



Perception IV: Place Recognition & Line Extraction

Autonomous Mobile Robots

Margarita Chli

Roland Siegwart, Juan Nieto and Nick Lawrence

Today's Lecture

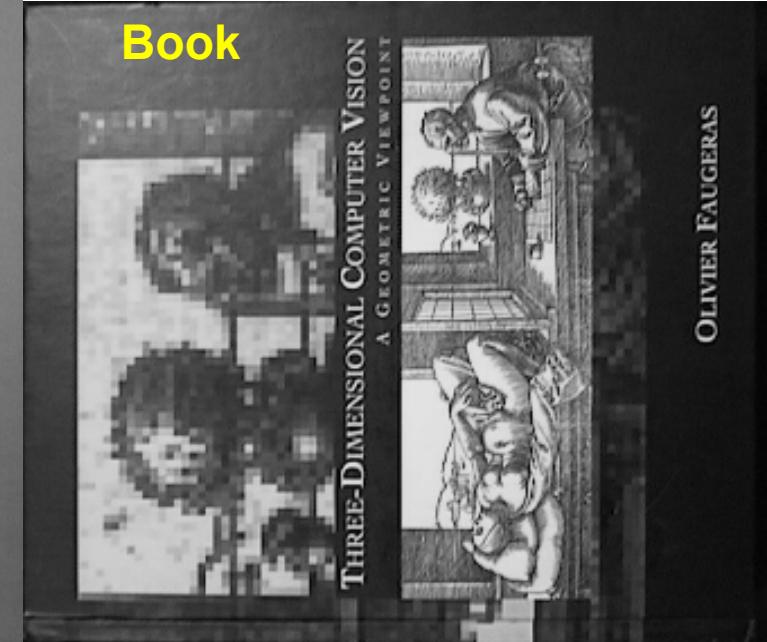
- **Place Recognition – Section 4.6**
 - Bag of Words
 - The Vocabulary Tree
 - FABMAP & other place recognition methods from the state of the art
- **Uncertainties:** (mainly) Section 4.7
 - Representation + Propagation – Section 4.1.3
 - **Line extraction** from a point cloud
 - Split-and-merge
 - Line-Regression
 - RANSAC
 - Hough Transform

Optional Reading:

- “[Video Google](#)”, J. Sivic and A. Zisserman, ICCV 2003
- “[Scalable Recognition with a Vocabulary Tree](#)”, D. Nistér and H. Stewénius, CVPR 2006.
- “[FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance](#)”, M. Cummins and P. Newman, IJRR 2008.
- “[SeqSLAM: Visual route-based navigation for sunny summer days and stormy winter nights](#)”, M. Milford and G. Wyeth, ICRA 2012
- “[Bags of Binary Words fro Fast Place Recognition in Image Sequences](#)”, D. Gálvez-López and Tardos, TRO 2012

Object recognition

Q: Is this Book present in the Scene?



Object recognition | with SIFT features

Q: Is this Book present in the Scene?



**Extract keypoints
in both images**

Object recognition | with SIFT features

Q: Is this Book present in the Scene?



Look for corresponding
matches

Most of the Book's keypoints are present in the Scene

⇒ A: The Book is present in the Scene

Object recognition | taking this a step further...

- Find an object in an image
- Find an object in multiple images
- Find multiple objects in multiple images



?



?

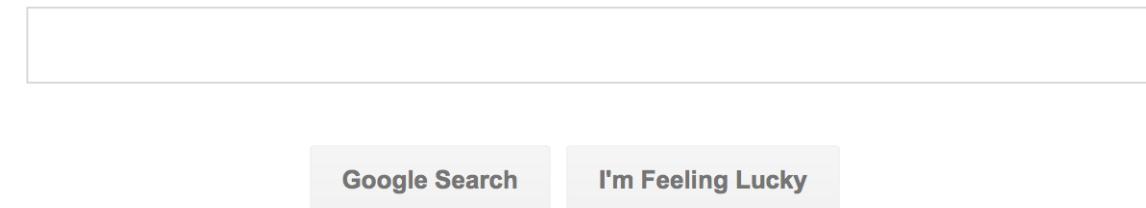


?



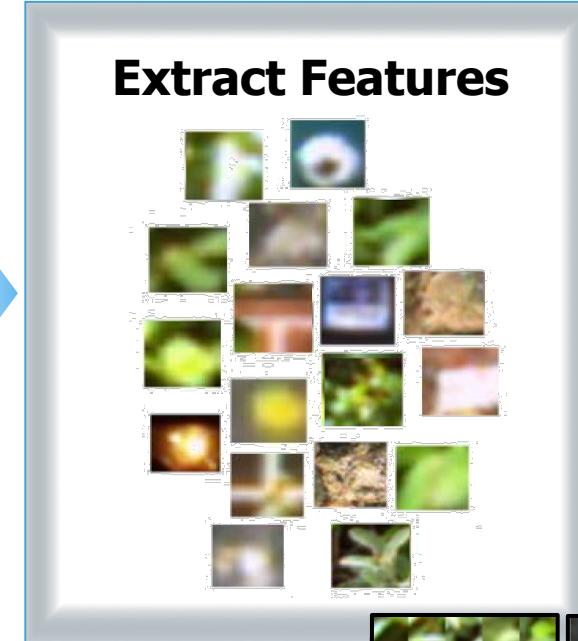
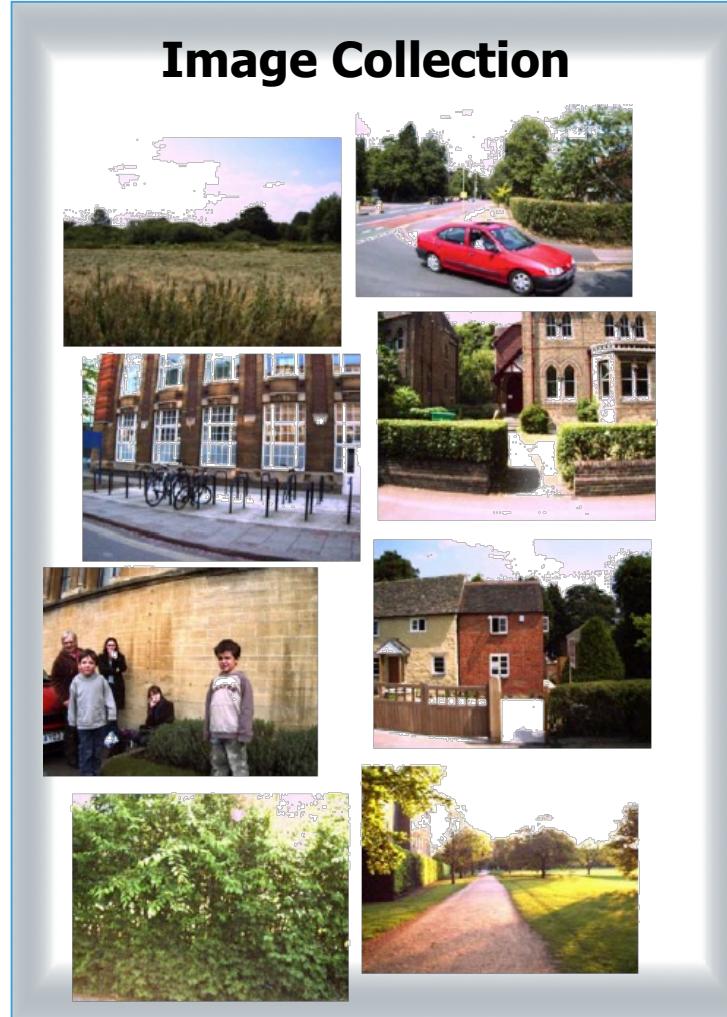
Image Retrieval | the Bag of Words (BoW) approach

- Extension to scene/place recognition:
 - Is this image in my database?
 - Robot: Have I been to this place before?
⇒ ‘loop closure’ problem, ‘kidnapped robot’ problem
- Use analogies from text retrieval:
 - Visual Words
 - Vocabulary of Visual Words
 - “Bag of Words” (BoW) approach

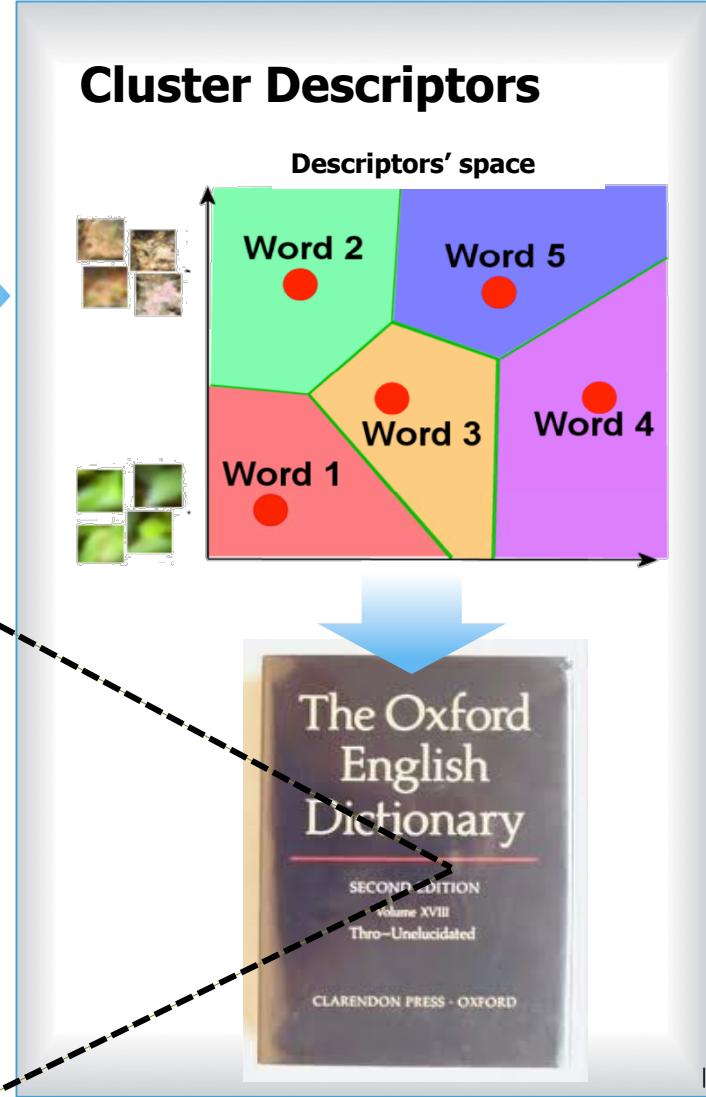
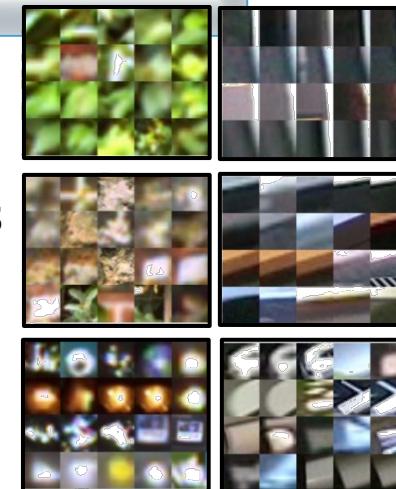


BoW | building the visual vocabulary

Images for this slide are courtesy of Mark Cummins



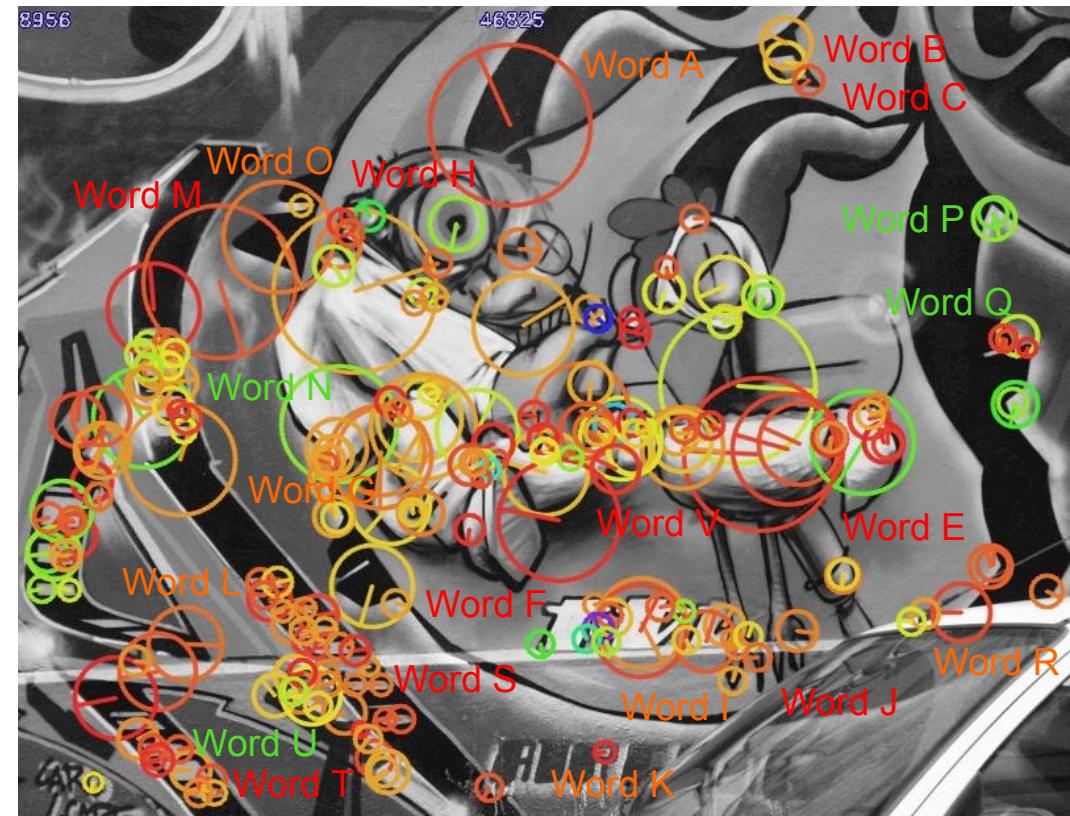
Examples
of
Visual
Words:



BoW | efficient image retrieval

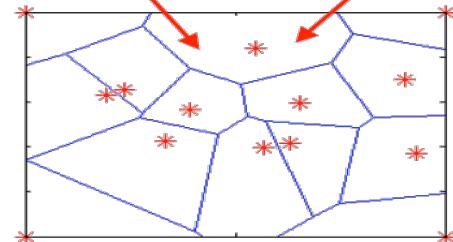
- **Image retrieval** represents a more general problem of object or place recognition.
- We can describe a **scene as a collection of words** and look up in the database for **images with a similar collection of words**

Graffiti Image = {Word A, Word A, Word B, ..., Word U}



BoW | Video Google: one image a thousand words?

