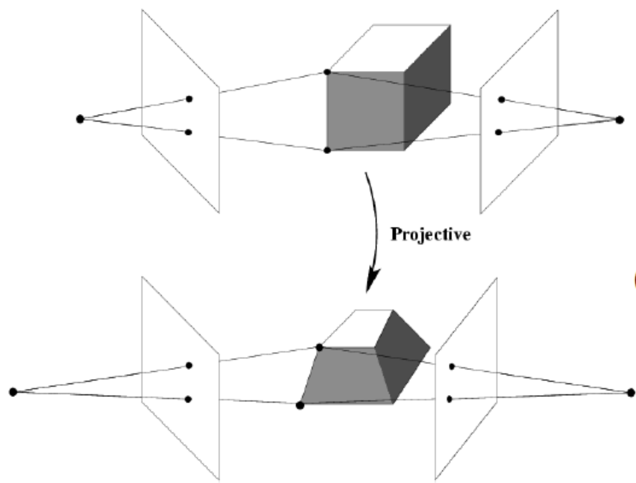
- Appearance-based Matching (sec. 4.1)
- Projective Geometry - Basics (sec. 2.1.1-2.1.4)
- Geometry of a Single Camera (sec 2.1.5, 2.1.6)
  - Camera versus Human Perception
  - The Pinhole Camera
  - Lens effects
- Geometry of two Views (sec. 7.2)
  - The Homography (e.g. rotating camera)
  - Camera Calibration (3D to 2D Mapping)
  - The Fundamental and Essential Matrix (two arbitrary images)
- Robust Geometry estimation for two cameras (sec. 6.1.4)
- Multi-View 3D reconstruction (sec. 7.3-7.4)
  - General scenario
  - From Projective to Metric Space
  - Special Cases

Scale ambiguity

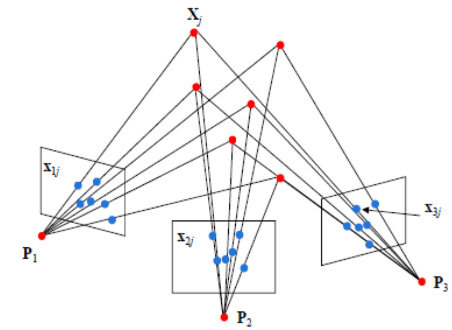
Is the pumpkin 5m or 30cm tall?

Projective ambiguity

We can write (most general): $x_{ij} = P_j X_i = P_j Q^{-1} Q X_i = P_j' X_i'$



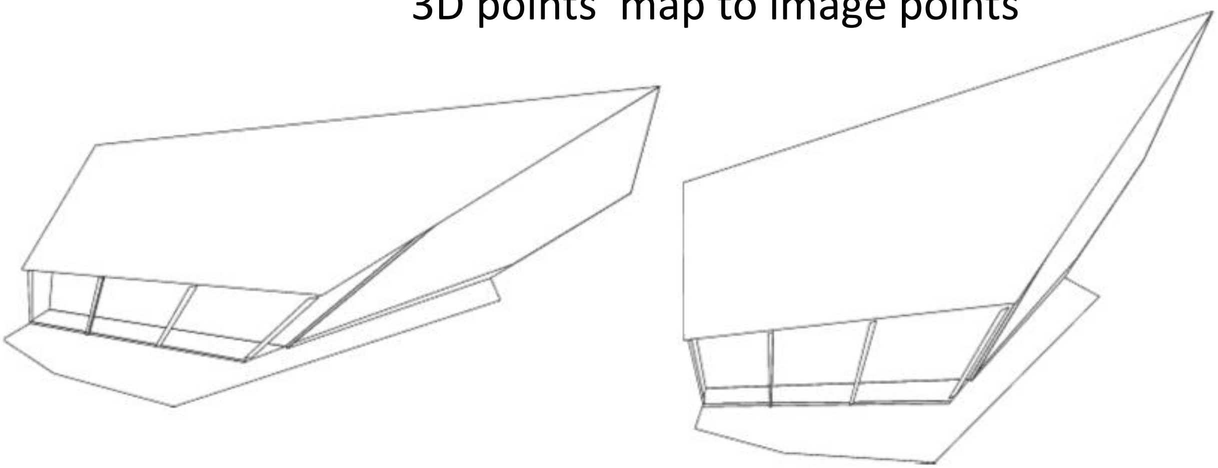
$$\mathbf{Q} = \begin{bmatrix} \mathrm{A} & \mathrm{t} \\ \mathrm{v}^{\mathsf{T}} & v \end{bmatrix}$$



- Q has 15 DoF (projective ambiguity)
- If we do not have any additional information about the cameras or points then we cannot recover Q.
- Possible information (we will see details later)
  - Calibration matrix is same for all cameras
  - External constraints: orthogonal vanishing points

Projective ambiguity

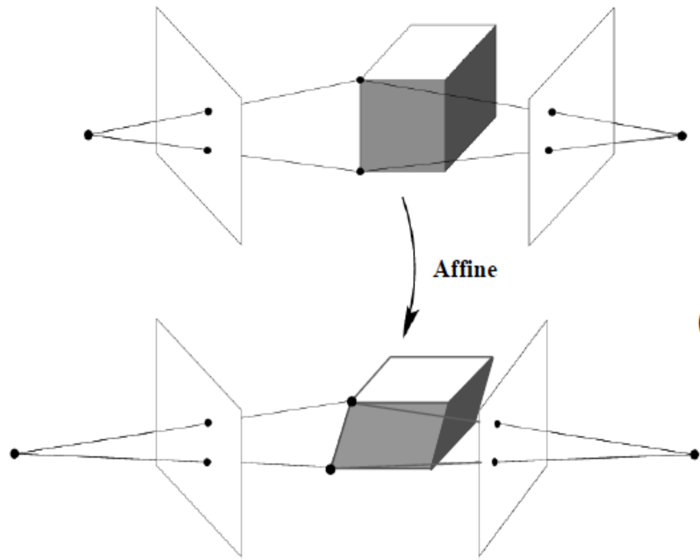
3D points map to image points



This is a “protectively” correct reconstruction
… but not a nice looking one

Affine ambiguity

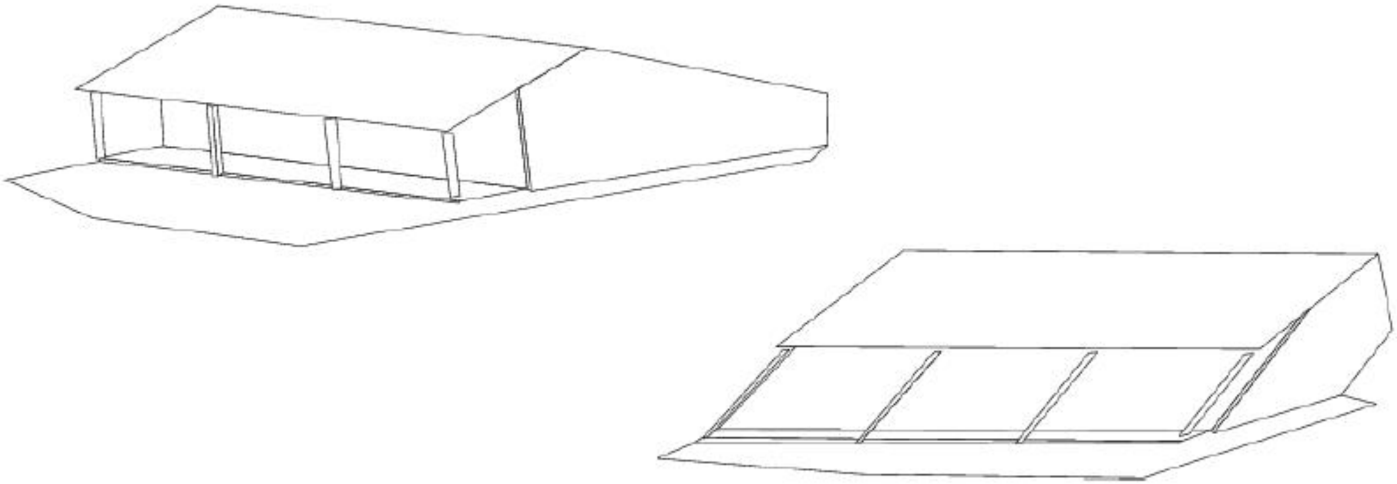
We can write (most general): $x_{ij} = P_j X_i = P_j Q^{-1} Q X_i = P'_j X'_i$



$$\mathbf{Q} = \begin{bmatrix} \mathrm{A} & \mathrm{t} \\ 0^{\mathsf{T}} & 1 \end{bmatrix}$$

- Q has now 12 DoF (affine ambiguity)
- Q leaves the plane at infinity $\pi_\infty = (0,0,0,1)^T$ in place, since any point on π_∞ moves like: $Q(a,b,c,0)^T = (a',b',c',0)$
- Therefore parallel 3D lines stay parallel for any Q

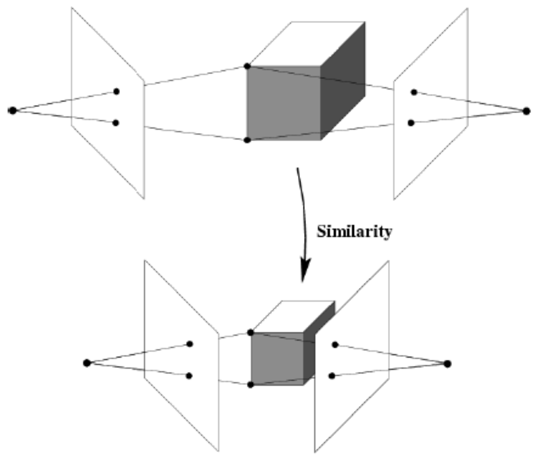
Affine ambiguity



3D Points at infinity stay at infinity

Similarity Ambiguity (Metric space)

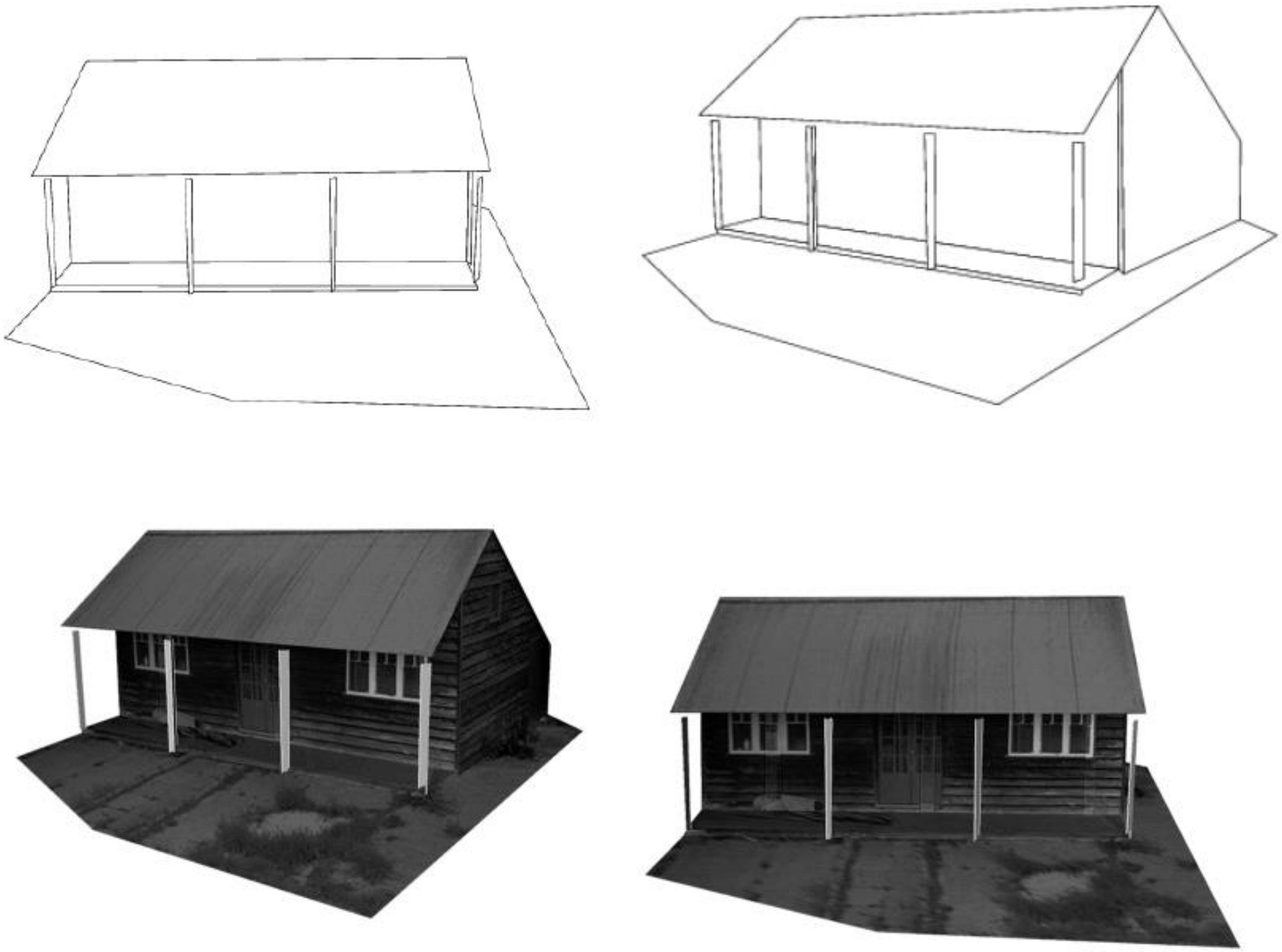
We can write (most general): $x_{ij} = P_j X_i = P_j Q^{-1} Q X_i = P_j' X_i'$



$$\mathbf{Q}_s = \begin{bmatrix} s\mathrm{R} & \mathrm{t} \\ 0^\mathsf{T} & 1 \end{bmatrix}$$

- Q has now 7 DoF (similarity ambiguity)
- Q preserves angles, ratios of lengths, etc.
- For visualization purpose this ambiguity is sufficient. (We often do not need to know if a reconstruction has the size of 1m, 1cm)
- Note, if we do not care about the choice of Q we can set for instance the camera center of first camera to $\tilde{C} = (0,0,0)$.

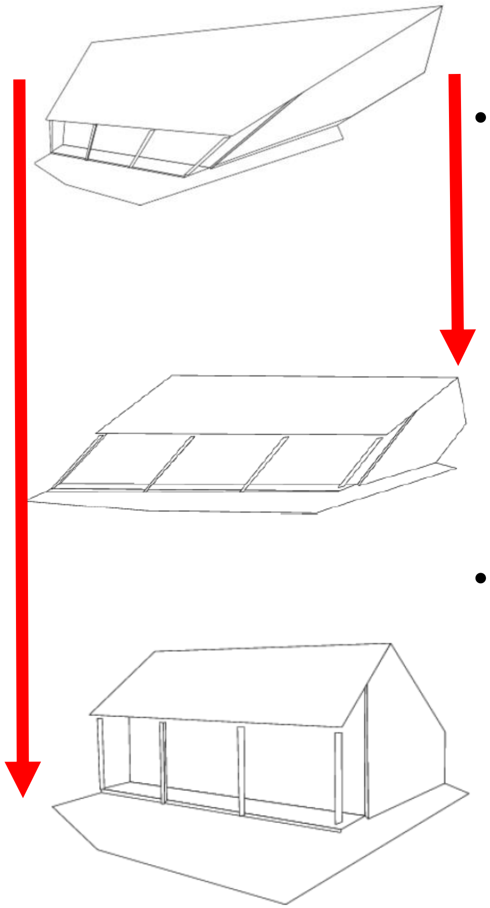
Similarity Ambiguity



How to “upgrade” a reconstruction

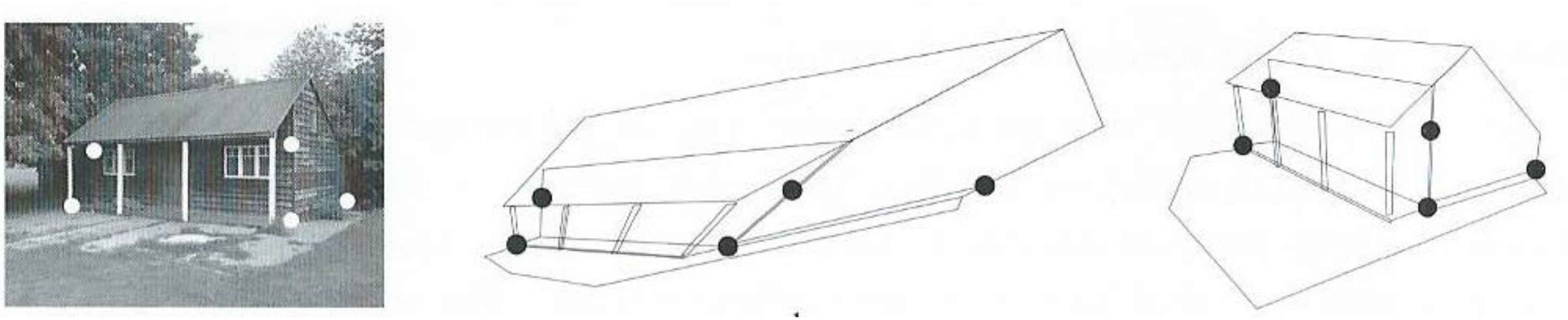
Illustrating some ways to upgrade from Projective to Affine and then to Metric Space
(see details in HZ page 270ff and chapter 19)

- Camera is calibrated
- Calibration from external constraints
  (Example(1): 5 known 3D points)
- Calibration from a mix of in- and external constraints
  (Example(2): single camera and 3 orthogonal vanishing points and a square-pixel camera)
- Calibration from internal constraints only (known as auto-calibration)
  (Examples(3): 2 views with unknown focal lengths)



- Find plane at infinity and move it to canonical position:
  - One of the cameras is affine (3^{rd} of camera matrix is plane at infinity. See HZ page 271)
  - 3 non-collinear 3D vanishing points
- Translational motion (HZ page 268)

Projective to Metric: Direct Method (Example 1)



<u>Given:</u> Five known 3D points (e.g. measured)

<u>Compute Q:</u>

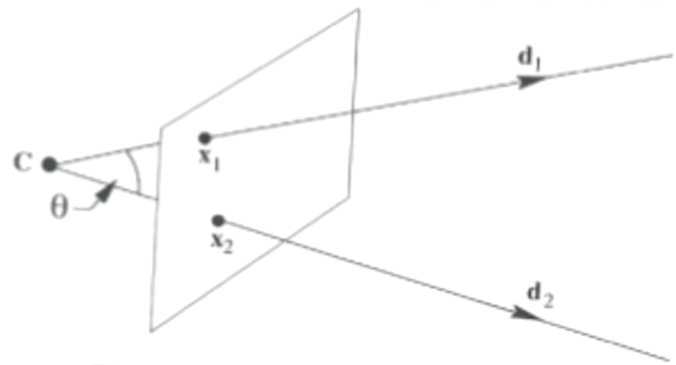
1) $QX_i = X_i'$ (each 3D point gives 3 linear independent equations)
2) 5 points give 15 equations, enough to compute Q (15 DoF) using SVD

<u>Upgrade cameras and points:</u>

$P_j' = P_j Q^{-1}$ and $X_i' = QX_i$ (remember: $x_{ij} = P_j X_i = P_j Q^{-1} Q X_i$)

(Same method as above: "Step 3: Compute initial reconstruction")

But without external knowledge?



- For a camera $P = K\,[I \mid 0]$ the ray outwards is:
  $x = P\,X$ hence $\tilde{X} = K^{-1}x$

- The angle Θ is computed as the normalized rays d_1, d_2:

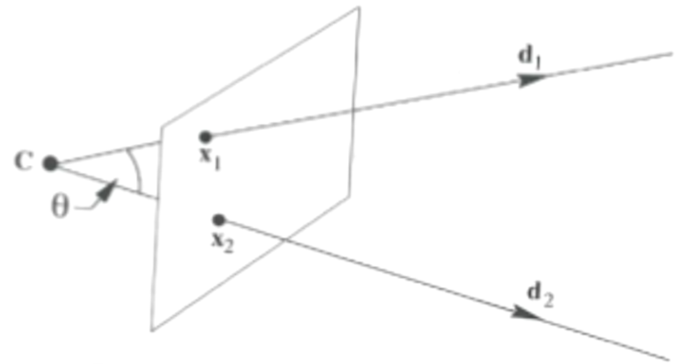
$$\cos\Theta = \frac{d_1^T d_2}{\sqrt{d_1^T d_1}\sqrt{d_2^T d_2}} = \frac{(K^{-1}x_1)^T (K^{-1}x_2)}{\sqrt{(K^{-1}x_1)^T (K^{-1}x_1)}\sqrt{(K^{-1}x_2)^T (K^{-1}x_2)}}$$

$$= \frac{x_1^T \omega\, x_2}{\sqrt{x_1^T \omega x_1}\sqrt{x_2^T \omega x_2}}$$

- We define the matrix: $\omega = K^{-T} K^{-1}$

- Comment: $(K^{-1})^T = (K^T)^{-1} =: K^{-T}$

But without external knowledge?



- We have: $\cos\Theta = \dfrac{x_1^T \omega\, x_2}{\sqrt{x_1^T \omega x_1}\sqrt{x_2^T \omega x_2}}$
- If we were to know ω then we can compute angle Θ
(Comment, if $\Theta = 90^o$ then we have $x_1^T \omega\, x_2$ = 0)
- K can be derived from $\omega = K^{-T} K^{-1}$ using Cholesky decomposition
(see HZ page 582)
- Note, ω depends on K only and not on $R, \tilde{C}$.
Hence it plays a central role in auto-calibration.
- How do we get ω?

Degrees of Freedom of ω

- We have:

$$K = \begin{bmatrix} f & s & p_x \\ 0 & mf & p_y \\ 0 & 0 & 1 \end{bmatrix} \quad \text{then } K^{-1} = \begin{bmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & 1 \end{bmatrix}$$

where a, b, c, d, e are some values that depend on: f, m, s, p_x, p_y

- Then it is:

$$\omega = (K^{-1})^T K^{-1} = \begin{bmatrix} a & 0 & 0 \\ b & d & 0 \\ c & e & 1 \end{bmatrix} \begin{bmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} a^2 & ab & ac \\ ab & b^2 + d^2 & bc + de \\ ac & bc + de & c^2 + e^2 + 1 \end{bmatrix}$$

$$= \begin{bmatrix} \omega_1 & \omega_2 & \omega_3 \\ \omega_2 & \omega_4 & \omega_5 \\ \omega_3 & \omega_5 & \omega_6 \end{bmatrix}$$

- This means that ω has 5 DoF (scale is not unique)

- ω is a 2D conic (see definition of conic from pervious lecture)

Degrees of Freedom of ω (special case)

- Assume we have a "square-pixel" camera, i.e. $m = 1$ and $s = 0$ (practically this is often the case)
- We have:

$$K = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \quad \text{then } K^{-1} = \begin{bmatrix} f^{-1} & 0 & a \\ 0 & f^{-1} & b \\ 0 & 0 & 1 \end{bmatrix}$$

where a, b are some values that depend on: f, p_x, p_y

- Then it is:

$$\omega = (K^{-1})^T K^{-1} = \begin{bmatrix} f^{-1} & 0 & 0 \\ 0 & f^{-1} & 0 \\ a & b & 1 \end{bmatrix} \begin{bmatrix} f^{-1} & 0 & a \\ 0 & f^{-1} & b \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} f^{-2} & 0 & f^{-1}a \\ 0 & f^{-2} & f^{-1}b \\ f^{-1}a & f^{-1}b & a^2 + b^2 + 1 \end{bmatrix}$$

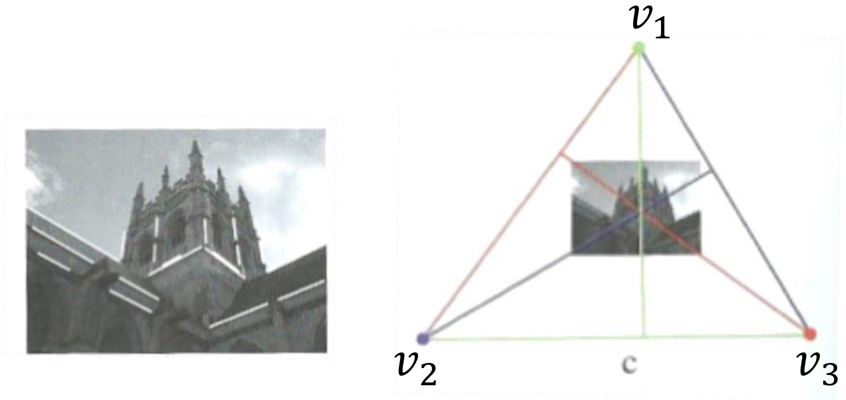
$$= \begin{bmatrix} \omega_1 & 0 & \omega_2 \\ 0 & \omega_1 & \omega_3 \\ \omega_2 & \omega_3 & \omega_4 \end{bmatrix}$$

- This means that ω has 3 DoF (scale is not unique)

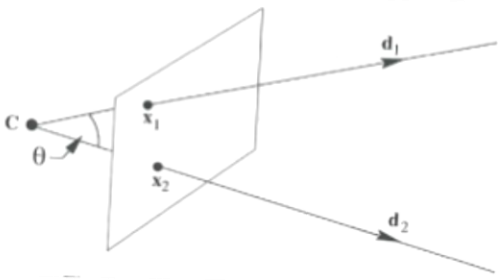
Single Camera: internal + external constraints (Example 2)

- Square pixel cameras (i.e. $m = 1, s = 0$ in K) gives

$$\omega = \begin{bmatrix} \omega_1 & 0 & \omega_2 \\ 0 & \omega_1 & \omega_3 \\ \omega_2 & \omega_3 & \omega_4 \end{bmatrix}$$ with only 3 DoF



- Given 3 image points v_{1-3} that correspond to orthogonal directions
  We know: $v_1^T \omega\, v_2 = 0$; $v_1^T \omega\, v_3 = 0$; $v_2^T \omega\, v_3 = 0$



$$\cos\Theta = \frac{x_1^T \omega\, x_2}{\sqrt{x_1^T \omega x_1}\sqrt{x_2^T \omega x_2}}$$

- This gives a linear system of equations $A\omega = 0$ with A of size 3×4.
  Hence ω can be obtained with SVD

Practically most important case (Example 3)

See HZ, example 19.8 (page 472)

- Assume two cameras with: $s = 0, m = 1, and\ p_x, p_y$ known
- Let us shift images to get $p_x = 0, p_y = 0$

  we get: $Tx = TK\ R\ (I_{3\times3} \mid -C)\ X$

$$T = \begin{bmatrix} 1 & 0 & -p_x \\ 0 & 1 & -p_y \\ 0 & 0 & 1 \end{bmatrix} \text{ and } K = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \text{ then } TK = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Between 2 views we have the so-called Kruppa equations:
  (see explanation in HZ ch. 19.4)

$$\frac{u_1^T \omega_0^{-1} u_1}{\sigma_0^2 v_0^T \omega_1^{-1} v_0} = \frac{u_0^T \omega_0^{-1} u_1}{\sigma_0 \sigma_1 v_0^T \omega_1^{-1} v_1} = \frac{u_0^T \omega_0^{-1} u_0}{\sigma_1^2 v_1^T \omega_1^{-1} v_1}$$

where SVD of $F = [u_0\ u_1\ e_1] \begin{bmatrix} \sigma_0 & 0 & 0 \\ 0 & \sigma_1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} v_0^T \\ v_1^T \\ e_0^T \end{bmatrix}$

and $\omega_i^{-1} = \left(K_i^{-T} K_i^{-1}\right)^{-1} = K_i\ K_i^T = \text{diag}(f_i^2, f_i^2, 1)$

- This can be solved for f_0, f_1 in closed form (see next slide)

The solution for f_0, f_1

$$\frac{a + b f_0^2}{c + d f_1^2} \overset{(1)}{=} \frac{a' + b' f_0^2}{c' + d' f_1^2} \overset{(2)}{=} \frac{a'' + b'' f_0^2}{c'' + b'' f_1^2}$$

(change coord)

$$\Rightarrow a + b f_1^2 + c f_0^2 + d f_0^2 f_1^2 = 0 \quad (1)$$

$$a' + b' f_1^2 + c' f_0^2 + d' f_0^2 f_1^2 = 0 \quad (2)$$

set $x := f_0^2$ $y := f_1^2$

$$(1) \quad a + by + cx + dxy = 0 \Rightarrow x = \frac{-a - by}{c + dy}$$

$$(2) \quad a' + b' y + c' x + d' xy = 0$$

put (1) in (2)

$$a' + b' y + \frac{c'(-a-by)}{c + dy} + d' y \left(\frac{-a - by}{c + dy} \right) = 0 \quad \Big| \cdot (c + dy)$$

$$\Rightarrow a + by + cy^2 = 0$$

$$\Rightarrow y = \pm \sqrt{d}$$

$$\Rightarrow f_1 = \pm \sqrt{\pm \sqrt{d}} = d^{1/4} \quad \text{since } f_1 \text{ positive}$$