Image courtesy of Andrew Zisserman



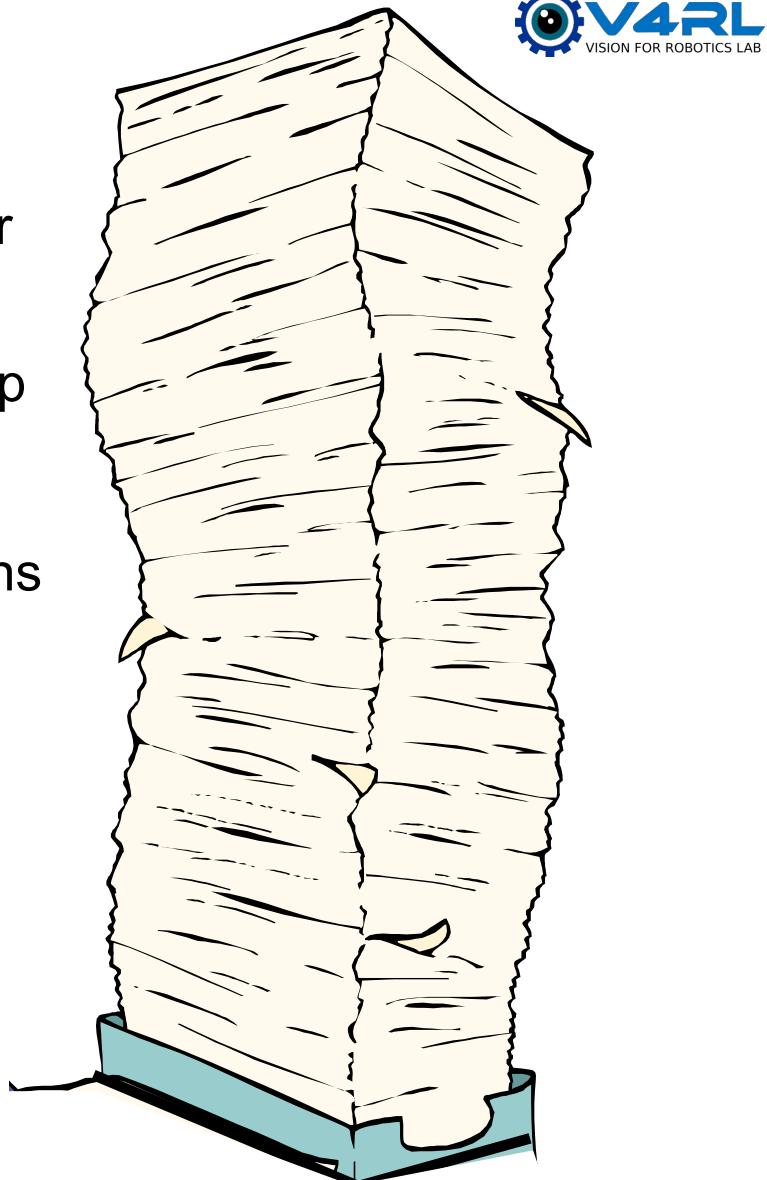
These features map to the same visual word

- “Video Google” [J.Sivic and A. Zisserman, ICCV 2003]
- Demo:



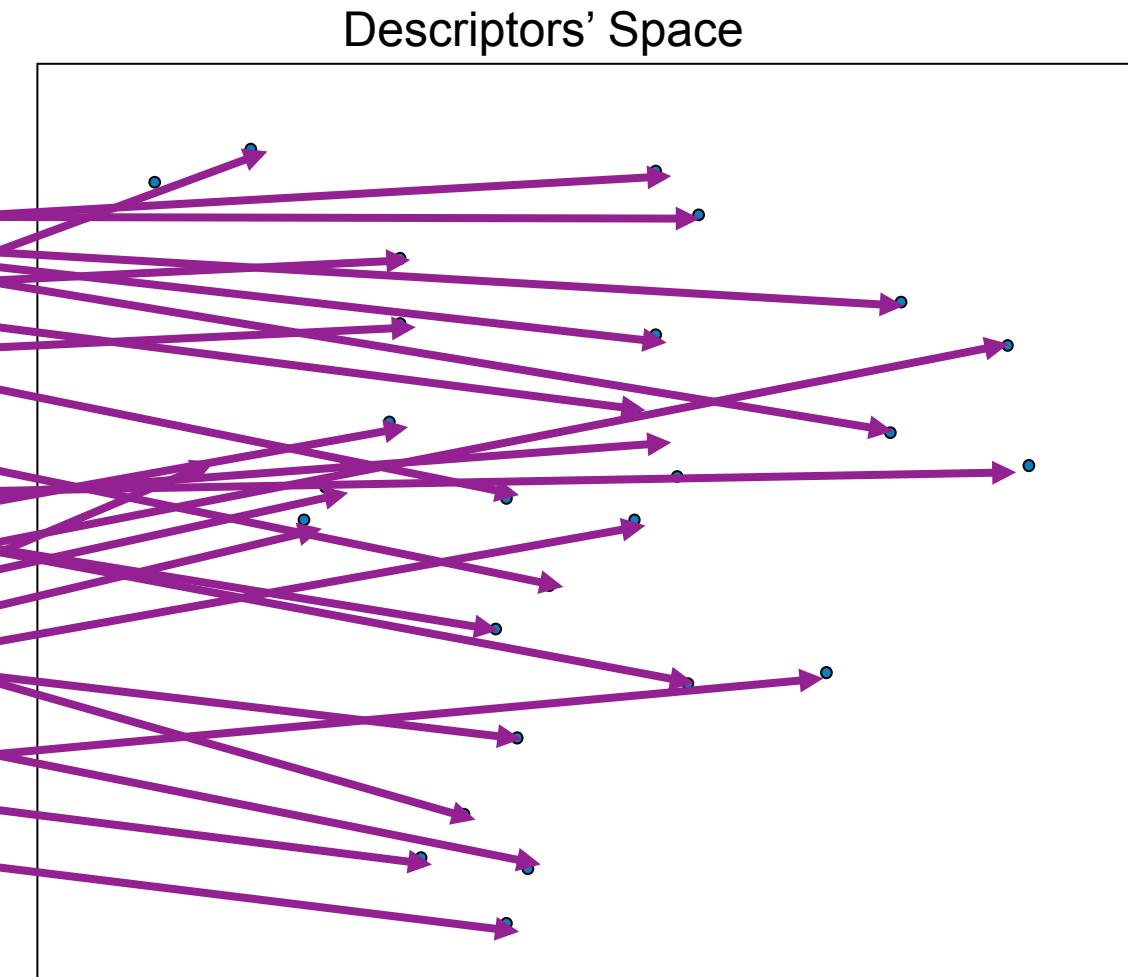
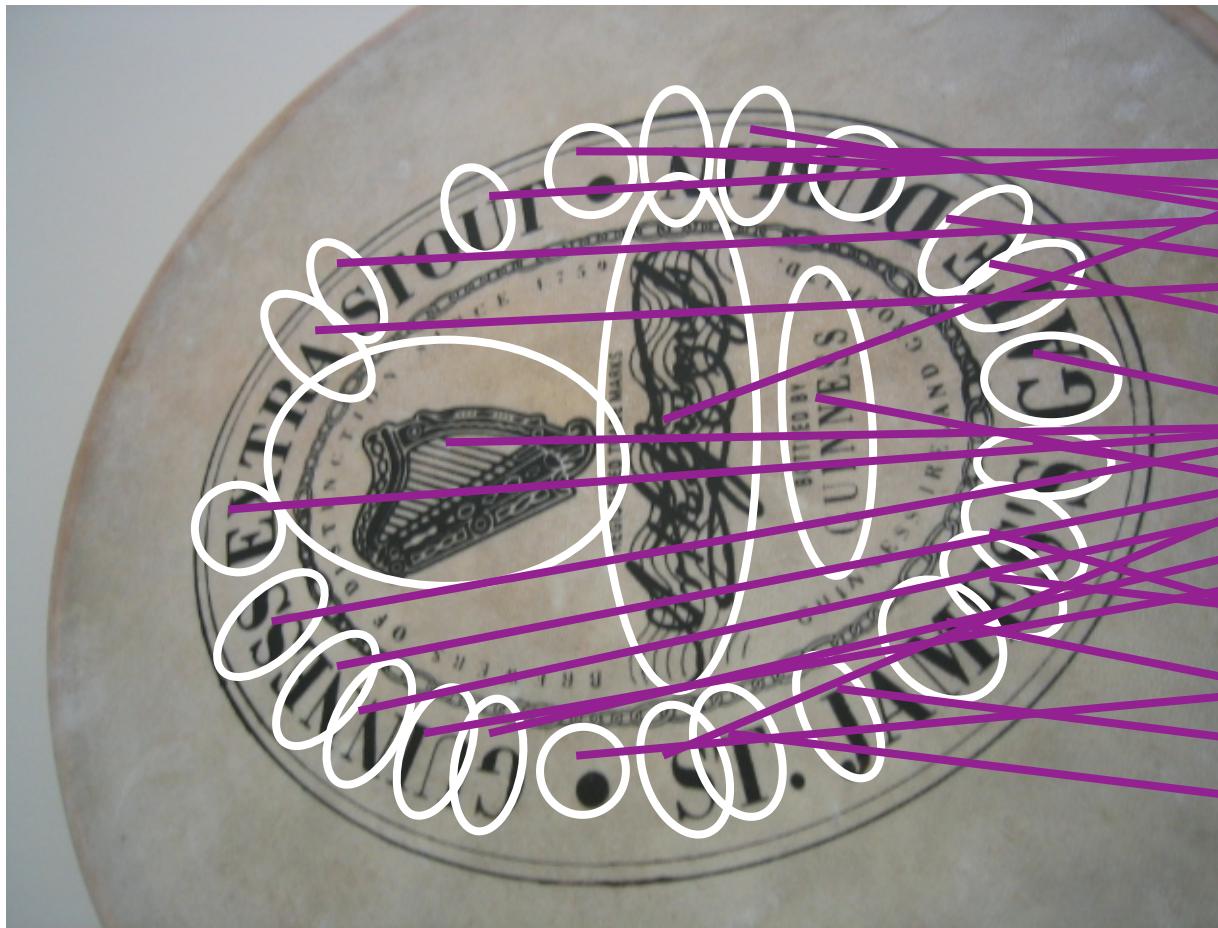
BoW | efficient image retrieval

- **Image retrieval** represents a more general problem of object or place recognition.
- We can describe a **scene as a collection of words** and look up in the database for **images with a similar collection of words**
- What if we need to find an object/scene in a database of millions of images?
 - Build **Vocabulary Tree** via hierarchical clustering
 - Use the **Inverted File system**:
a way of efficient indexing
(each node in the tree is associated with a list of images containing an instance of this node)



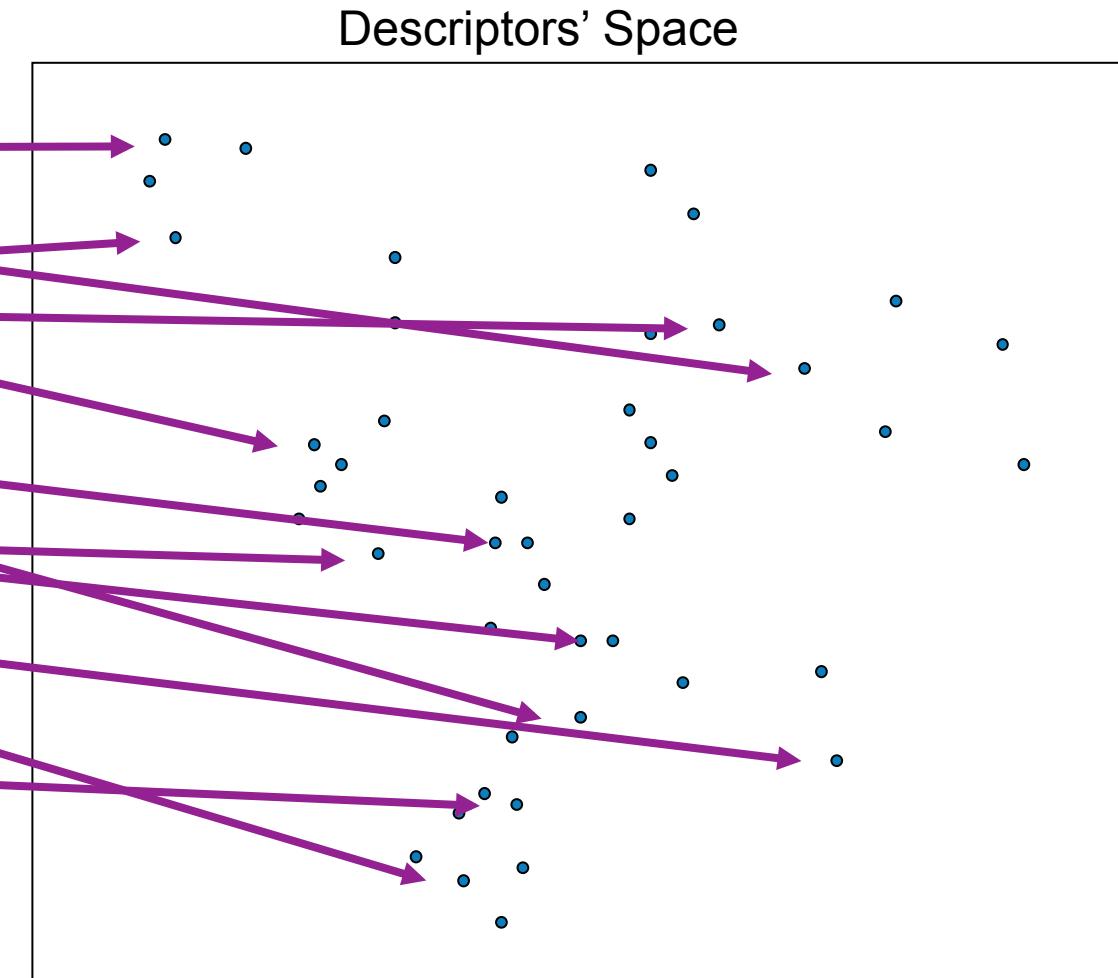
[Nistér and Stewénius, CVPR 2006]

Vocabulary tree | extract features



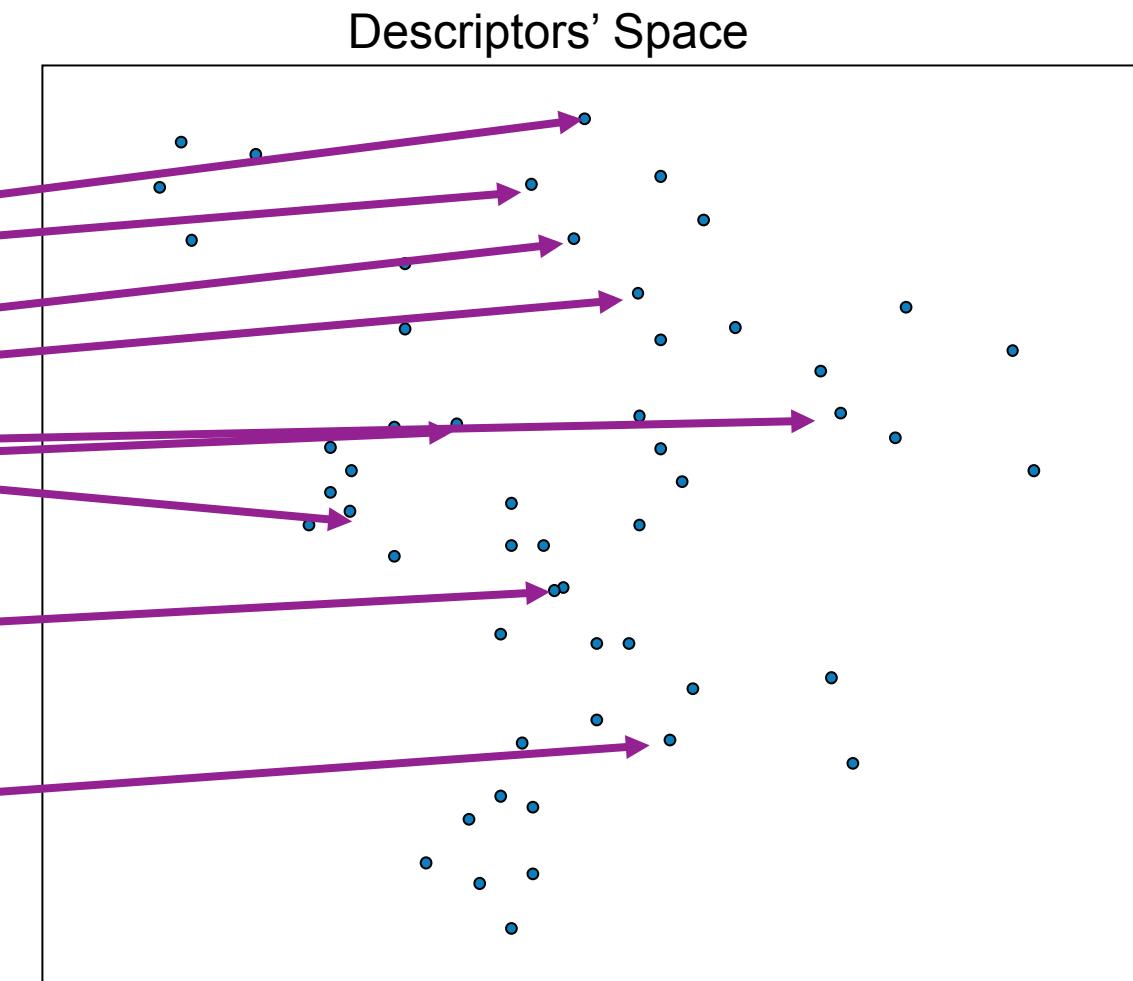
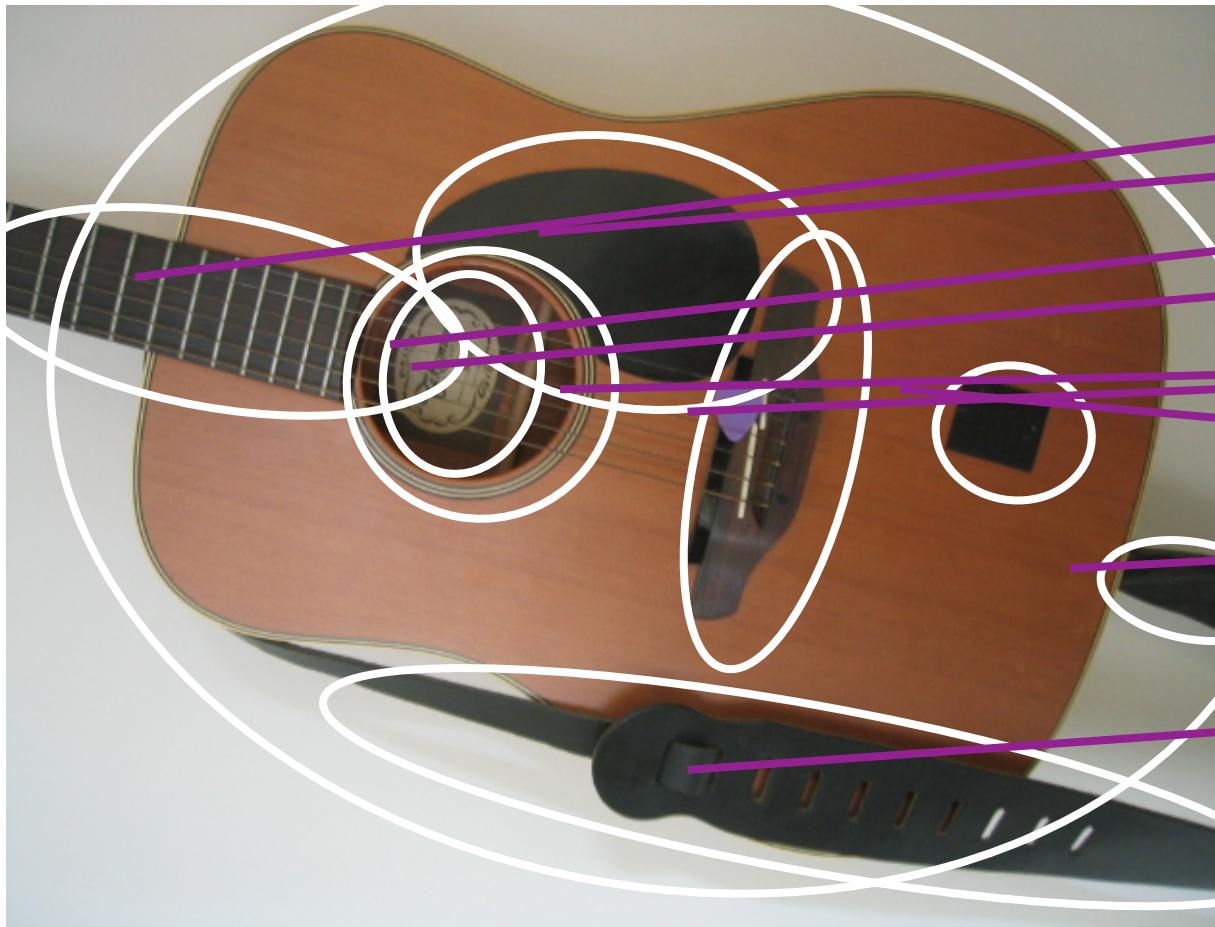
Based on D. Nister's slides