from (1): $a + b f_0^2 = 0 \Rightarrow f_0 = \sqrt{-\frac{a}{b}}$

Auto-Calibration: Only internal constraints

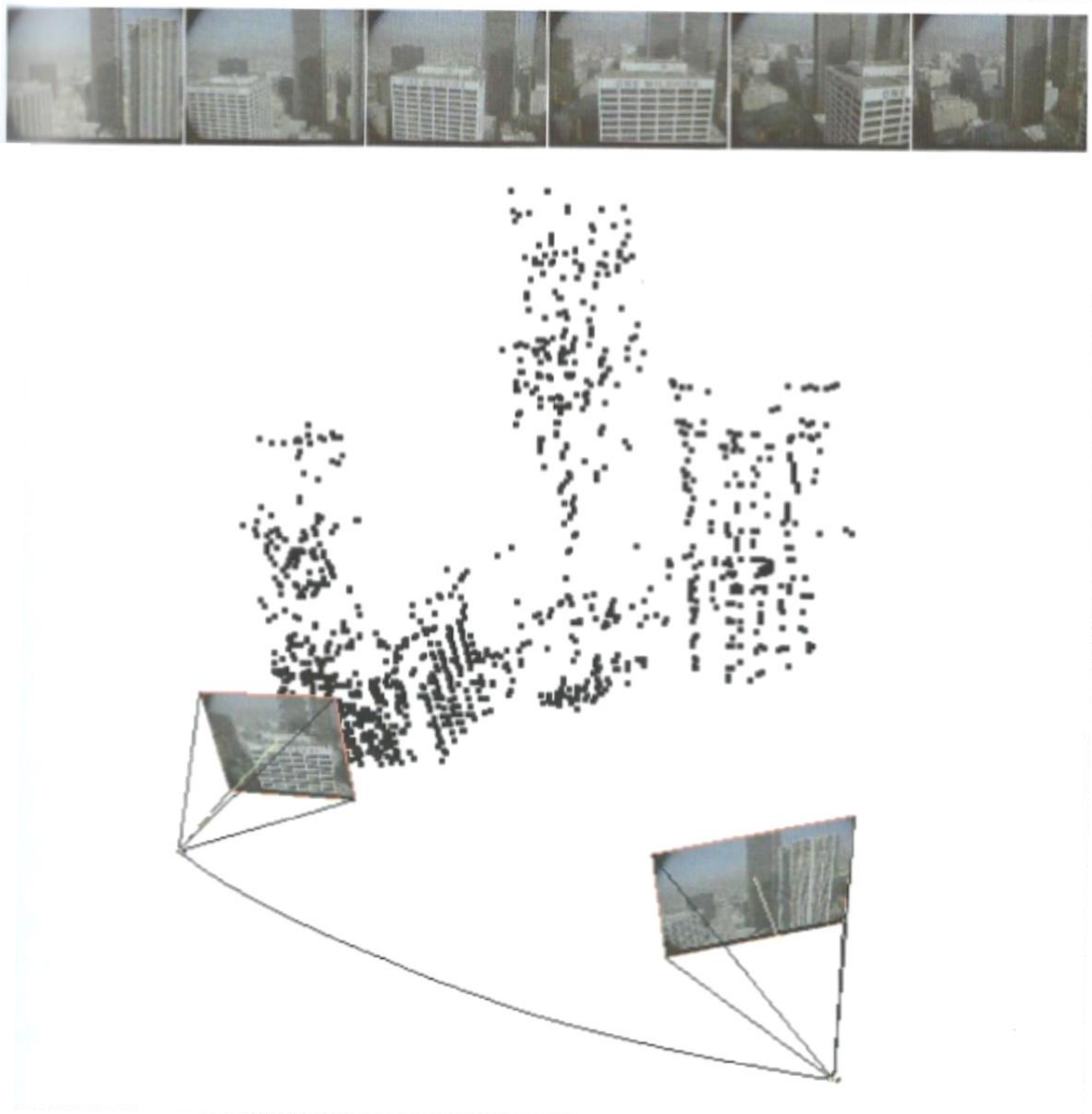
- Chapter 19 HZ
- Insight: Multiple views automatically give extra constraints (not discussed here)

Condition	fixed intrinsic	known intrinsic	views m
Constant internal parameters	5	0	3
Aspect ratio and skew known, focal length and principal point vary	0	2	4*
Aspect ratio and skew constant, focal length and principal point vary	2	0	5*
Skew zero, all other parameters vary	0	1	8*
p.p. known all other parameters vary	0	2	4*, 5(linear)
p.p. known skew zero	0	3	3(linear)
p.p., skew and aspect ratio known	0	4	2, 3(linear)

Remember: We have 5 intrinsic parameters:

$$K = \begin{bmatrix} f & s & p_x \\ 0 & mf & p_y \\ 0 & 0 & 1 \end{bmatrix}$$

Example – Reconstruction from a Video



Building Rome in a day – Reconstruction from Flickr

[Agarwal, Snavely, Simon, Seitz, Szeliski; ICCV 2009]

Roadmap for next four lectures

- Appearance-based Matching (sec. 4.1)

- Projective Geometry - Basics (sec. 2.1.1-2.1.4)

- Geometry of a Single Camera (sec 2.1.5, 2.1.6)
  - Camera versus Human Perception
  - The Pinhole Camera
  - Lens effects

- Geometry of two Views (sec. 7.2)
  - The Homography (e.g. rotating camera)
  - Camera Calibration (3D to 2D Mapping)
  - The Fundamental and Essential Matrix (two arbitrary images)

- Robust Geometry estimation for two cameras (sec. 6.1.4)

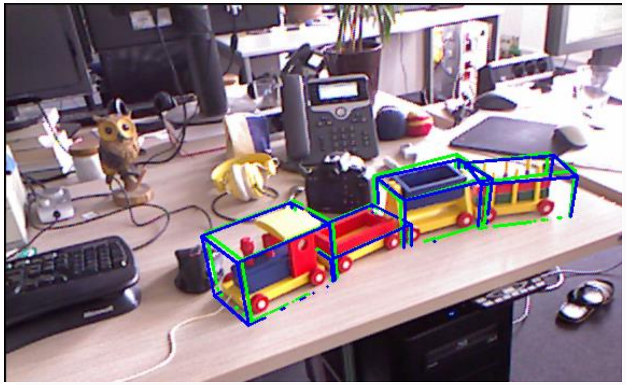
- Multi-View 3D reconstruction (sec. 7.3-7.4)
  - General scenario
  - From Projective to Metric Space
  - Special Cases (skip – see slides at the end)

Decision Trees in Computer Vision

Classification forests

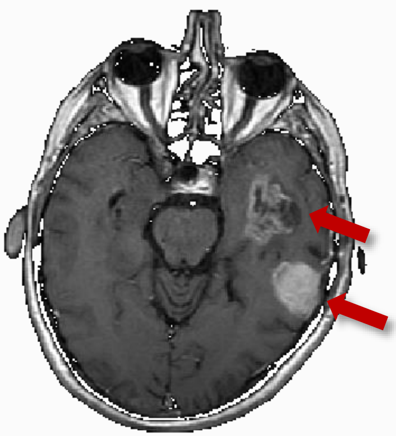
e.g. semantic segmentation

Regression forests



e.g. object localization

Density forests



e.g. novelty detection

Manifold forests

e.g. dimensionality reduction

Semi-supervised forests

e.g. semi-sup. semantic segmentation

Mushroom example


eatable


not eatable


eatable

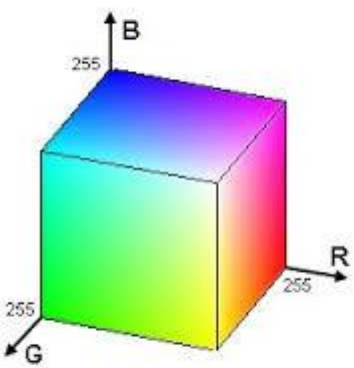

not eatable


not eatable


not eatable

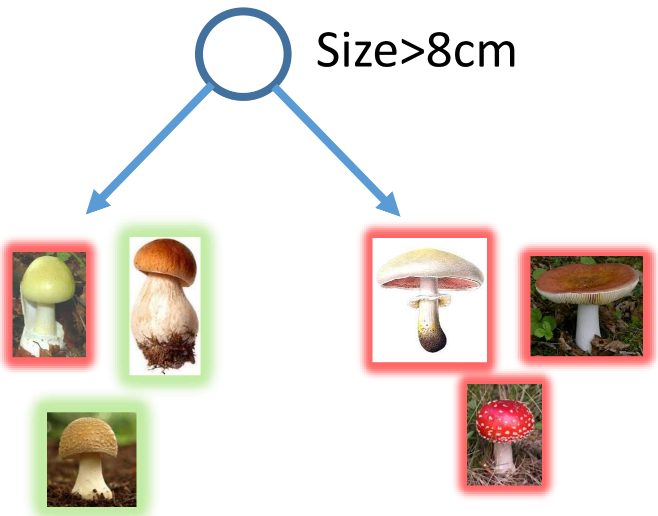
Measure attributes:

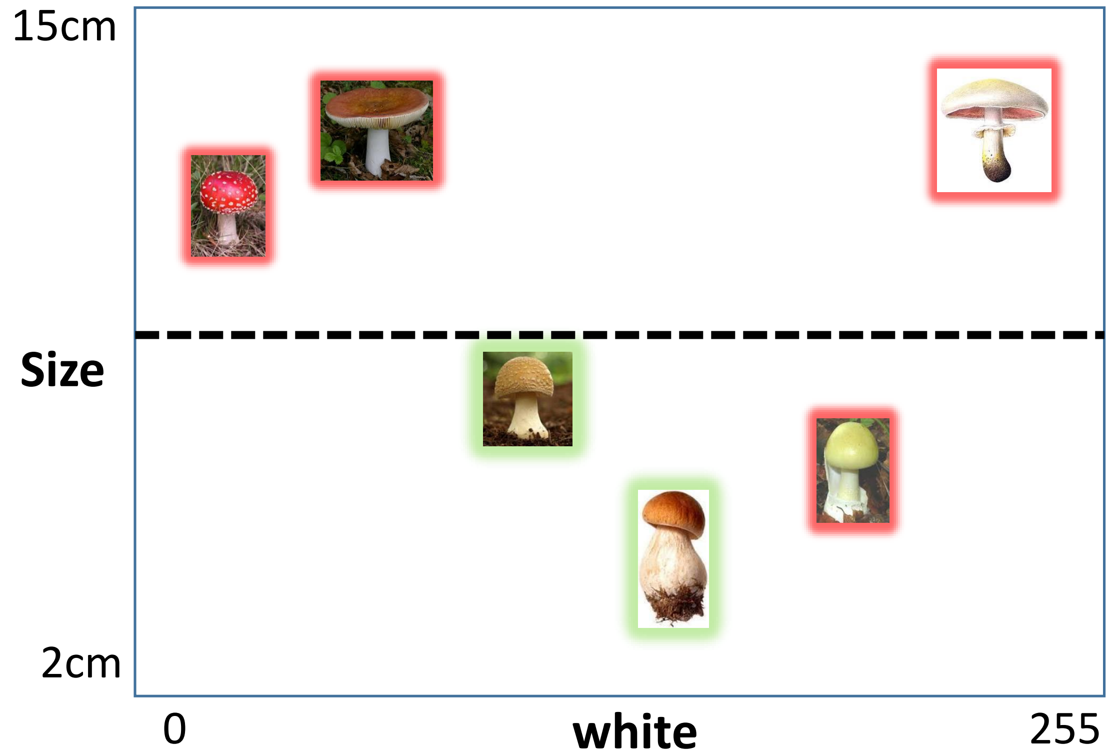
- Size in centimeters
- Color: Average “whitness” of the mushroom (i.e. value along diagonal in RGB cube)



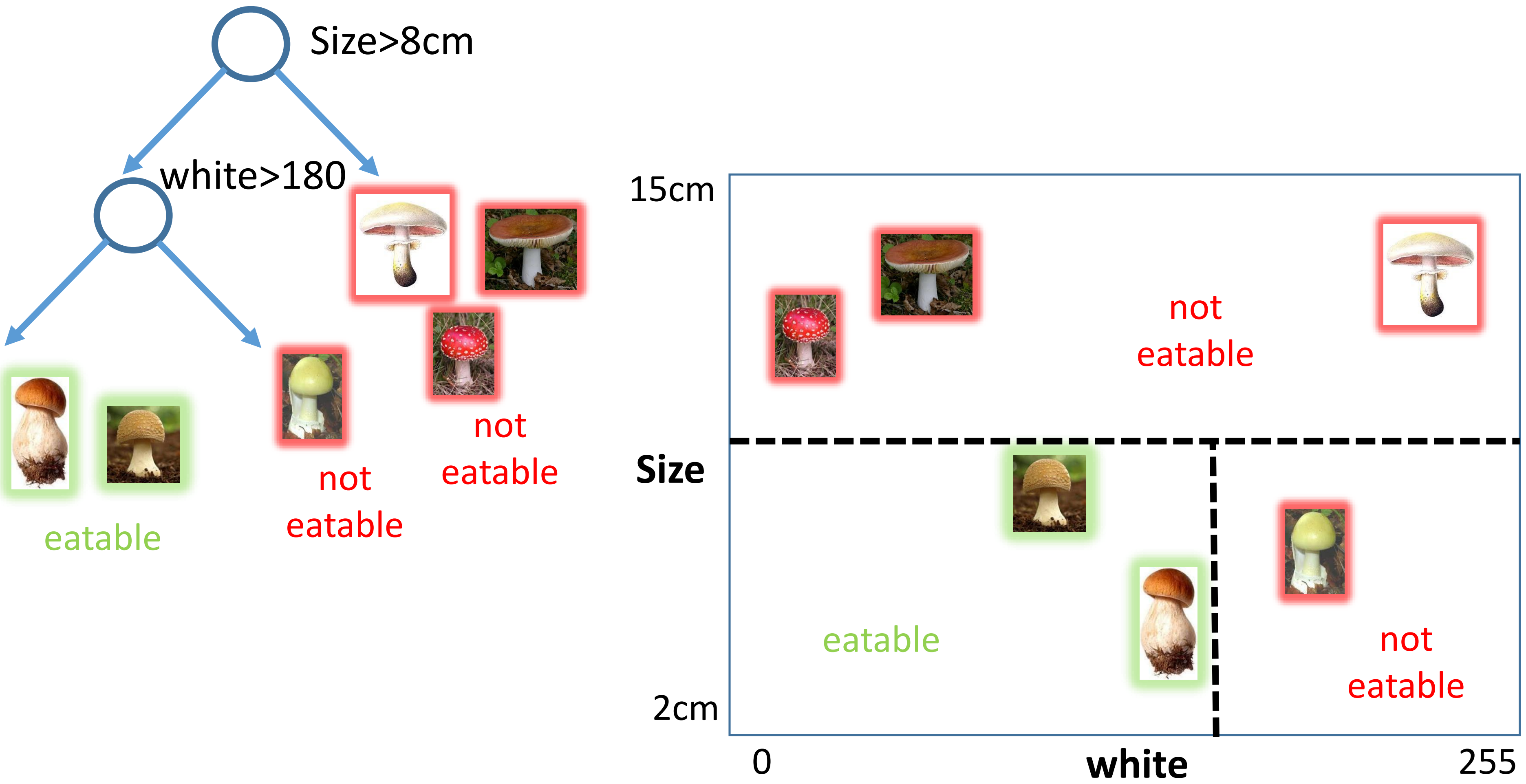
Task: Build a decision tree such that you can distinguish eatable form not eatable mushrooms.