Vocabulary tree | extract features



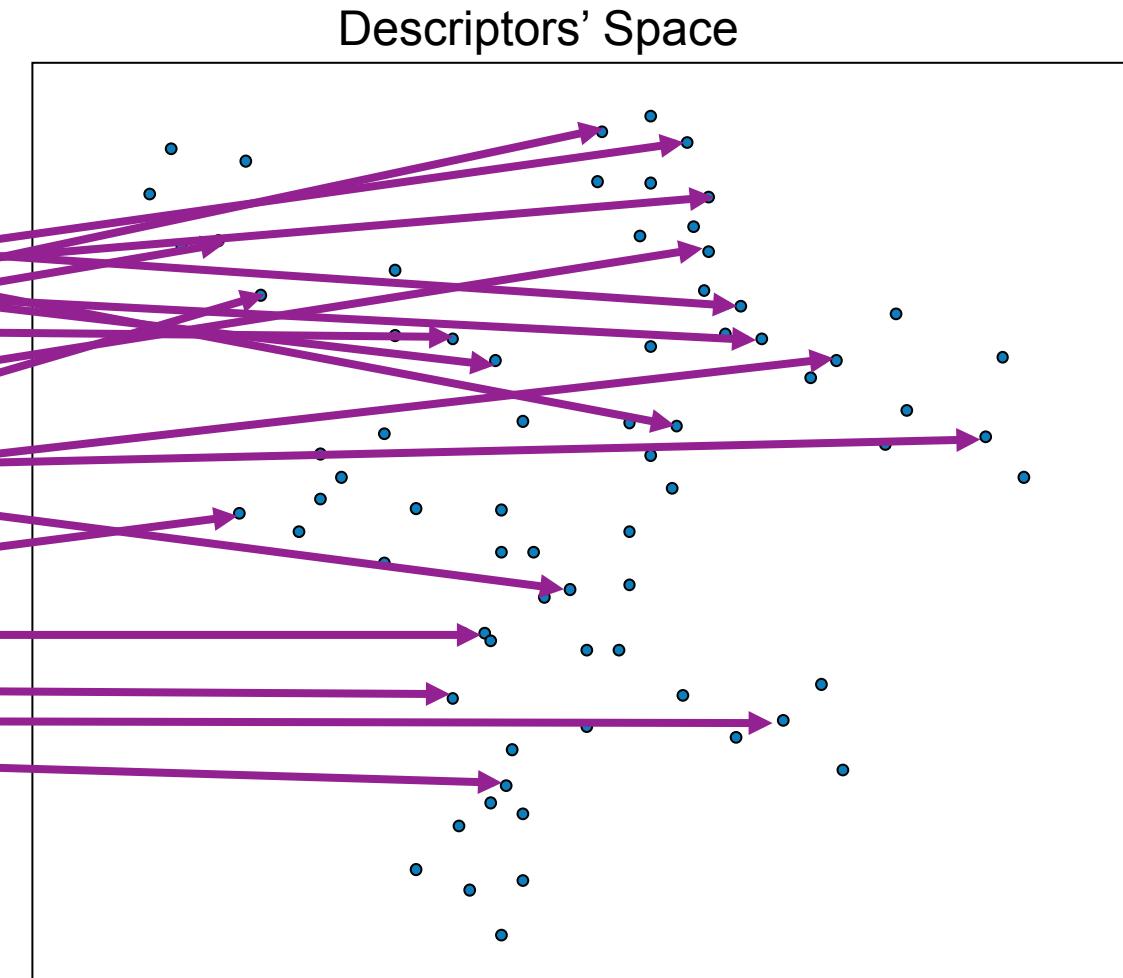
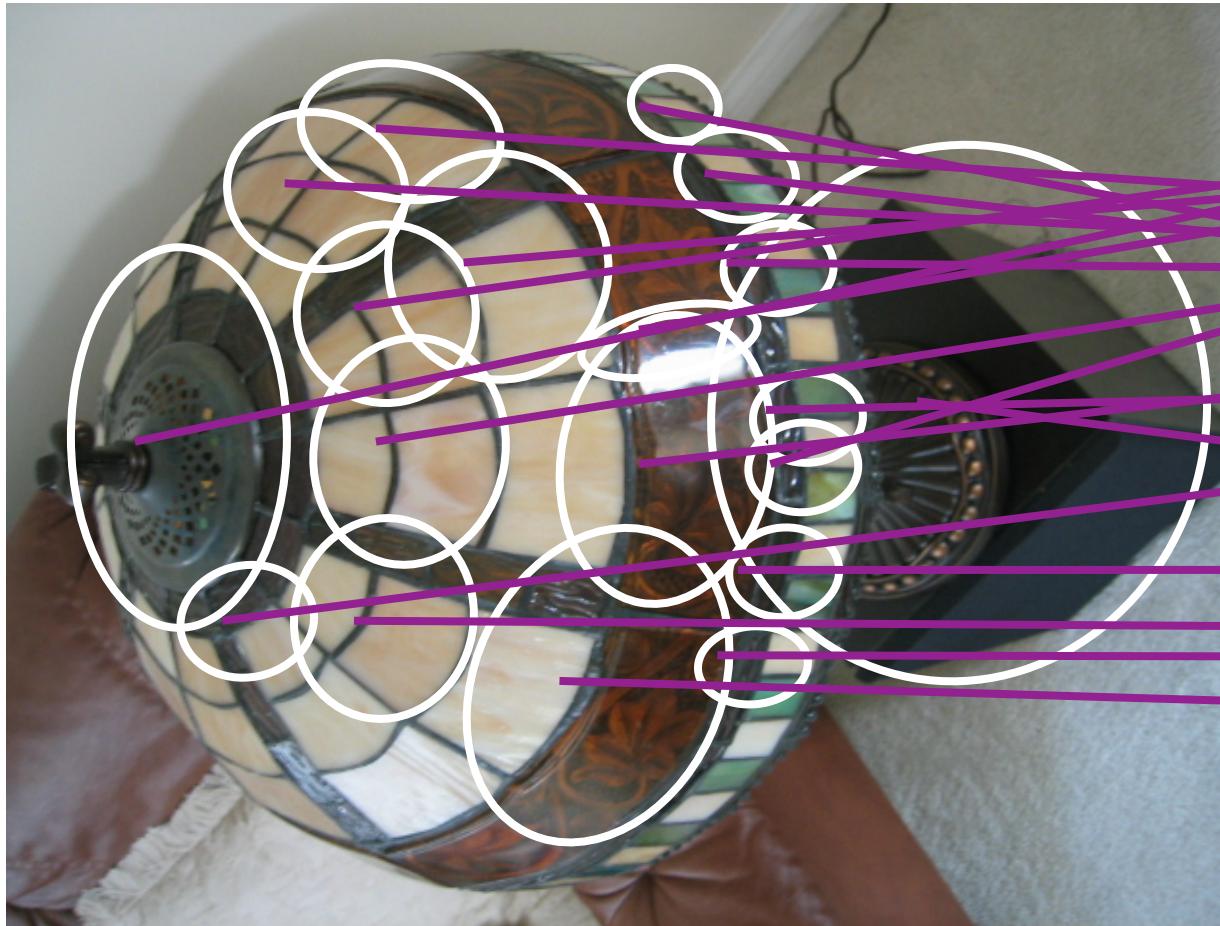
Based on D. Nister's slides

Vocabulary tree | extract features



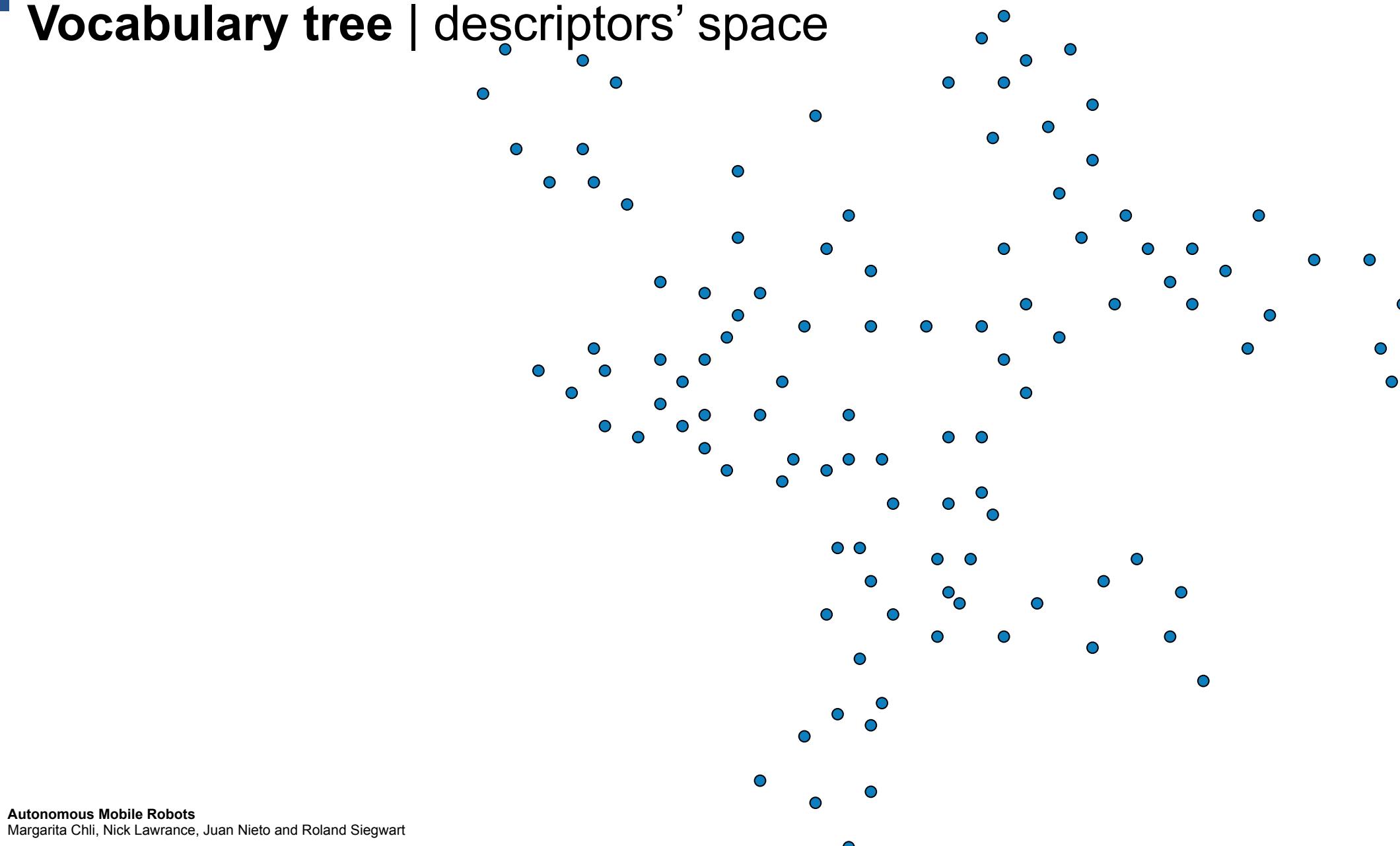
Based on D. Nister's slides

Vocabulary tree | extract features



Based on D. Nister's slides

Vocabulary tree | descriptors' space



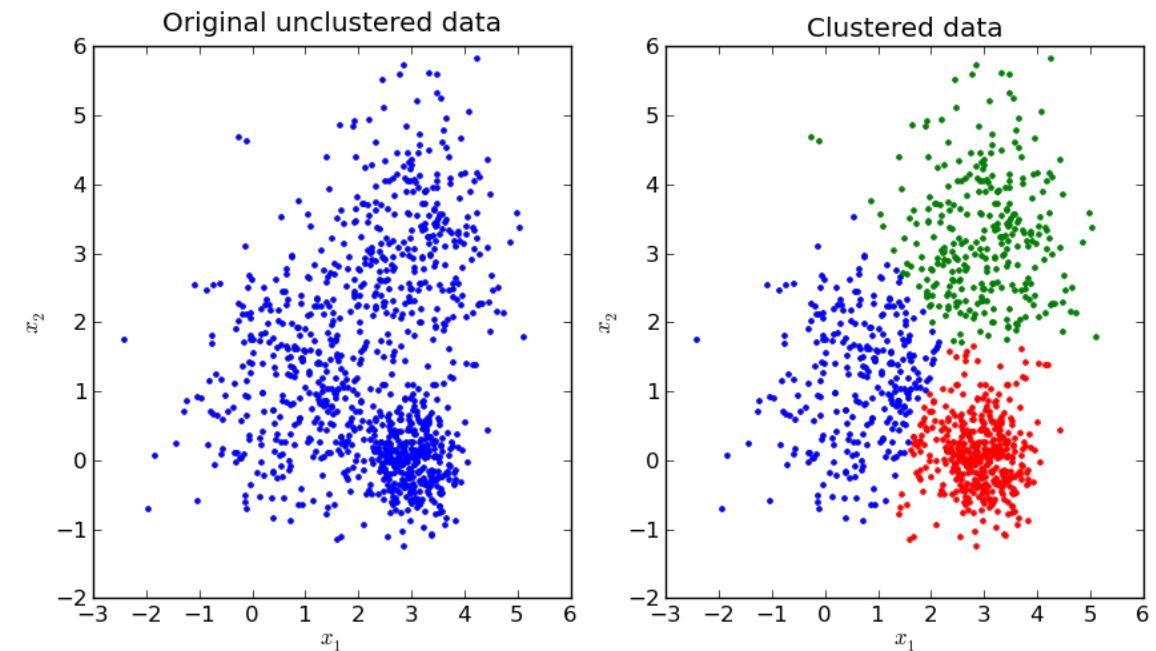
Based on D. Nister's slides

Vocabulary tree | k-means clustering, review

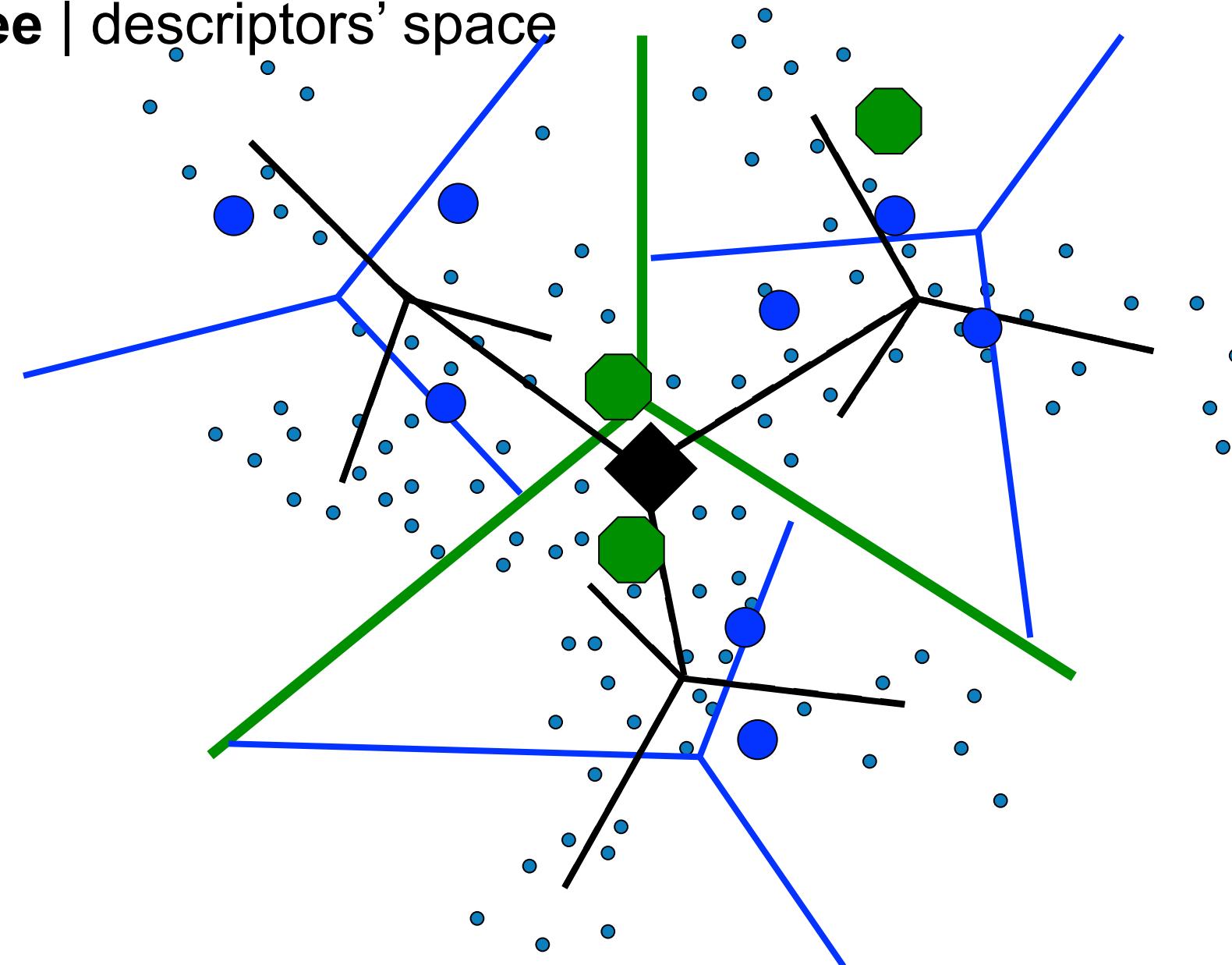
- **k-means clustering:** partitions a point cloud into k clusters, such that each point belongs to one cluster
- Minimizes the Sum of Squared Euclidean Distances between points and their nearest cluster-centers

Algorithm:

- Randomly initialize k cluster centers
- until (convergence) do:
 - assign each data-point to its nearest cluster-center
 - Re-compute each cluster-center as the mean of all points assigned to each cluster

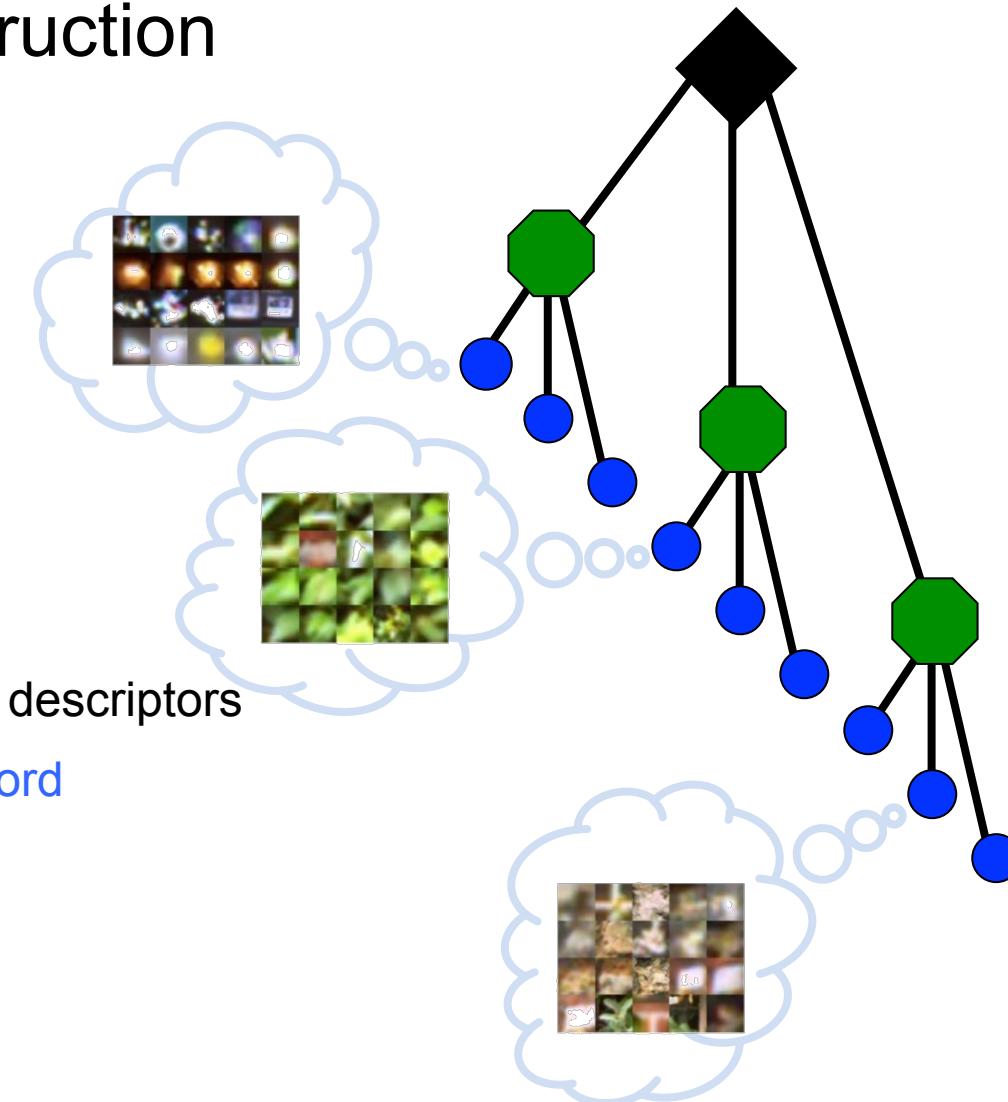


Vocabulary tree | descriptors' space



Based on D. Nister's slides

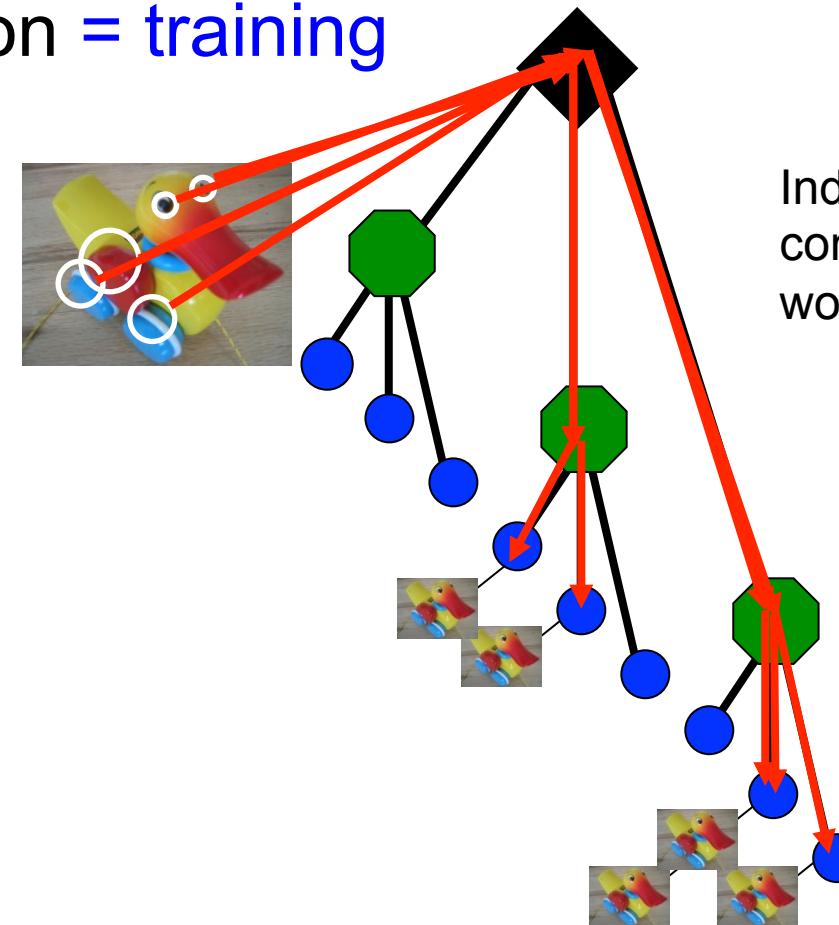
Vocabulary tree | construction



Each node represents a cluster of descriptors
⇒ each leaf represents a [visual word](#)