Decision Tree

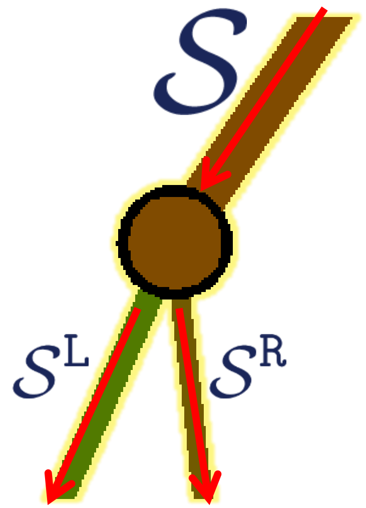


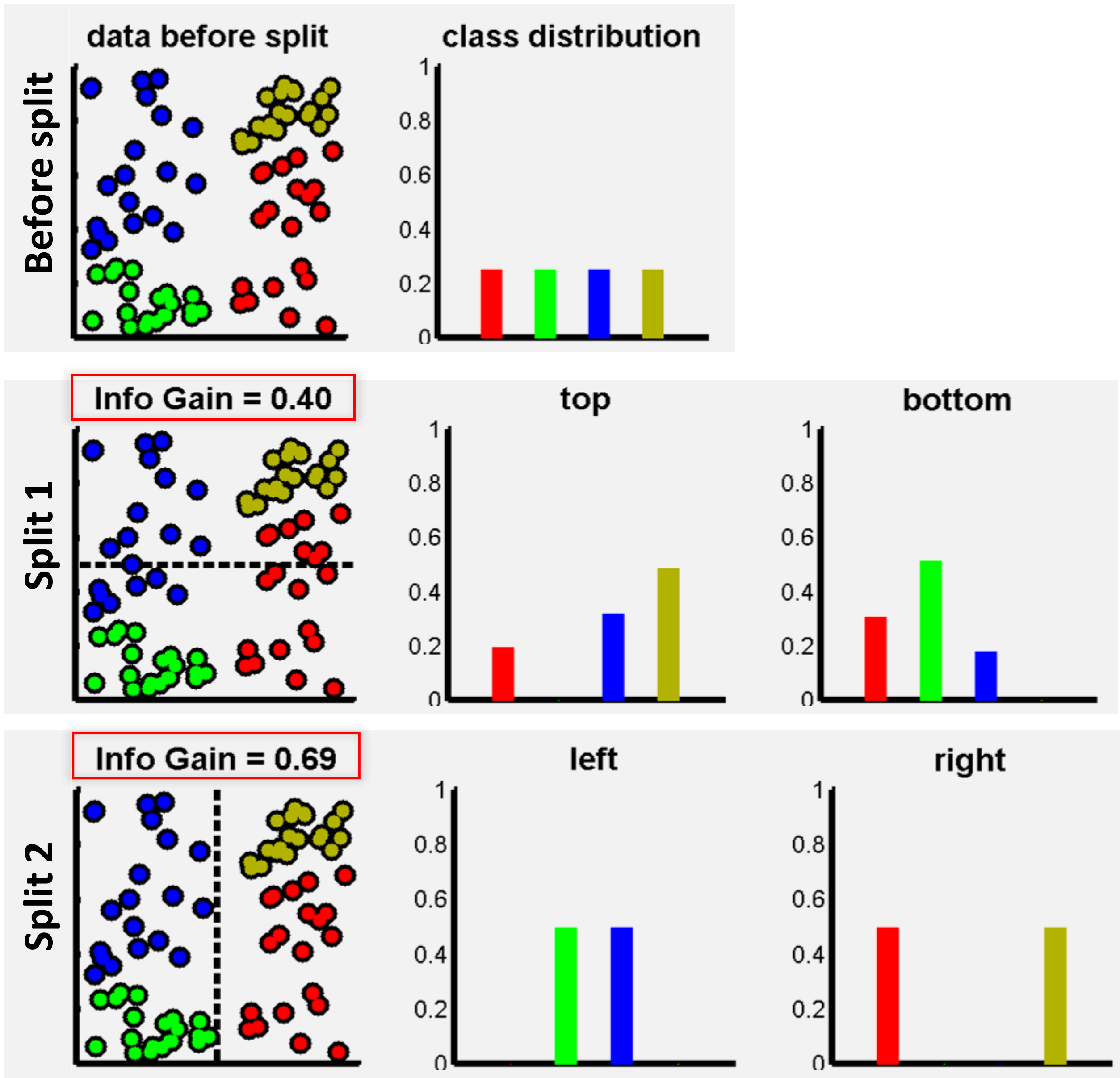


Decision Tree



Decision Tree – Split Criteria





Information gain

$$I(\mathcal{S},\boldsymbol{\theta}) = H(\mathcal{S}) - \sum_{i\in\{\mathrm{L},\mathrm{R}\}} \frac{|\mathcal{S}^i|}{|\mathcal{S}|} H(\mathcal{S}^i)$$

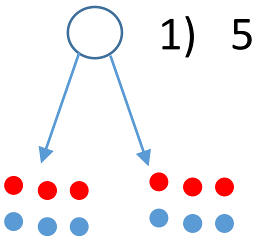
Shannon's entropy

$$H(\mathcal{S}) = -\sum_{c\in\mathcal{C}} p(c)\log(p(c))$$

Think of minimizing Entropy

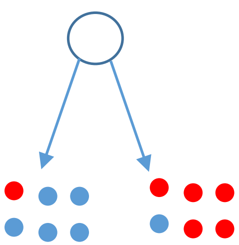
Decision Tree – Split Criteria

- We have $|S| = 12$
- In S we have 6 red and 6 blue points (2 classes)
- $H(S) = -(0.5\log(0.5) + 0.5\log(0.5)) = 1$
- We look at two possible splits
  (in both cases we happen to have $|S^L| = 6$ and $|S^R| = 6$, but could be different)



1) 50%-50% class-split (each side (S^L and S^R) gets 3 red and 3 blue)

$$H(S^L) = -(0.5\log(0.5) + 0.5\log(0.5)) = 1$$
$$H(S^R) = -(0.5\log(0.5) + 0.5\log(0.5)) = 1$$
$$I(S) = H(S) - (0.5 + 0.5) = H(S) - 1 = 0$$ (Lower information gain)



2) 16%-84% class-split (right side has 5 red and 1 blue, left side has 5 blue and 1 red)

$$H(S^L) = -\left(\frac{1}{6}\log\left(\frac{1}{6}\right) + \frac{5}{6}\log\left(\frac{5}{6}\right)\right) = 0.64$$
$$H(S^R) = -\left(\frac{1}{6}\log\left(\frac{1}{6}\right) + \frac{5}{6}\log\left(\frac{5}{6}\right)\right) = 0.64$$
$$I(S) = H(S) - (0.5 * 0.64 + 0.5 * 0.64) = H(S) - 0.64 = 0.36$$

(Higher information gain)

Generalization

Training Data:

eatable

not eatable

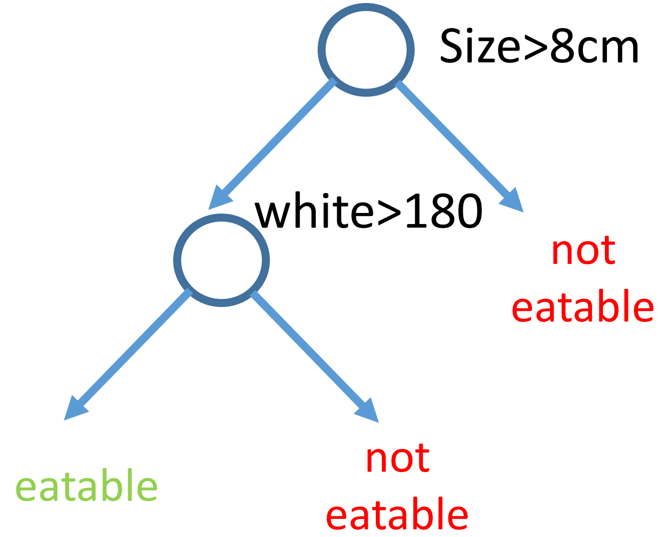
eatable

not eatable

not eatable

not eatable

Test Data:



System is optimal!

Generalization

Training Data:


eatable


not eatable


eatable


not eatable

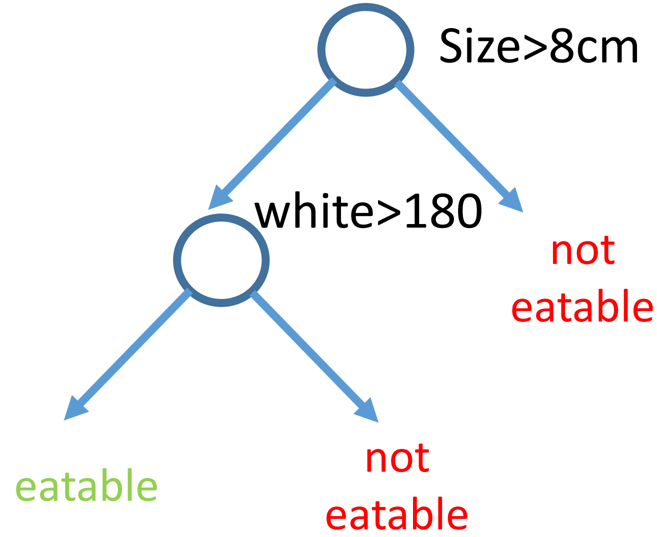

not eatable


not eatable

Test Data:

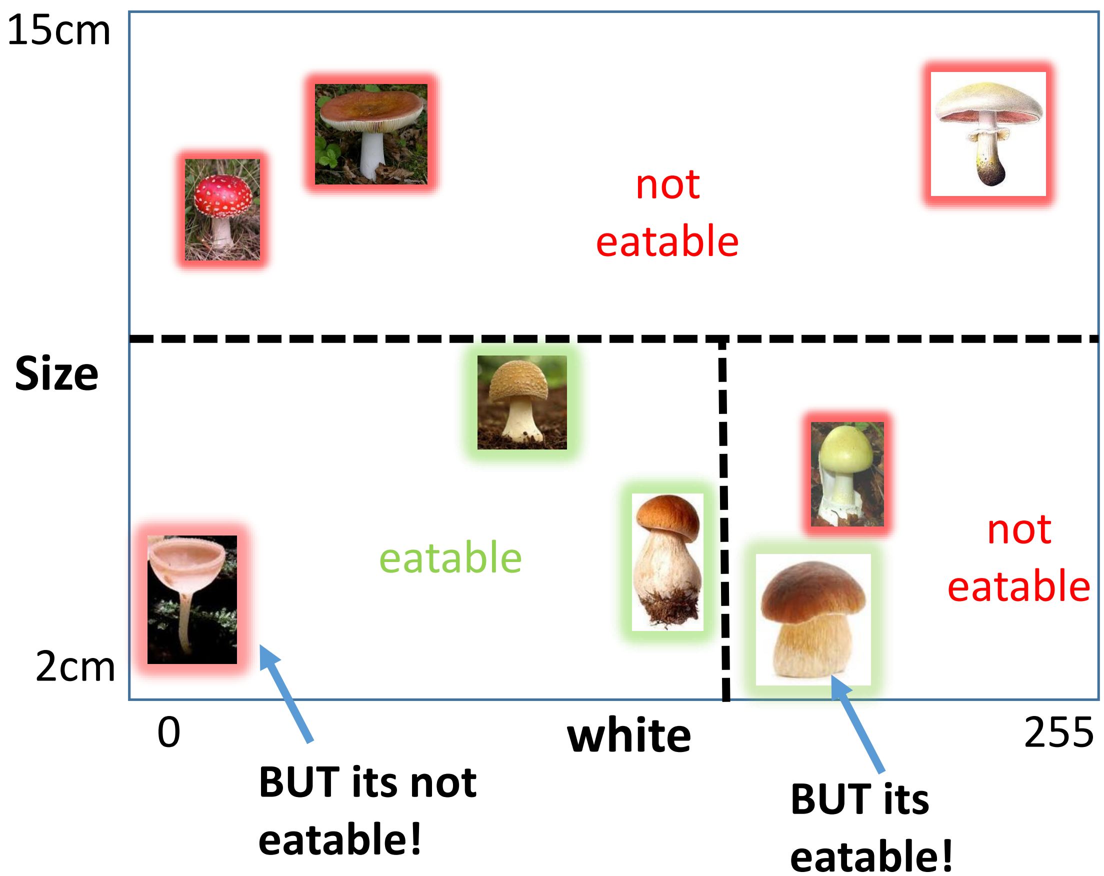

eatable


not eatable



System may not be optimal!

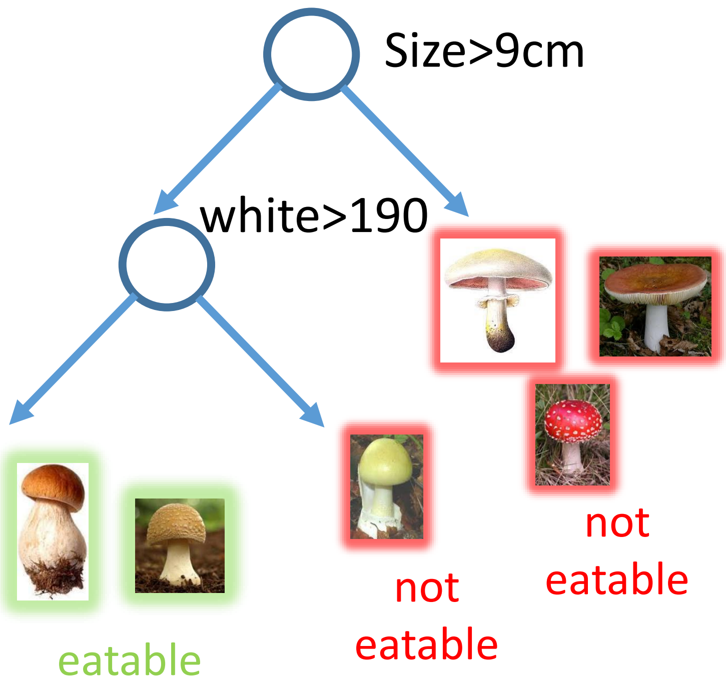
Generalization

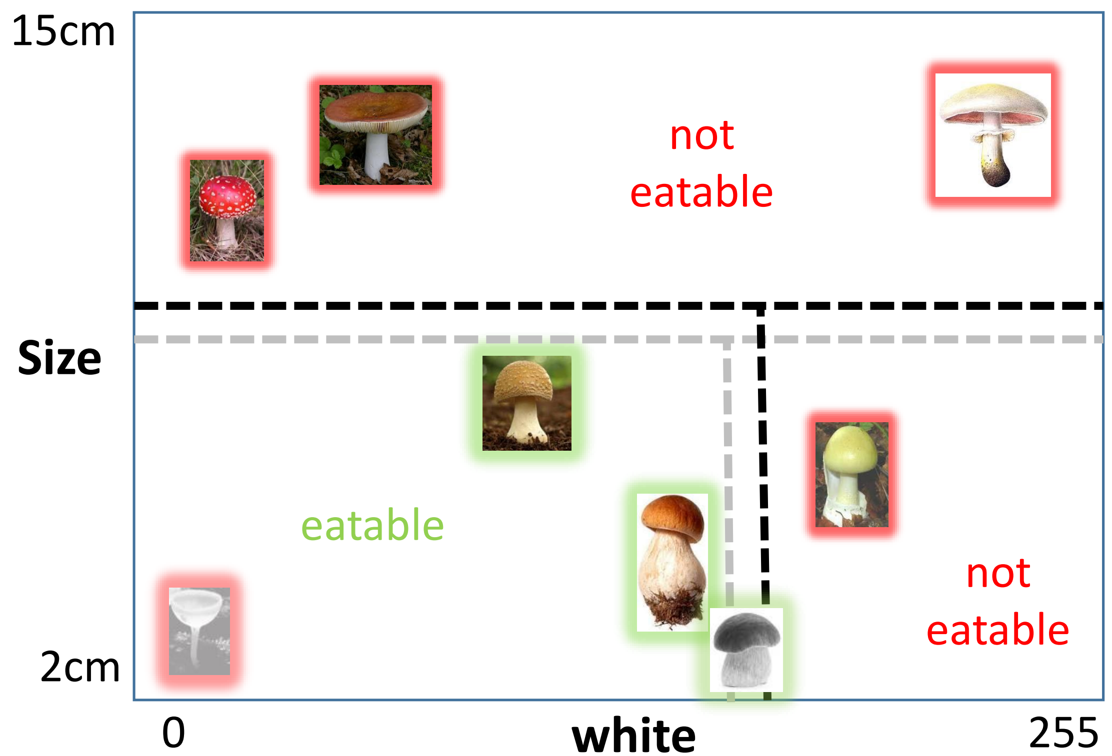


Definition

- **Over-fitting**: Is the effect that the model perfectly memorizes the training data, but does not perform well on test data
- **Generalization**: One of the most important aspect of a model is its ability to generalize. That means that new (unseen) test data is correctly classified. A model which overfitts does not generalize well.
- **How to avoid over-fitting?**
  - Idea 1: Where to place decision boundary?
  - Idea 2: Do not make the trees too deep

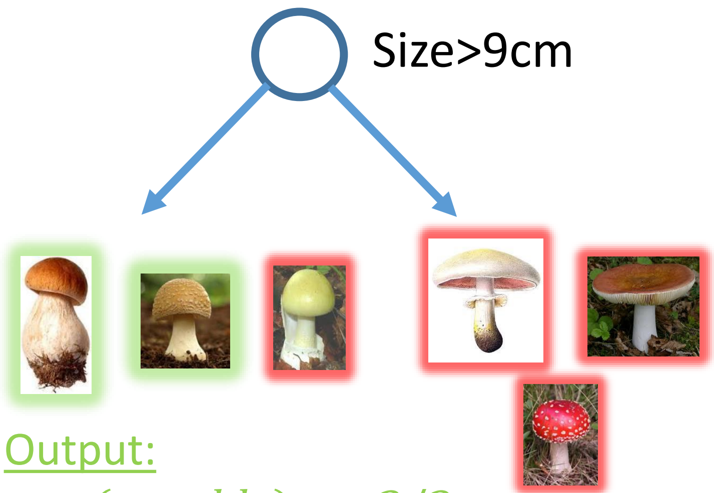
Decision Boundary





Place "decision boundary" such that the distance to all (some) examples is maximized!