Vocabulary tree | construction = training

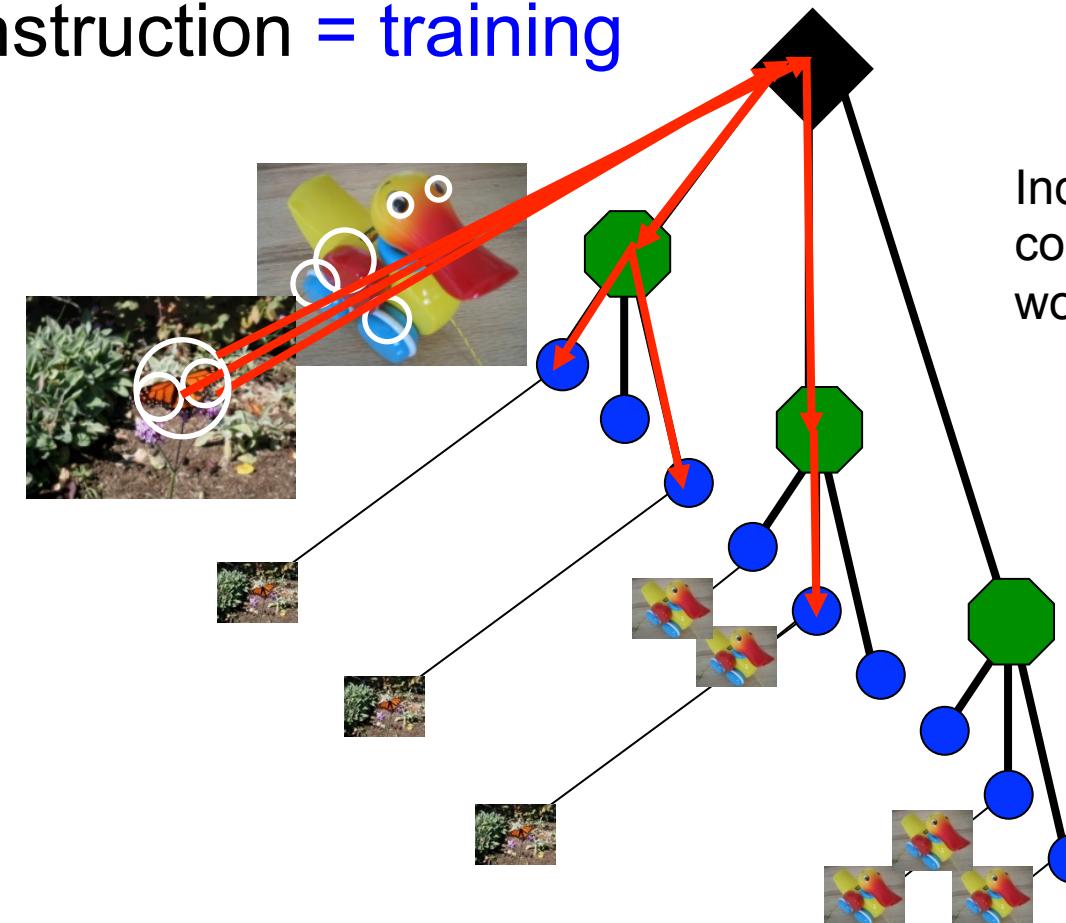
Model images



Index tree-leaves with images containing corresponding visual word

Vocabulary tree | construction = training

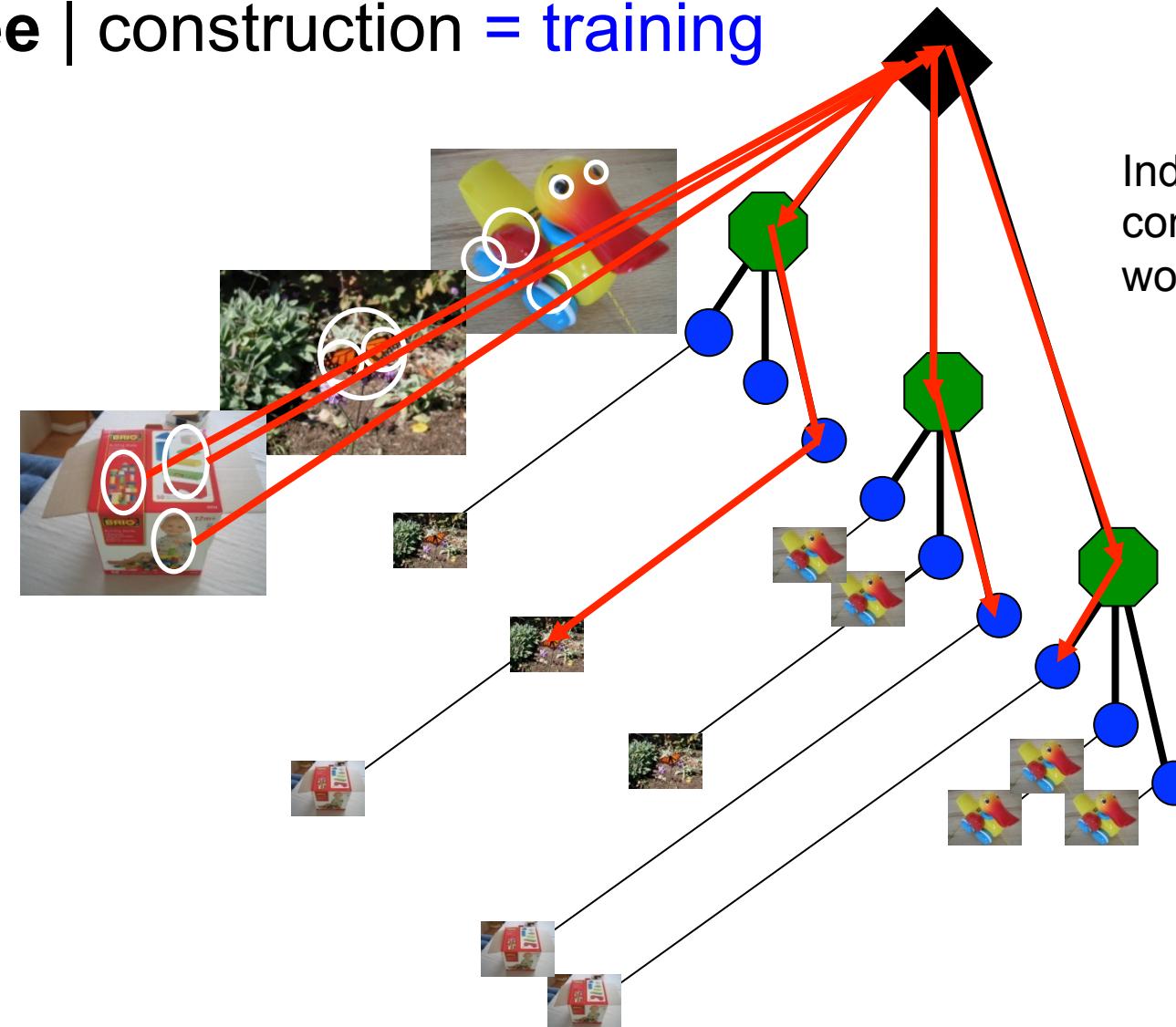
Model images



Index tree-leaves with images containing corresponding visual word

Vocabulary tree | construction = training

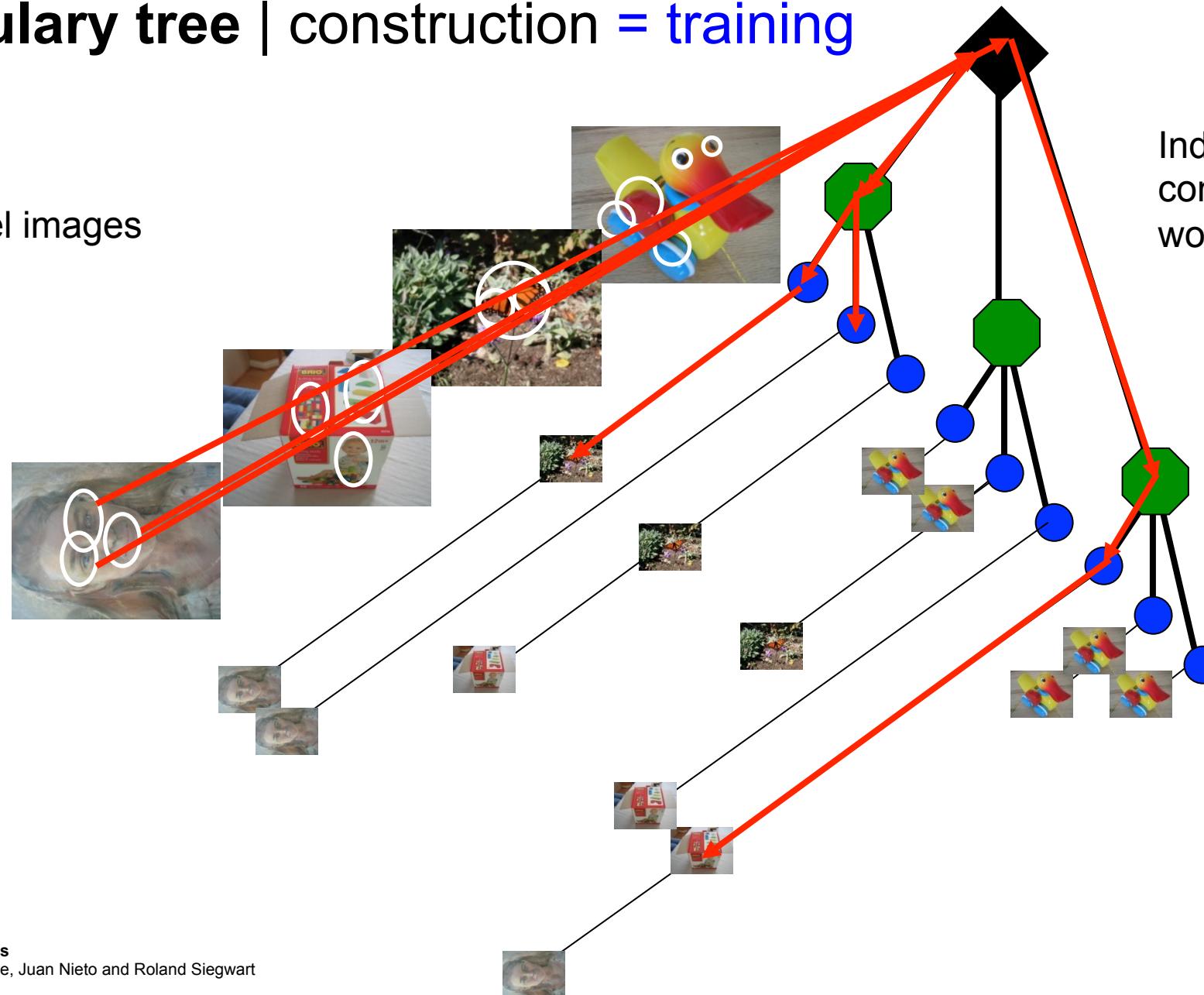
Model images



Index tree-leaves with images containing corresponding visual word

Vocabulary tree | construction = training

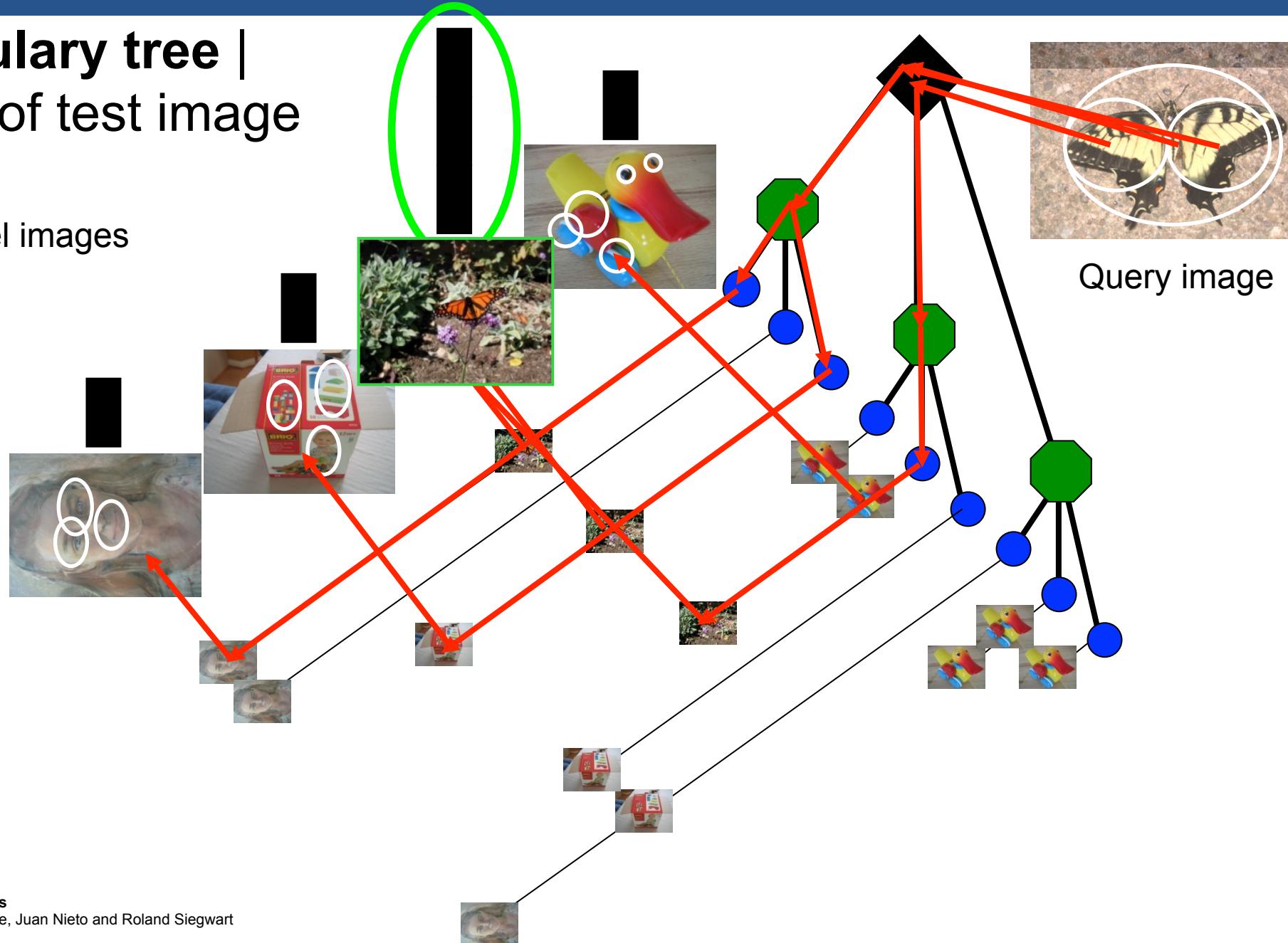
Model images



Index tree-leaves with images containing corresponding visual word

Vocabulary tree | lookup of test image

Model images

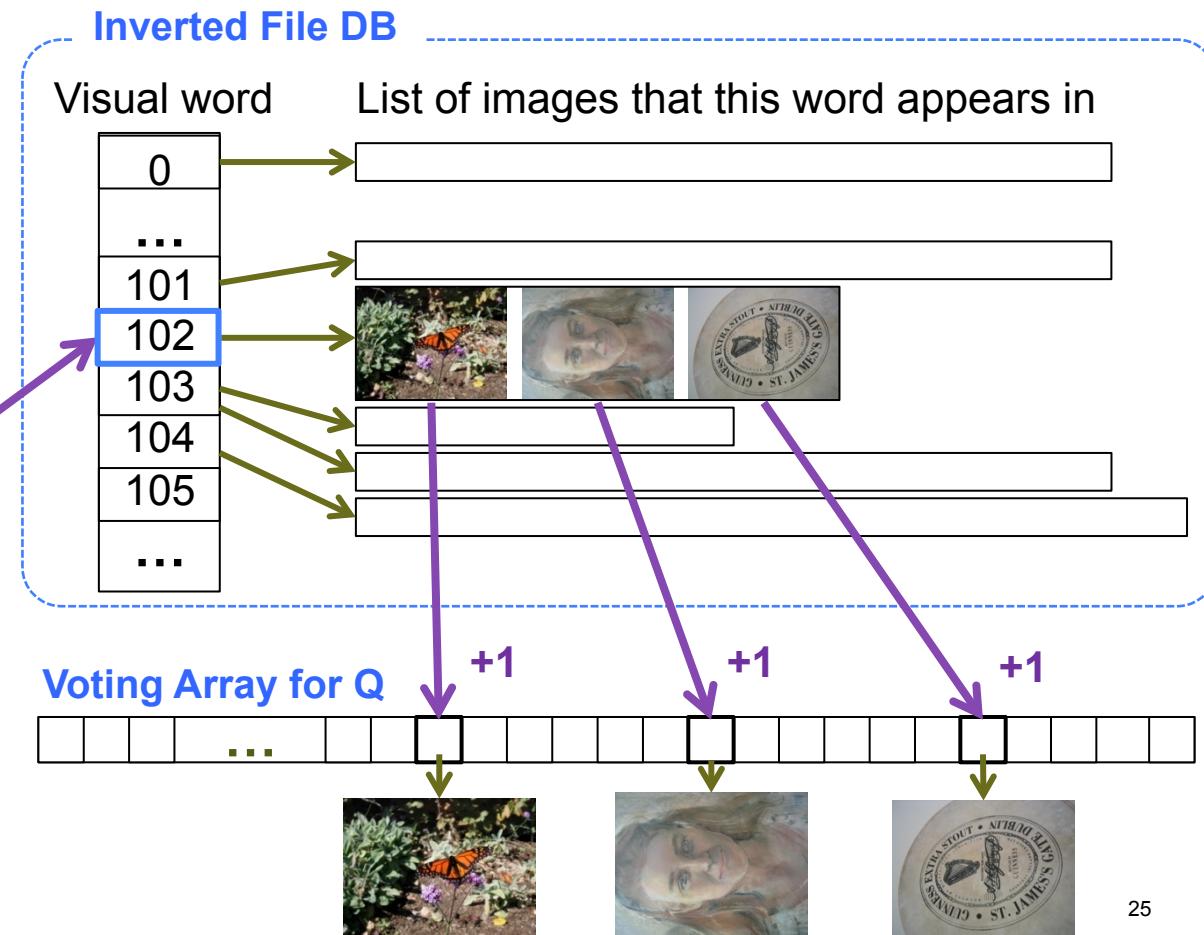
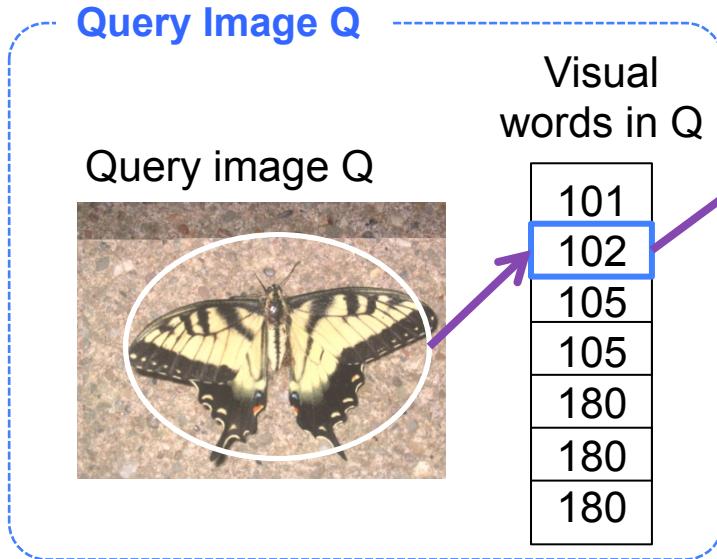


Based on D. Nister's slides

Vocabulary tree | inverted file index



- Which image(s) look most like the [Query Image Q](#) ?
 - An [Inverted File DB](#) lists all possible visual words
 - Each word points to a list of images where this word occurs
 - [Voting array](#): has as many cells as the images in the DB – each word in query image votes for an image



Vocabulary tree | tf-idf

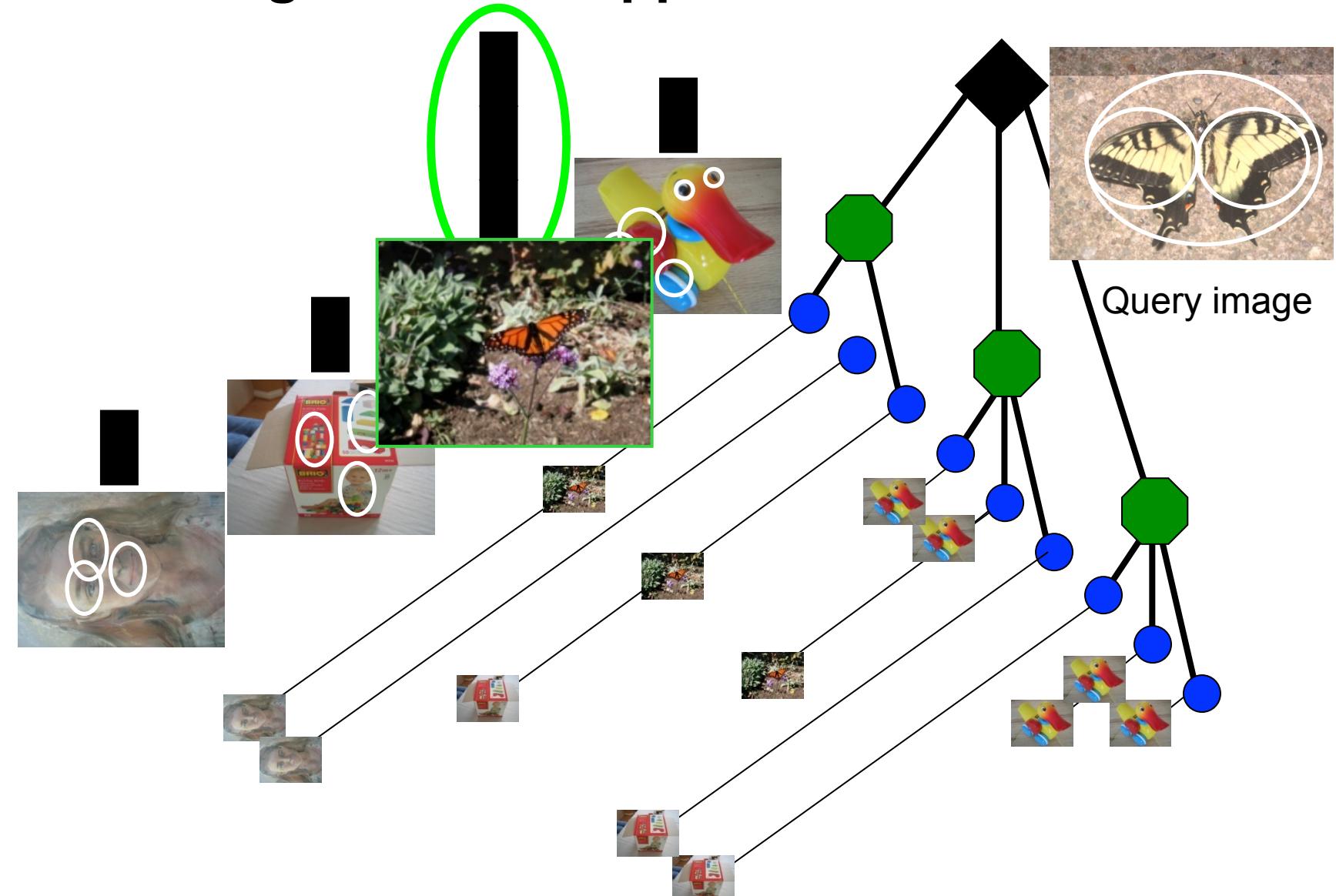
- **tf-idf:** term frequency-inverse document frequency
- measures the importance of a visual word inside a document (as part of a document DB)

When scaling such an approach to billions of images, one of two following problems occur, depending on the database density. On one hand, if the feature count per image is high and the vocabulary small, the index density is high and the number of "random" hits in the first stage results in a large number of candidate documents for which the match geometry would ideally be verified. This verification can become expensive, since it typically involves fetching additional data for each tested document. If, on the other hand, the feature count per document is low and the vocabulary size is large, the density is low, and the

- term frequency: frequency of word w_i in image j : $tf_{ij} = \frac{n_{i,j}}{\sum_k n_{k,j}}$
- inverse document frequency: $idf_i = \log \frac{|D|}{|\{d : w_i \in d\}|}$ ← No. all images (documents) ← No. all images containing w_i
- tf-idf of word w_i in image j is: $= tf_{ij} \cdot idf_i$
- Use it to weigh the importance of each word when voting for corresponding image

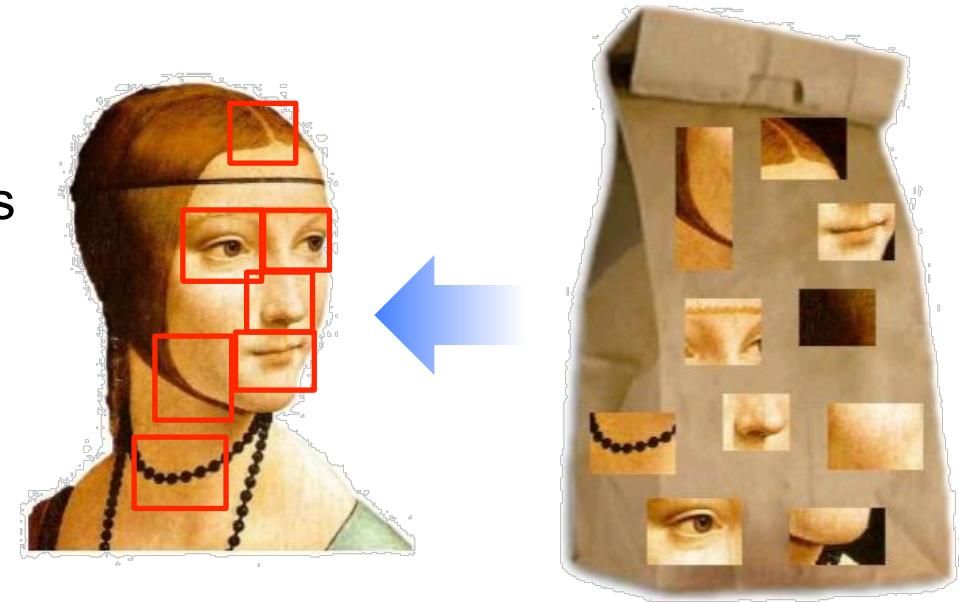
Place recognition with a Bag of Words Approach

- *What are the benefits?*
- *What are the drawbacks?*



Place Recognition | towards robust performance

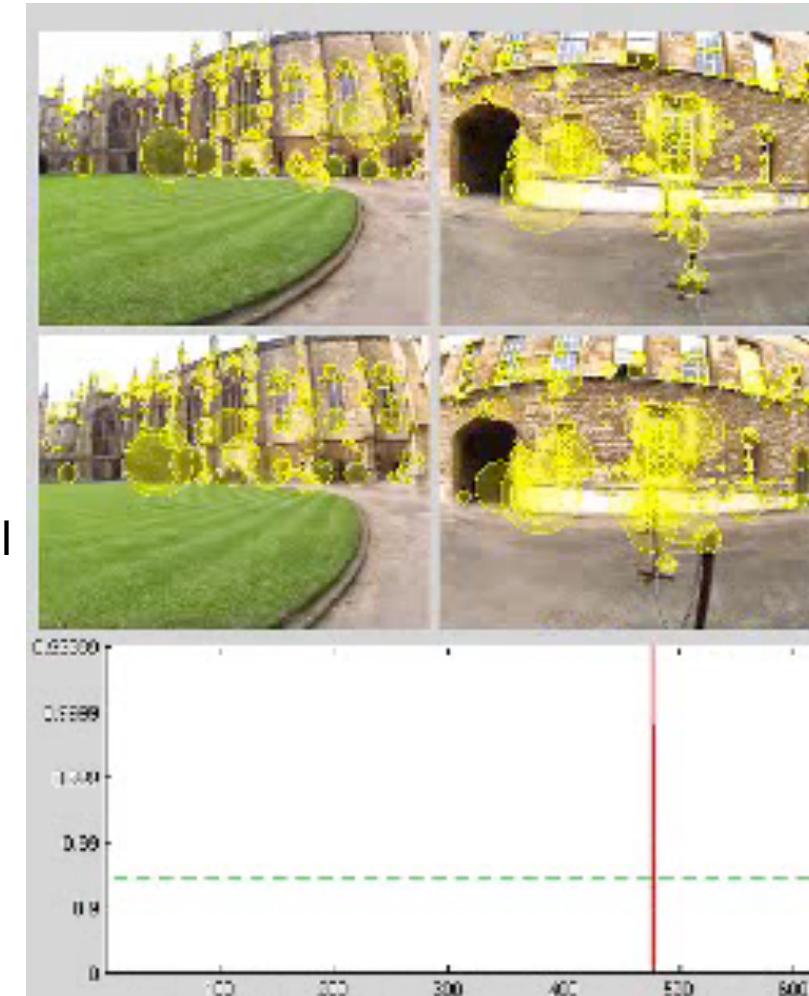
- Visual Vocabulary holds **appearance information**, but discards the **spatial** relationships between features
 - ⇒ each image is considered as a “**bag of words**”
 - ⇒ Two images with the same features shuffled around in the image will be a 100% match when using only appearance information.



Place recognition | FABMAP

[Cummins and Newman IJRR 2011]

- Place recognition for robot localization using stereo images
- Build the visual vocabulary using SURF features
- Probabilistic model of the world:
 - **World** = a set of discrete places
 - **Place** = a set of consecutive images
- At a new frame, compute:
 - $P(\text{being at a known place})$
 - $P(\text{being at a new place})$
- Captures the dependencies of visual words to distinguish the most characteristic structure of each scene (using the Chow-Liu tree)
- Code available online:
<http://www.robots.ox.ac.uk/~mjc/Software.htm>