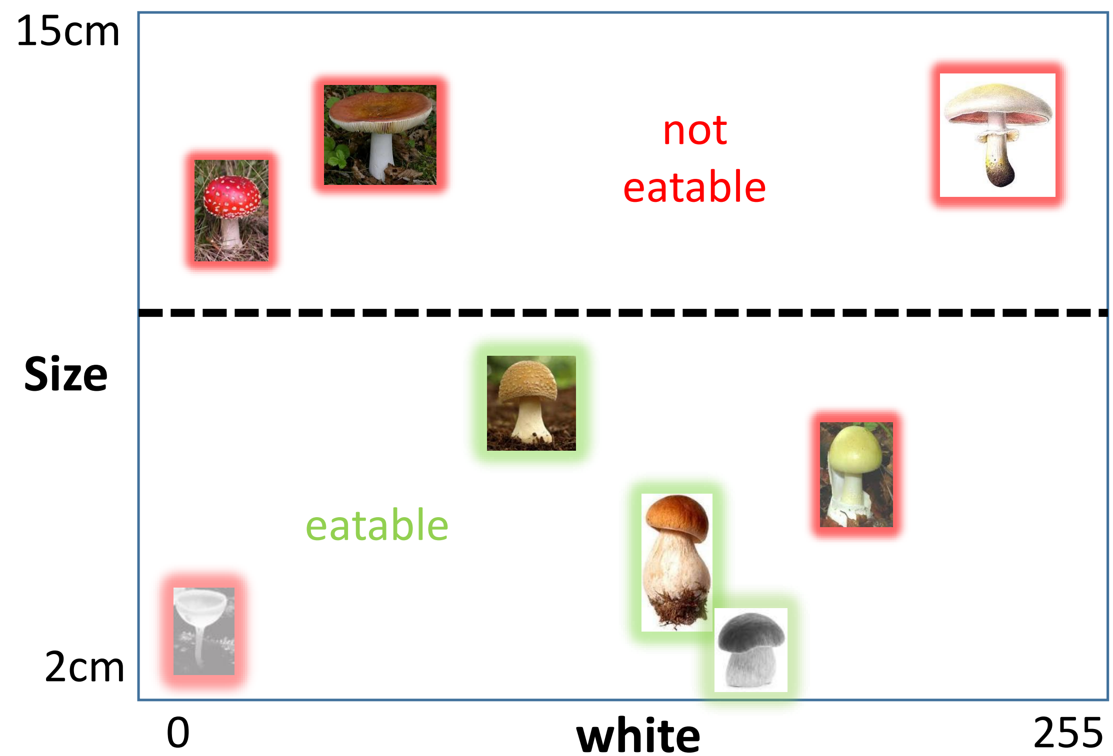
Tree Depth



Output:

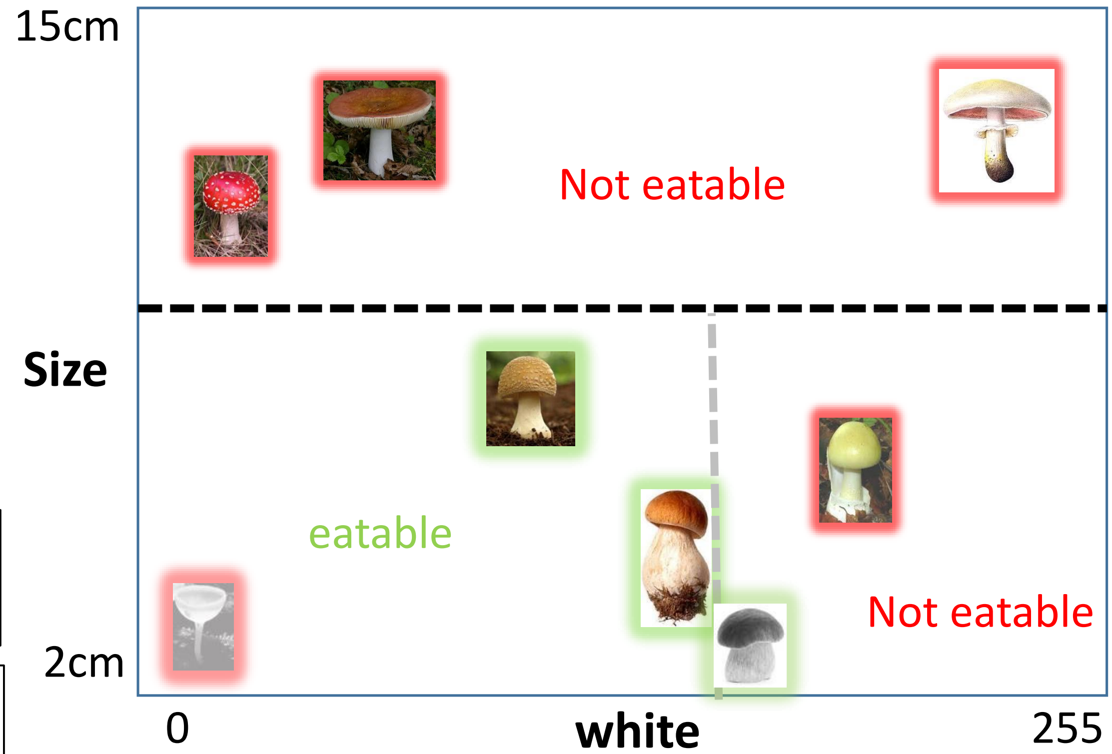
$$p(eatable) = 2/3$$
$$p(not\ eatable) = 1/3$$

not eatable



Tree Depth - Comparison

Test data:



System with 2 split-nodes makes:
0 mistake training; 2 mistakes testing

System with 1 split-nodes makes:
1 mistake training; 2 mistakes testing

Conclusion: A system which makes less mistakes during training may not be the better system at test time!

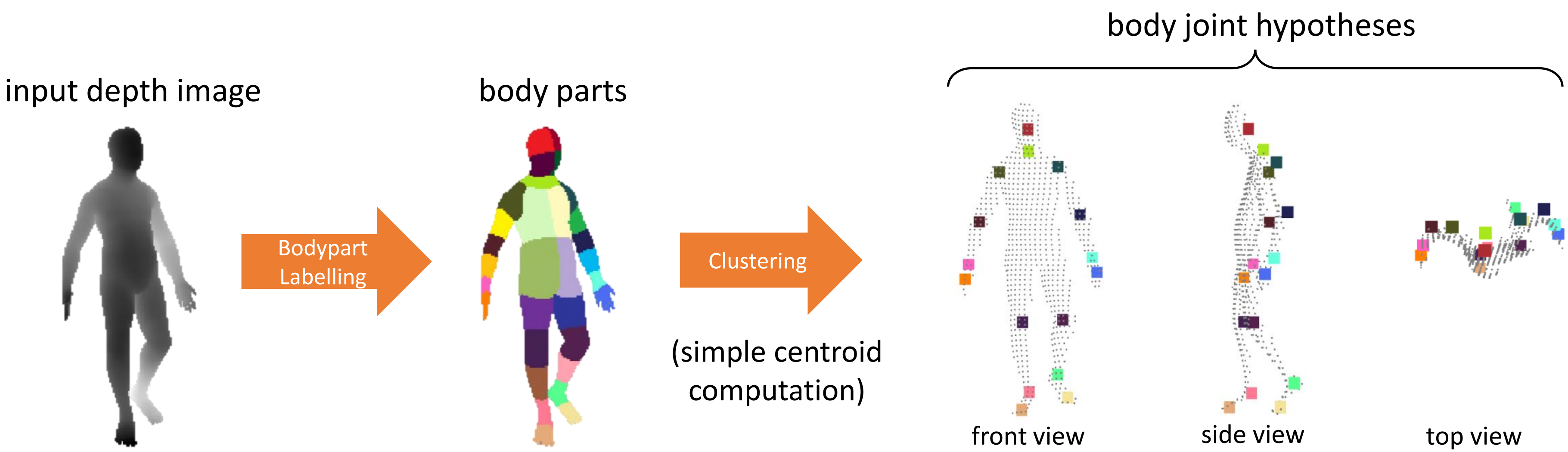
Example: People tracking

Depth
camera

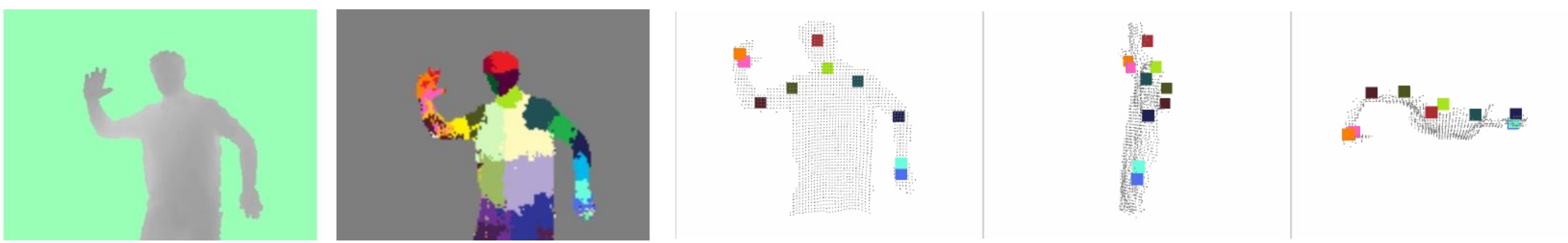
… what runs on Microsoft Xbox

[J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. **Real-Time Human Pose Recognition in Parts from a Single Depth Image**. *In Proc. IEEE CVPR*, June 2011.]

Example: People tracking



Body is divided
into 31 body parts



Train on synthetic data – test on real data

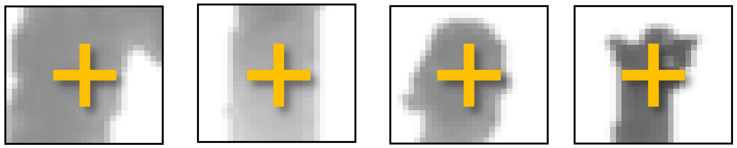
Synthetic (graphics)
Train data

Real (hand-labelled)
Test Data

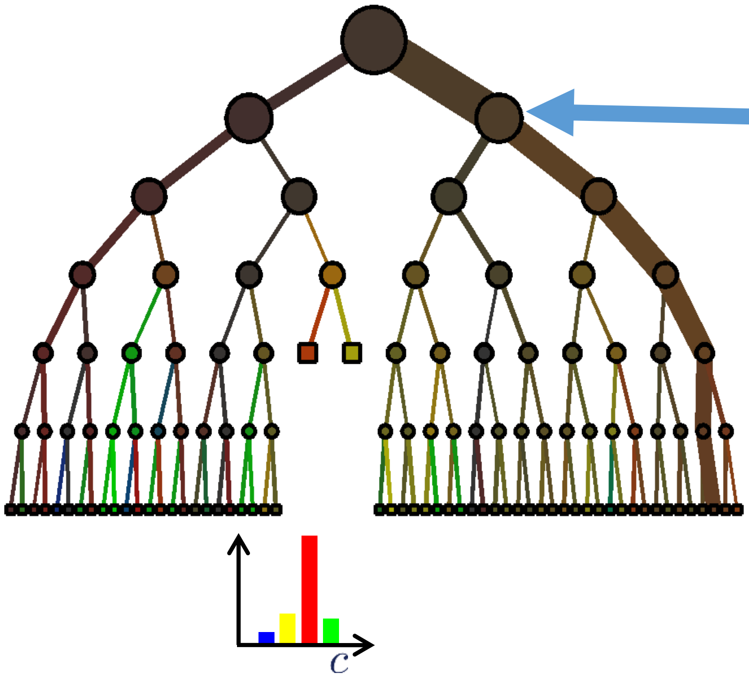
Decision Tree

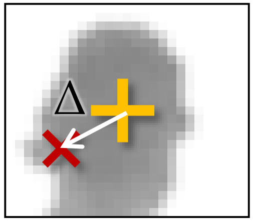
Input image

Output labeling (each pixel shows most probable labeling)



Classify each pixel (yellow cross) independently

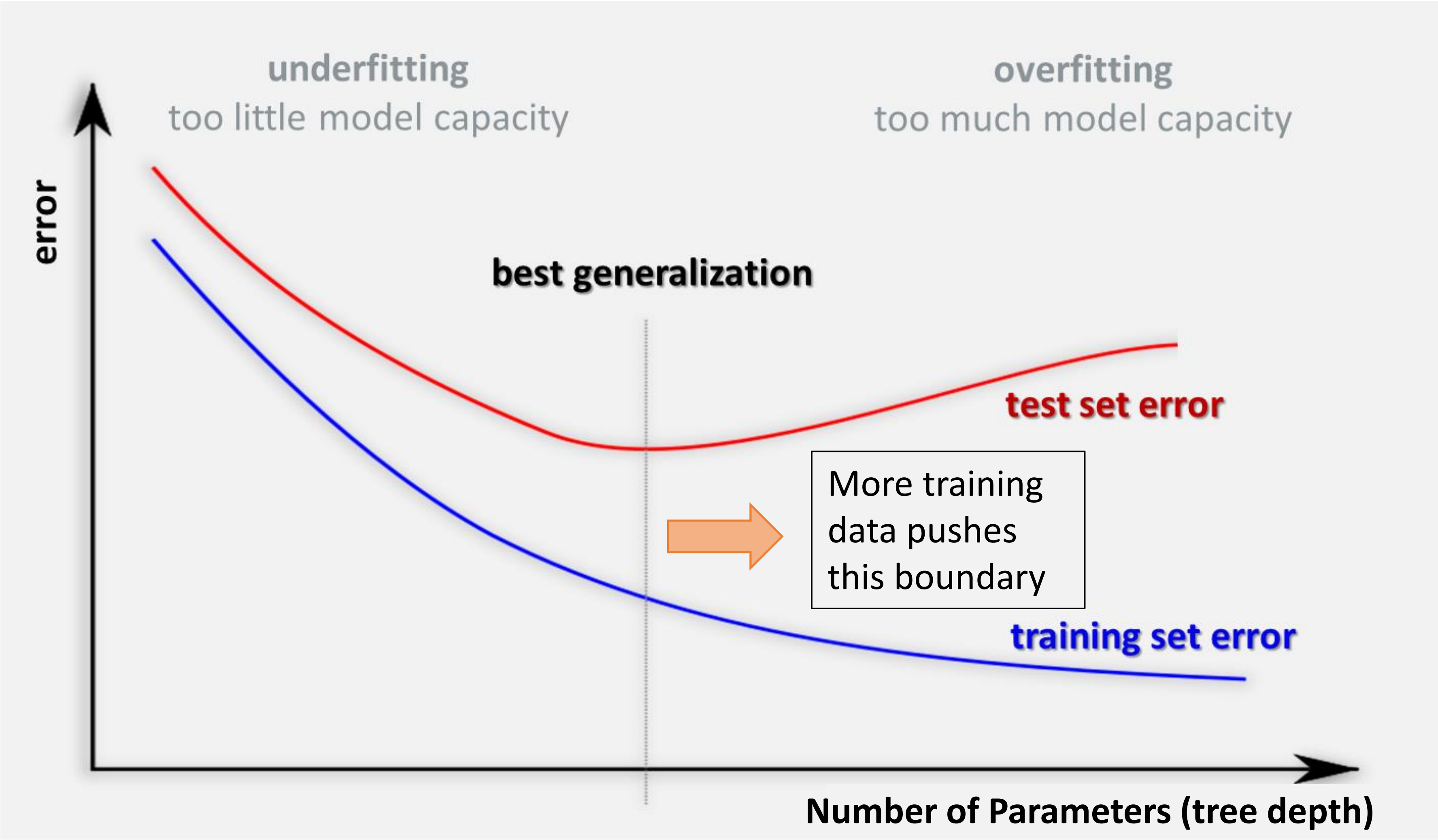




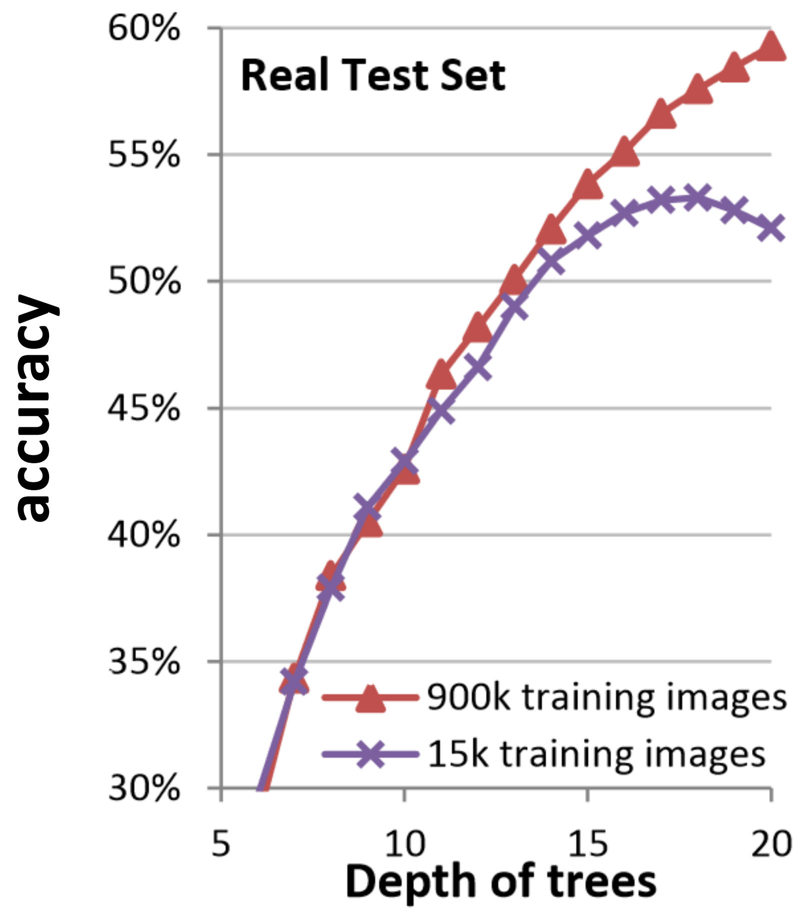
Simple feature test:
Depth value at red pixel > threshold?
(Optimize over Δ and threshold with Information Gain)

Each leaf stores a distribution over the 31 body parts

Tree Depth

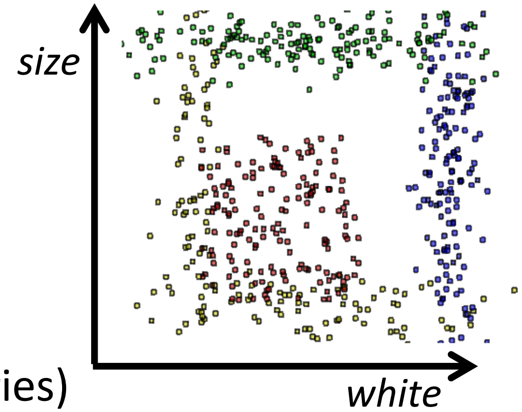

underfitting
too little model capacity
overfitting
too much model capacity
error
best generalization
test set error
More training data pushes this boundary
training set error
Number of Parameters (tree depth)

Real System performance that runs on Xbox



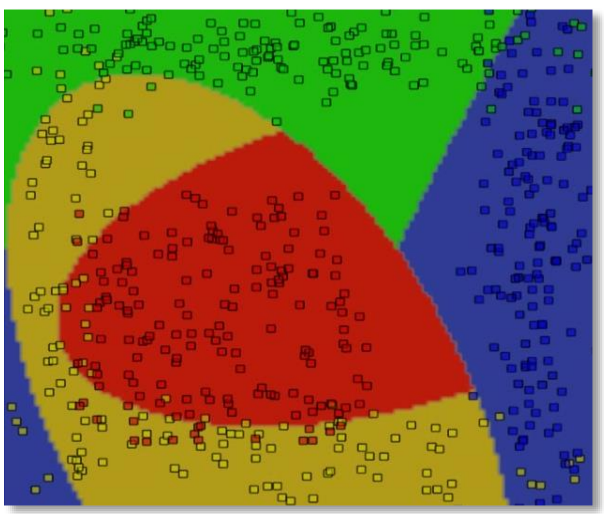
Amount of Training Data is one of the main factors

Example: Overfitting with tree depth

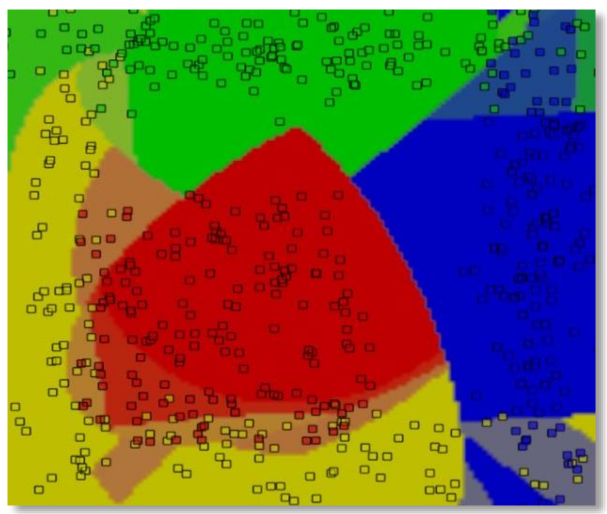


Mushroom example with 4 classes (e.g. how tasty)

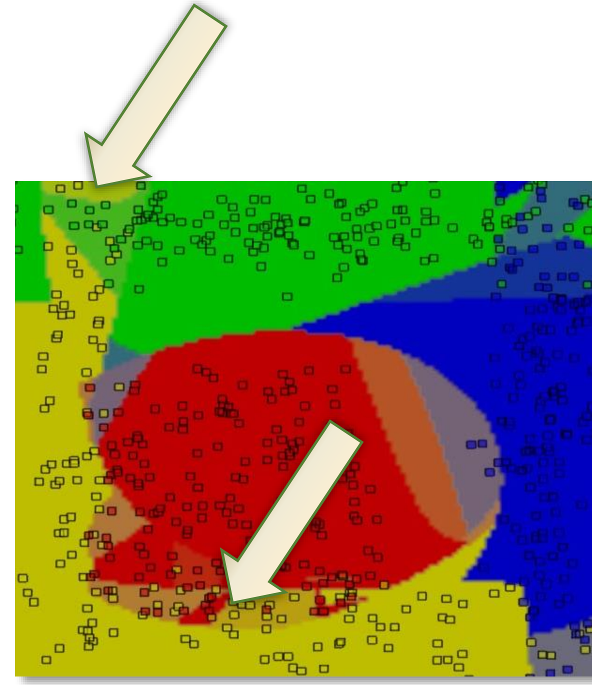
(non-linear decision boundaries)



Depth 3



Depth 6

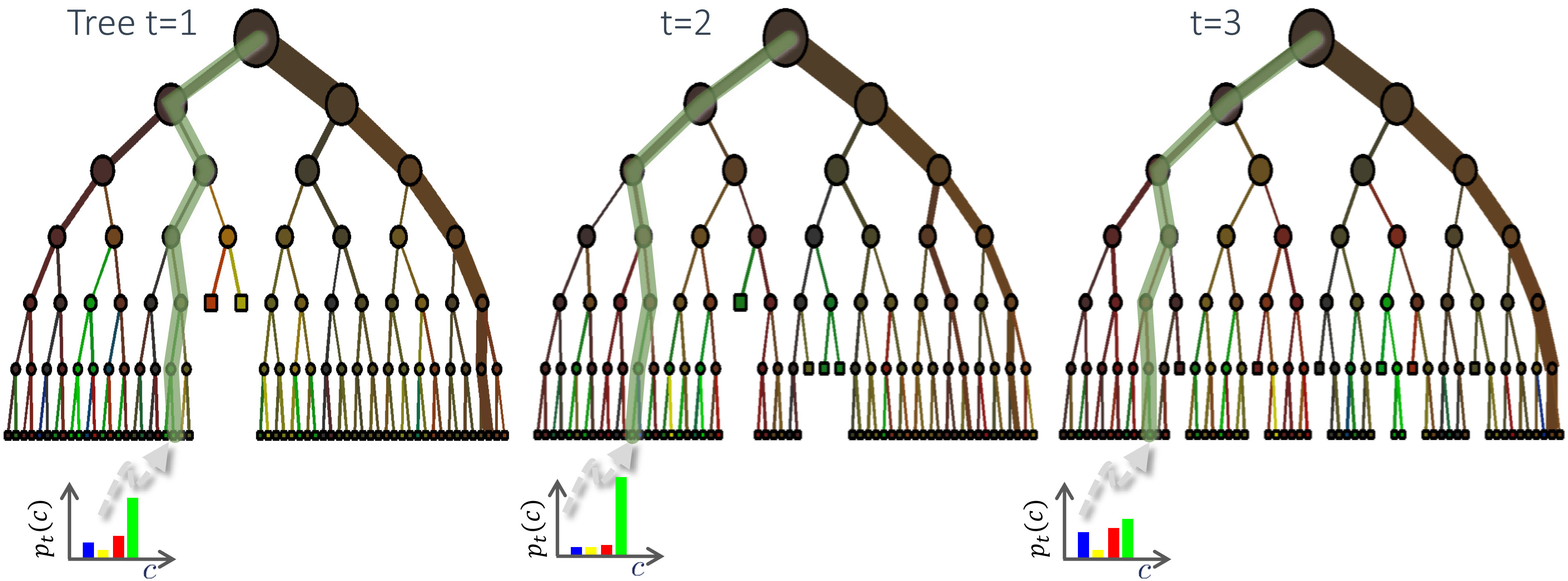


Depth 15

underfitting overfitting

Decision Forest for better Decision Boundary

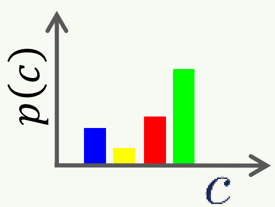
(Each tree is trained with a different subset of the training data)



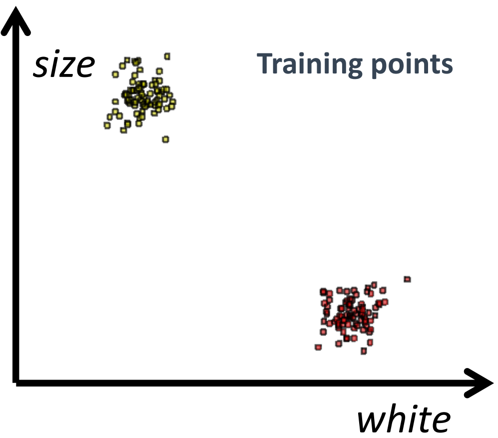
The ensemble model

Forest output probability $p(c) = \frac{1}{T}\sum_{t}^{T} p_t(c)$

T is the number of trees

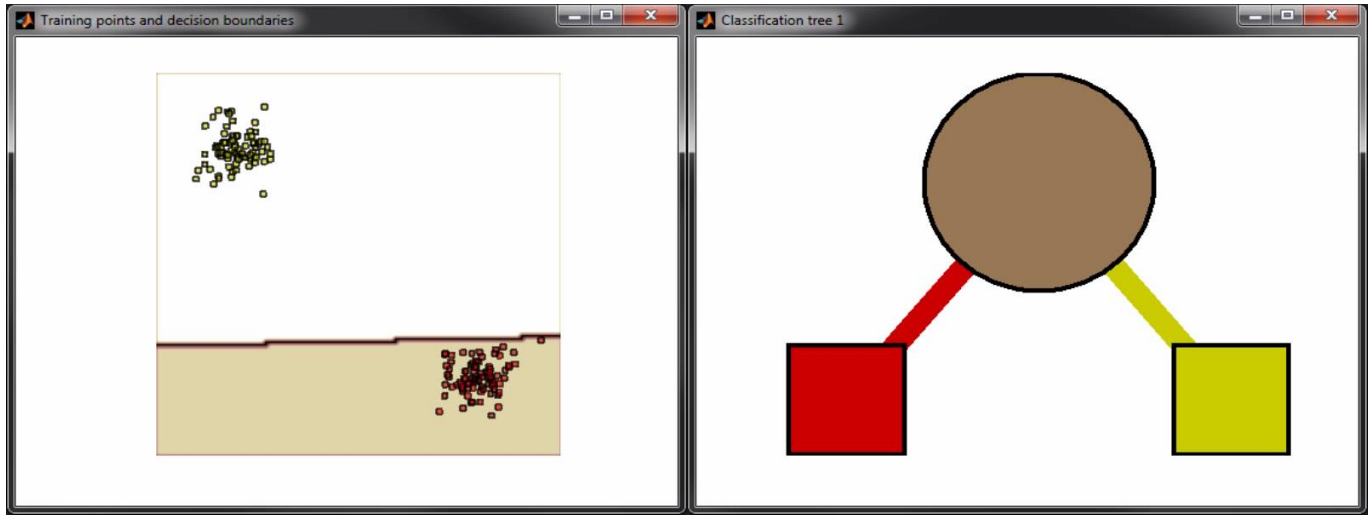


Example: Better Decision Boundary

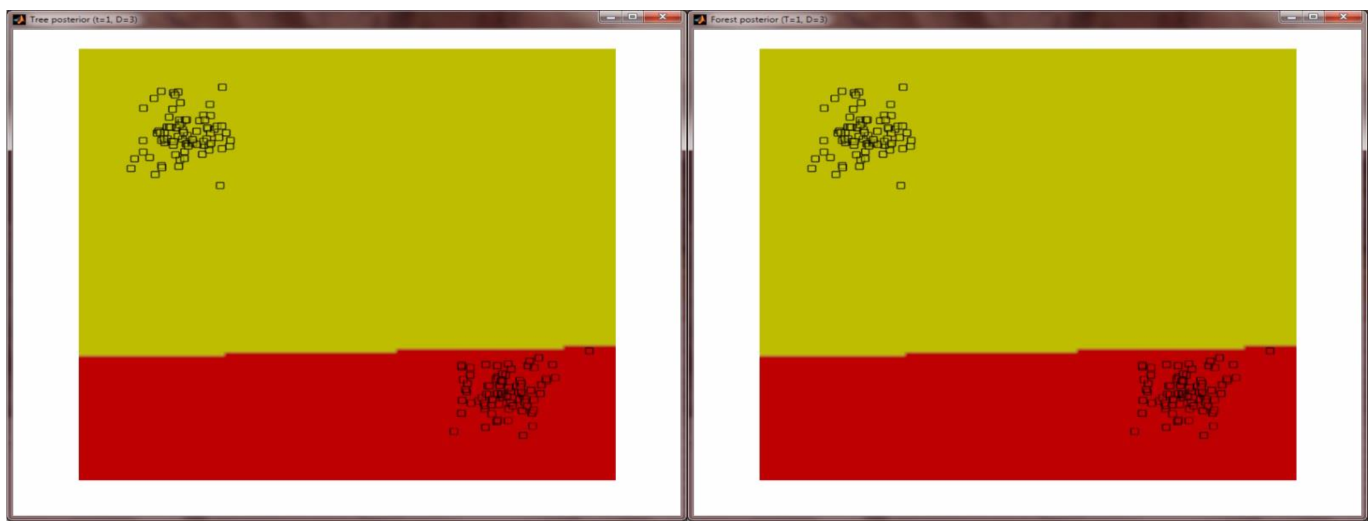


mushroom
example

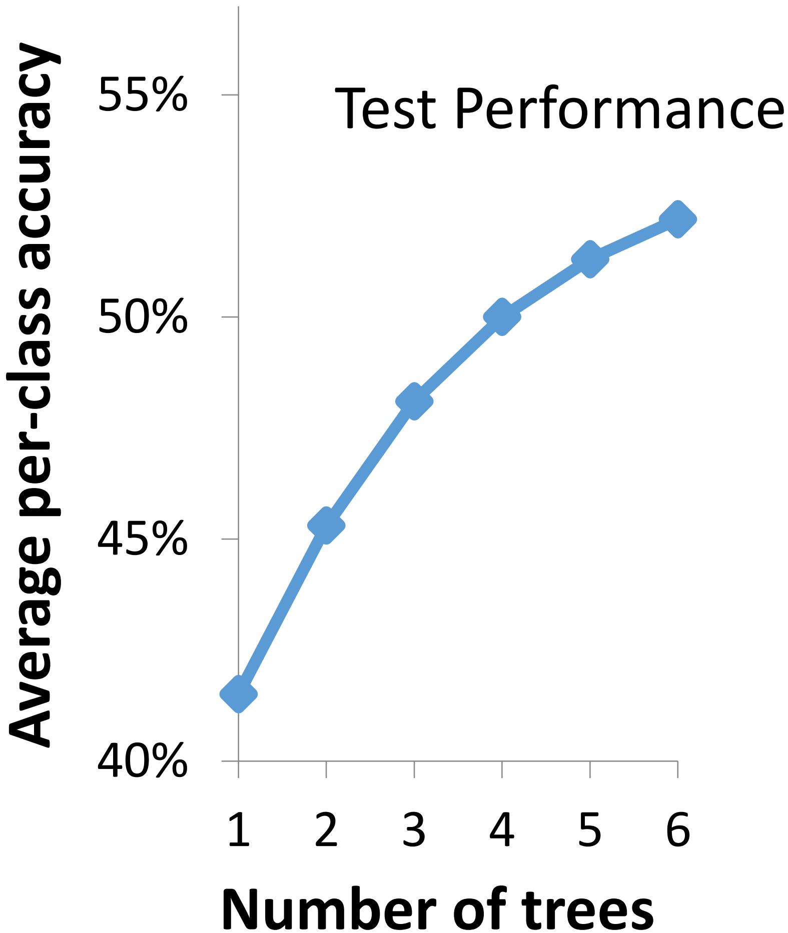
Training different trees in the forest

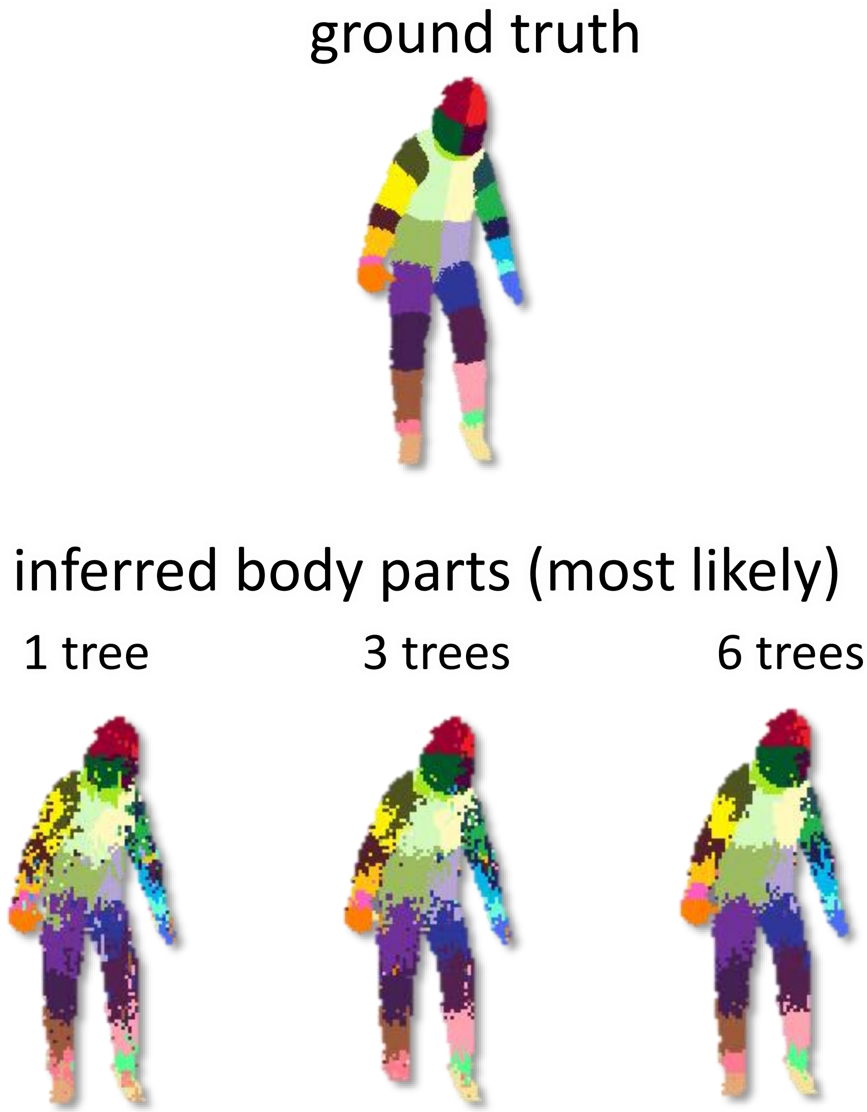


Testing different trees in the forest



Number of Trees





The following slides contain additional Information, which is not relevant for the exam

Reminder: affine cameras

- Affine camera has 8 DoF:

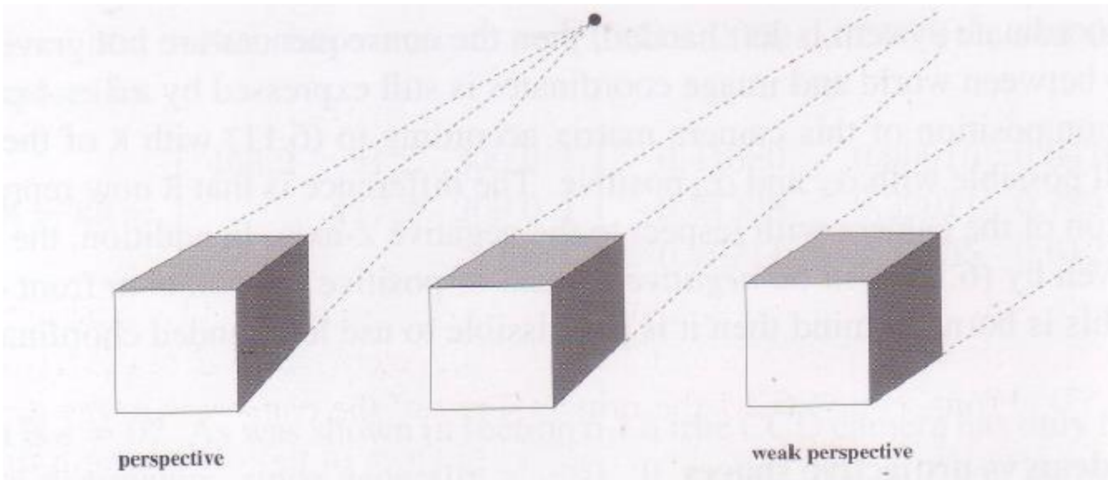
$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

- In short: $\tilde{x} = \underset{2\times 3}{M}\tilde{X} + \underset{2\times 1}{t}$

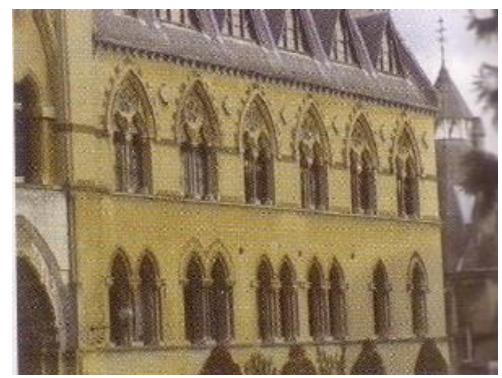
- Parallel 3D lines map to parallel 2D lines (since points stay at infinity)

$$\begin{bmatrix} x \\ y \\ 0 \end{bmatrix} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 0 \end{bmatrix}$$

Reminder: Affine cameras (from previous lecture)



(normal focal length)



(very large focal length)

“Close to parallel projection”

Affine Cameras give affine reconstruction

Assume we have reconstructed the scene with

$$P_j = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Then the transformations Q has to be an affine transformation in order to keep cameras affine:

$$x_{ij} = P_j X_i = P_j Q Q^{-1} X_i = P_j' X_i'$$

$$\mathbf{Q} = \begin{bmatrix} \mathrm{A} & \mathrm{t} \\ 0^{\mathsf{T}} & 1 \end{bmatrix} \quad \text{not:} \quad \mathbf{Q} = \begin{bmatrix} \mathrm{A} & \mathrm{t} \\ \mathrm{v}^{\mathsf{T}} & v \end{bmatrix}$$

Multi-View Reconstruction for affine cameras

(derivation on blackboard)

$$\tilde{x}_{ij} = M_j \tilde{X}_i + t_j$$

$$\min_{M_j, t_j, \tilde{X}_i} \sum_{ij} \| \tilde{x}_{ij} - M_j \tilde{X}_i - t_j \|^2$$

assume all point visible in all views

Get all t_j's in closed form:

consider 1D case and one camera M, t.

$$\frac{\partial}{\partial t} \sum_i (\underset{1}{x_i} - \underset{1\times 3}{M} \underset{3\times 1}{\tilde{X}_i} - \underset{1}{t})(x_i - M\tilde{X}_i - t) \stackrel{!}{=} 0$$

$$\Rightarrow \frac{\partial}{\partial t} \sum_i (\text{const} - 2x_i t + 2M\tilde{X}_i t + t^2) \stackrel{!}{=} 0$$

$$\Rightarrow \sum_i (-2x_i + 2M\tilde{X}_i + 2t) \stackrel{!}{=} 0$$

$$\Rightarrow -2\sum_i x_i + 2M\sum_i \tilde{X}_i + 2nt \stackrel{!}{=} 0$$

$$\Rightarrow t = \frac{1}{n}\sum_i x_i - \frac{1}{n} M \sum_i \tilde{X}_i$$

we choose centroid $\sum_i \tilde{X}_i = 0$

$$\Rightarrow t = \frac{1}{n}\sum x_i$$

Multi-View Reconstruction for affine cameras

(derivation on blackboard)

we get:

$$\min_{M_j, \tilde{X}_i} \sum_{ij} \|\tilde{x}_{ij} - M_j \tilde{X}_i\|^2$$

$$= \min_{M_j, \tilde{X}_i} \Big\| \underbrace{\begin{pmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & & \vdots \\ x_{m1} & \cdots & x_{mn} \end{pmatrix}}_{\substack{2m \times n \\ W}} - \underbrace{\begin{pmatrix} M_1 \\ \vdots \\ M_m \end{pmatrix}}_{\substack{2m \times 3 \\ M}} \underbrace{(\tilde{X}_1 \cdots \tilde{X}_n)}_{\substack{3 \times n \\ X}} \Big\|_F^2$$

Note, Frobenius norm:

$$\|A\|_F = \left(\sum_i \sum_j |a_{ij}|^2\right)^{\frac{1}{2}}$$

Optimum from SVD of W

$$\underset{2m \times n}{W} = (\underbrace{u_1\, u_2\, u_3}_{M} \cdots u_n) \begin{pmatrix} z_1 & & & & \\ & z_2 & & & \\ & & z_3 & & \\ & & & 0 & \\ & & & & \ddots \\ & & & & & 0 \end{pmatrix} \left.\begin{pmatrix} v_1^T \\ v_2^T \\ v_3^T \\ \vdots \\ v_n^T \end{pmatrix}\right\} X$$

$$M = (z_1 u_1, z_2 u_2, z_3 u_3) \quad X^T = (v_1, v_2, v_3)$$

also optimal under Frobenius norm if measurements are noisy.

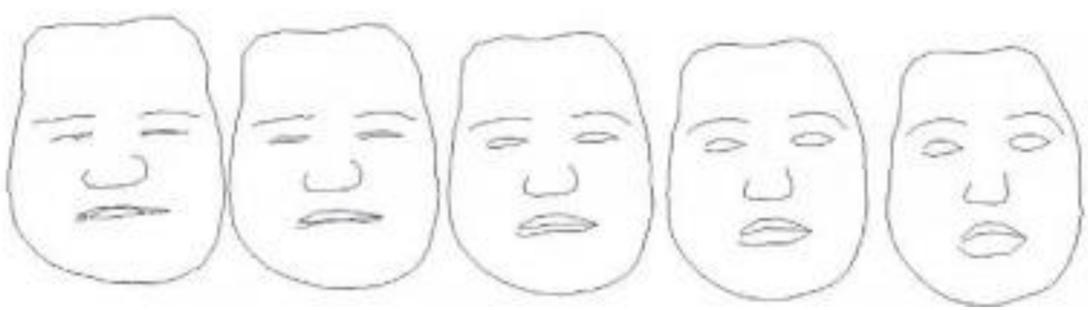
Comments / Extensions

- Main restriction is that all points have to be visible in all views.
  (can be used for a subset of views and then "zipping" sub-views together)

- Extensions to missing data have been done
  (see HZ ch. 18)

- Extensions to projective cameras have been done (see HZ ch. 18.4)

- Extensions to non-rigidly moving scenes (see HZ ch. 18.3)

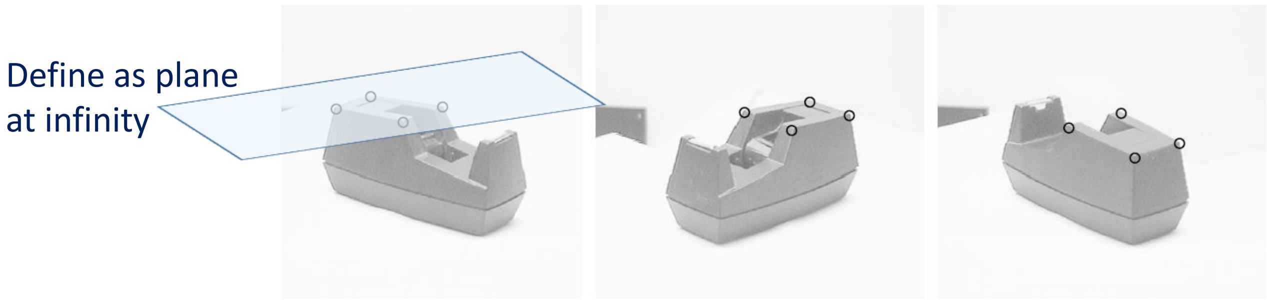


Direct reference plane approach (DRP)

- $H_\infty = KR$ is called infinity homography since it is the mapping from the plane at infinity to the image:

$$x = H_\infty(I| - \tilde{C})\begin{pmatrix} x \\ y \\ z \\ 0 \end{pmatrix} = H_\infty \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

- Basic Idea: simply define any plane as the plane at infinity $\pi_\infty = \ (0{,}0{,}0{,}1)^T$ (this can be done in projective space)



[Rother PhD Thesis 2003]

Direct reference plane approach (DRP)

Derivation on blackboard

1) Compute H_∞

4 image points $x_{1-4} \xleftarrow{H_\infty} \begin{pmatrix}1\\0\\0\end{pmatrix}, \begin{pmatrix}0\\1\\0\end{pmatrix}, \begin{pmatrix}0\\0\\1\end{pmatrix}, \begin{pmatrix}1\\1\\1\end{pmatrix}$

8 eqn. and 8 unknowns

2) It is for arbitrary point $\tilde{X}$ and Camera:

$$\lambda x = H_\infty (I \mid -\tilde{C}) \begin{pmatrix}\tilde{X}\\1\end{pmatrix}$$

$$\Rightarrow \underbrace{\lambda H_\infty^{-1} x}_{x'} = \tilde{X} - \tilde{C}$$

Take ratios:

① $\frac{\lambda x'_1}{\lambda x'_2} = \frac{\tilde{X}_1 - \tilde{C}_1}{\tilde{X}_2 - \tilde{C}_2} \Rightarrow x'_1 \tilde{X}_2 - x'_1 \tilde{C}_2 - x'_2 \tilde{X}_1 + x'_2 \tilde{C}_1 = 0$

② $\frac{\lambda x'_1}{\lambda x'_3} = \frac{\tilde{X}_1 - \tilde{C}_1}{\tilde{X}_3 - \tilde{C}_3} \Rightarrow x'_1 \tilde{X}_3 - x'_1 \tilde{C}_3 - x'_3 \tilde{X}_1 + x'_3 \tilde{C}_1 = 0$

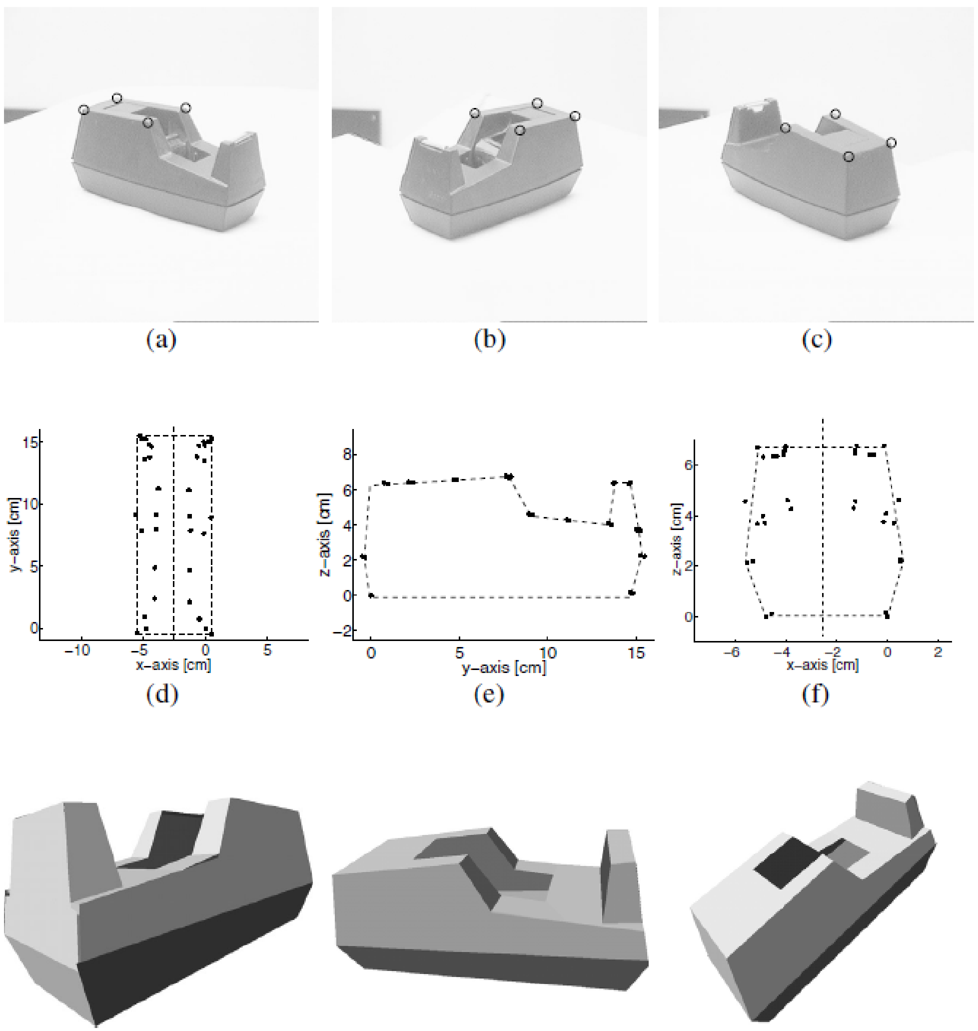
3) For every 3D point X^i visible in Camera P^j we get a constraint. Big linear system:

$$\left(\text{image measure } x_{ij} \right) \begin{pmatrix} \tilde{X}^1_1 \\ \tilde{X}^1_2 \\ \vdots \\ \tilde{X}^n_3 \\ \tilde{C}^1_1 \\ \tilde{C}^1_2 \\ \vdots \\ \tilde{C}^m_3 \end{pmatrix} = 0$$

4) Solved with SVD gives 4D Null space

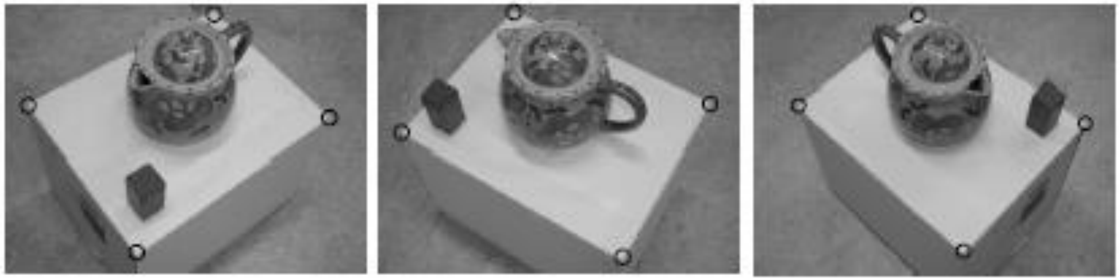
[Rother PhD Thesis 2003]

Results



How to get infinite Homographies

- Real Plane in the scene:

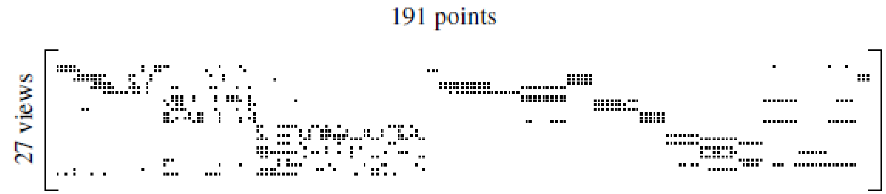


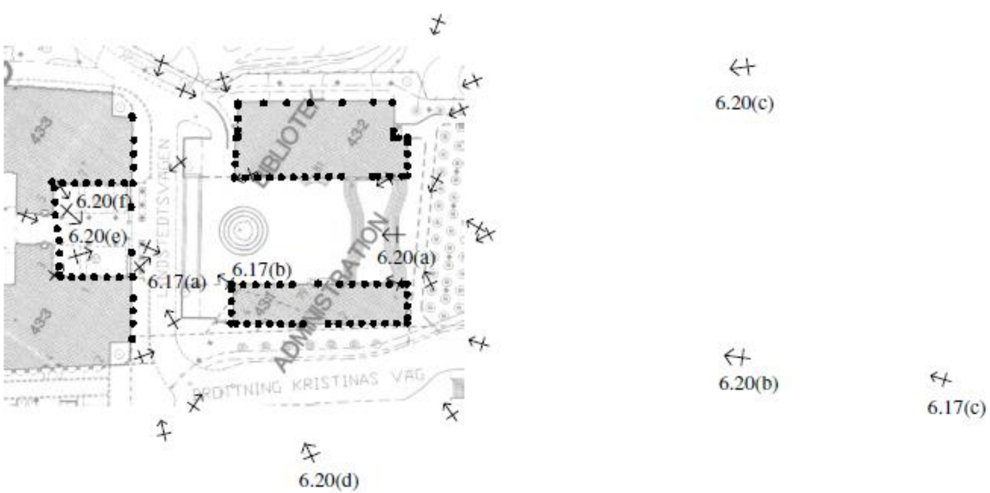
- Fixed / known K and R, e.g. translating camera with fixed camera intrinsic

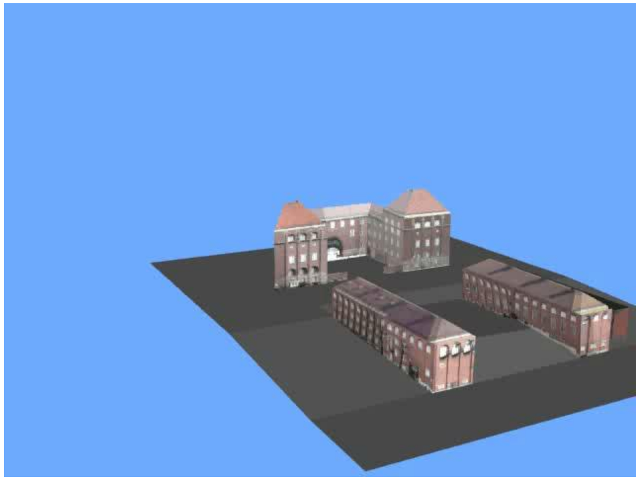
- Orthogonal scene directions and a square pixel camera.
  We can get out: K, R (up to a small, discrete ambiguity)

Results: University Stockholm

(a) (b) (c)







Constant intrinsic parameters (sketch only)

- Assume K is constant over 3+ Frames then K can be computed

- We know that we can get $K, R, \tilde{C}$ from $P = K\ R\ (I_{3\times3} \mid -\tilde{C})$

- We have P_1, P_2, P_3 and it is

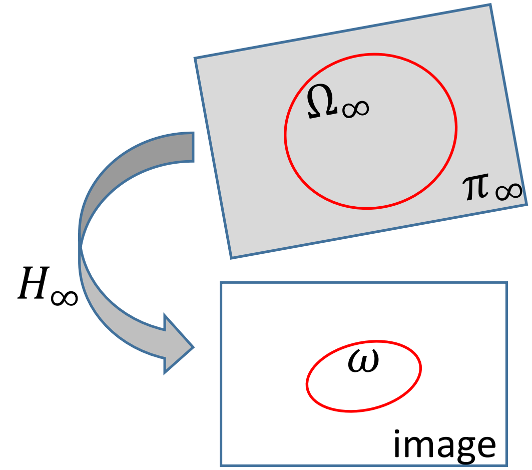
$$x_{i1} = P_1 X_i = P_1 Q^{-1} Q X_i = P_1' X_i'$$
$$x_{i2} = P_2 X_i = P_2 Q^{-1} Q X_i = P_2' X_i'$$
$$x_{i3} = P_3 X_i = P_3 Q^{-1} Q X_i = P_3' X_i'$$

- Try to find a Q such that all P_1, P_2, P_3 have the same K but different R_{1-3} and $\tilde{C}_{1-3}$

- See details in chapter 19 HZ

- (Note: this does not work if camera zooms during capture)

Side comment: Where does ω come from?



- There a “strange thing” call the absolute conic $\Omega_\infty = I_{3\times3}$ that lives on the plane at infinity $\pi_\infty = (0,0,0,1)^T$
- The absolute conic is an “imaginary circle with radius i”:

  $(x,y,1)\Omega_\infty(x,y,1)^T = 0$

  hence: $x^2 + y^2 = -1$
- ω is called the “image of the absolute conic”, since it is the mapping of the absolute conic onto the image plane
- Proof:
  1. The homography $H_\infty = KR$ is the mapping from the pane at infinity to the image plane. Since

     $$x = KR\,[I \mid -C]\,(x,y,z,0)^T$$

     hence $x = KR\,(x,y,z)^T$
  2. The conic $\Omega_\infty = \mathrm{I}_{3\times3}$ maps from the plane at infinity to π_∞ to the image as:

$$H_\infty^{-\mathrm{T}}\Omega_\infty H_\infty^{-1} = (KR)^{-T}I(KR)^{-1} = K^{-T}R^{-T}R^{-1}K^{-1} = K^{-T}K^{-1} = \omega$$