Place recognition | FABMAP examples

p = probability of images coming from the same place



(a) $p=0.53$



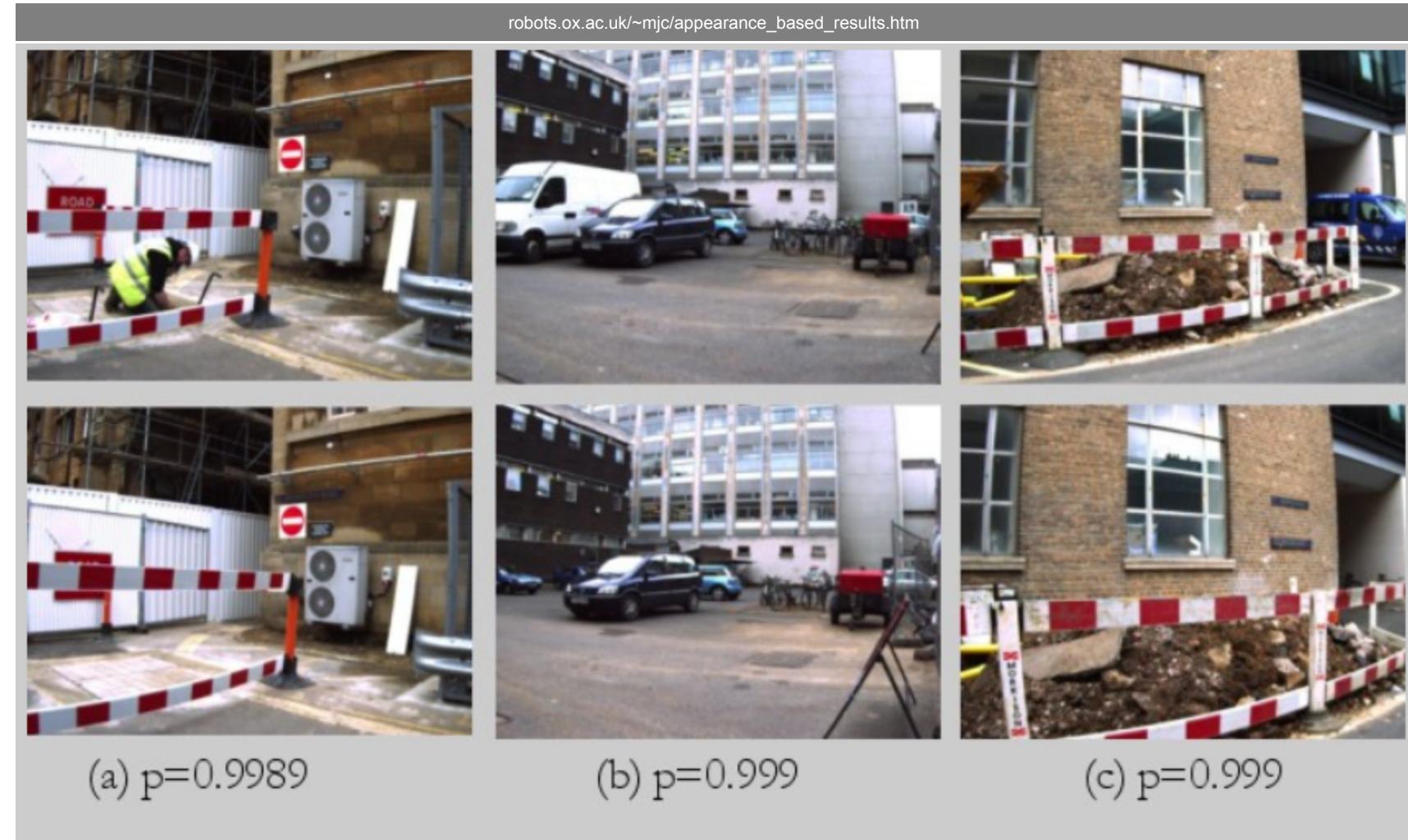
(b) $p=0.30$



(c) $p=0.31$

Place recognition | FABMAP examples

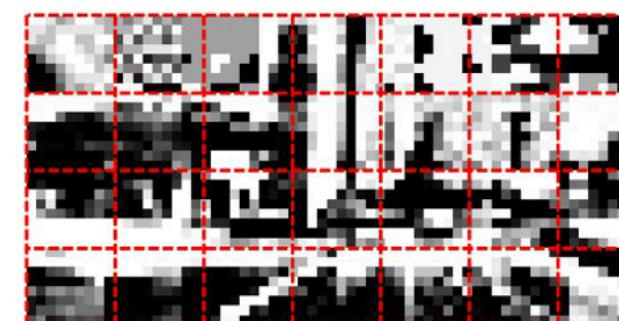
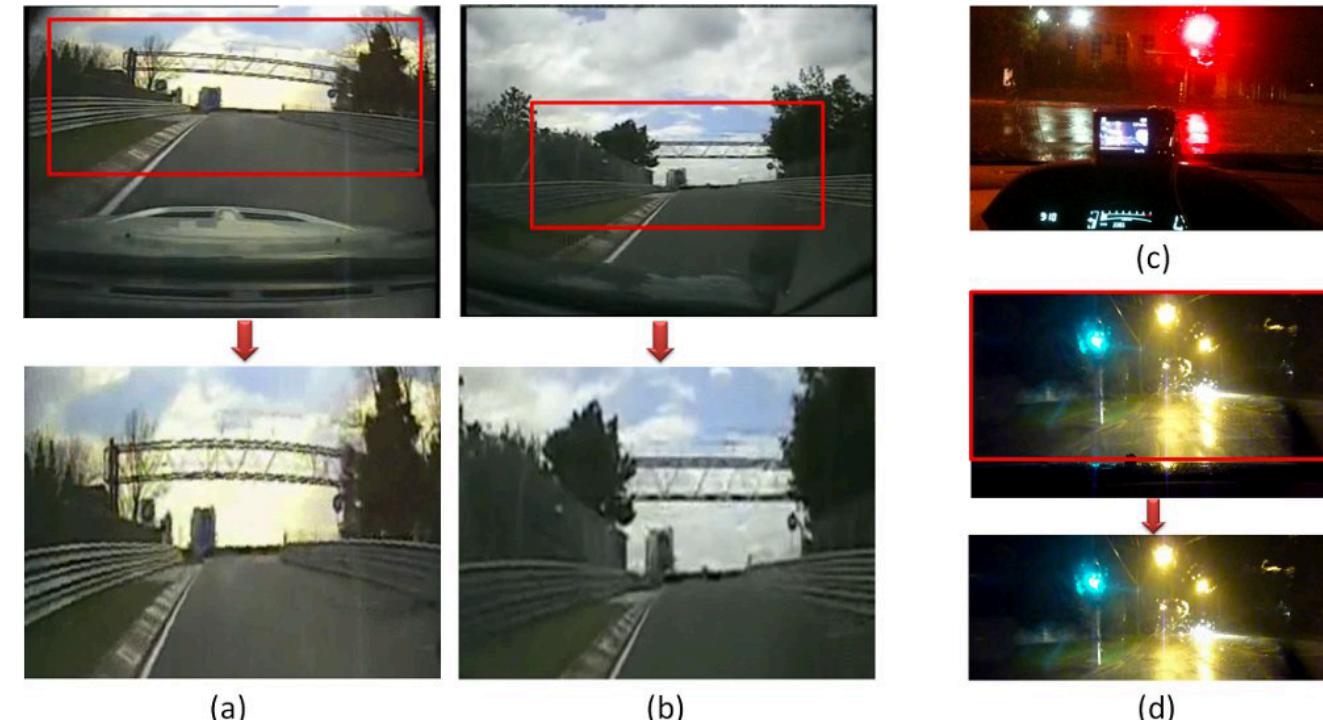
p = probability of images coming from the same place



Place recognition | SeqSLAM [Milford & Wyeth, ICRA 2012]

“Visual Route-Based Navigation for Sunny Summer Days and Stormy Winter Nights”

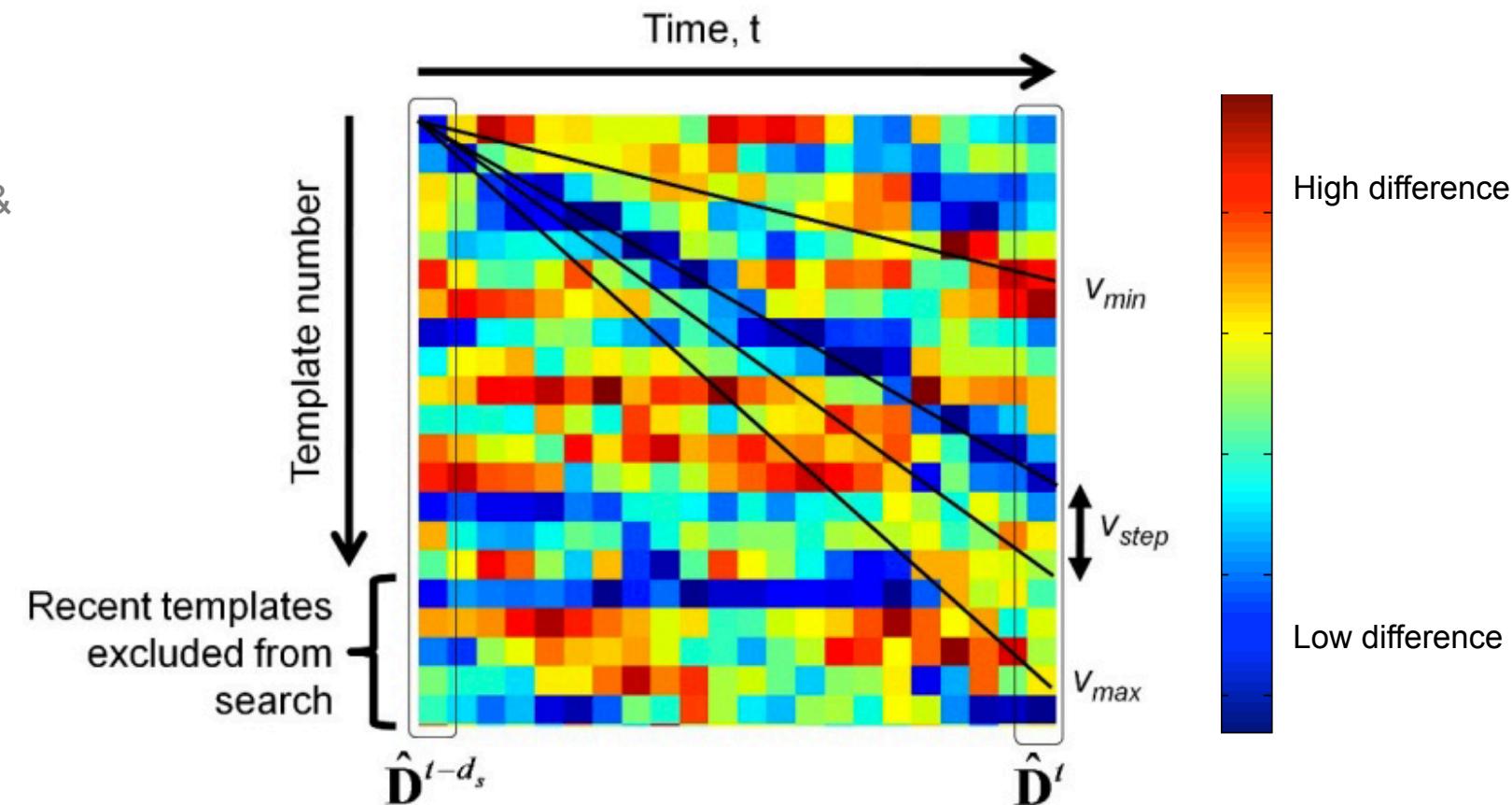
- Whole-image descriptor
- Image-preprocessing:
 - Crop,
 - Patch-normalize on a regular grid, &
 - Down-sample



Place recognition | SeqSLAM [Milford & Wyeth, ICRA 2012]

“Visual Route-Based Navigation for Sunny Summer Days and Stormy Winter Nights”

- Whole-image descriptor
- Image-preprocessing:
 - Crop,
 - Patch-normalize on a regular grid, &
 - Down-sample
- Build image-difference matrix



Place recognition | SeqSLAM [Milford & Wyeth, ICRA 2012]

“Visual Route-Based Navigation for Sunny Summer Days and Stormy Winter Nights”

- Whole-image descriptor
- Image-preprocessing:
 - Crop,
 - Patch-normalize,
 - Down-sample
- Build image-difference matrix
- Works with extreme illumination changes!
- Recognizes loop when scene is visited:
 - at the same speed,
 - from the same viewpoint
 - Relies on long sequence of matching frames

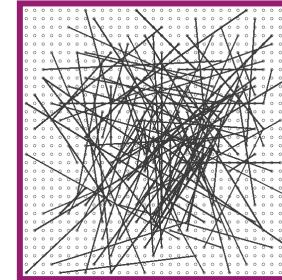
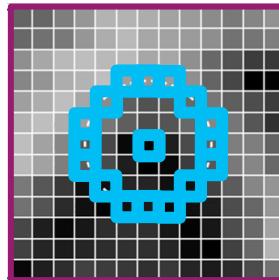


Place Recognition | bags of binary words

- **Bag of Binary Words (BoBW):**

[Gálvez-López & Tardós, TRO 2012]

- FAST Keypoints + BRIEF descriptor



- Very fast
- Works for up-right images
 - **not** viewpoint-invariant

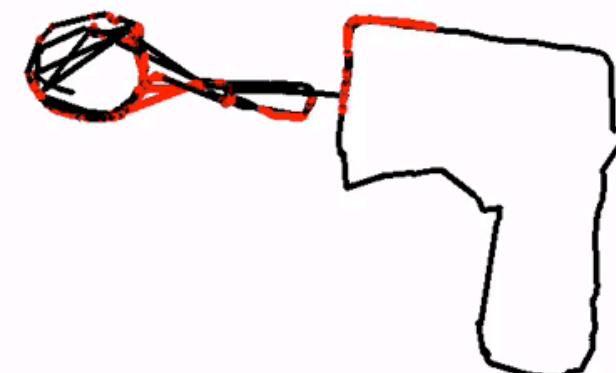
Current image



Loop detected



Execution time: 25.7 ms



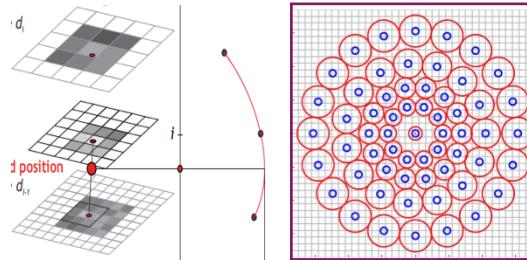
Place Recognition | towards robust performance

- Visual Vocabulary holds **appearance information**, but discards the **spatial** relationships between features
 - ⇒ each image is considered as a “**bag of words**”
 - ⇒ Two images with the same features shuffled around in the image will be a 100% match when using only appearance information.
- If different arrangements of the same features are expected then one might use **geometric verification**
 - for example: test the k most similar images to the query image for geometric consistency (e.g. using RANSAC)



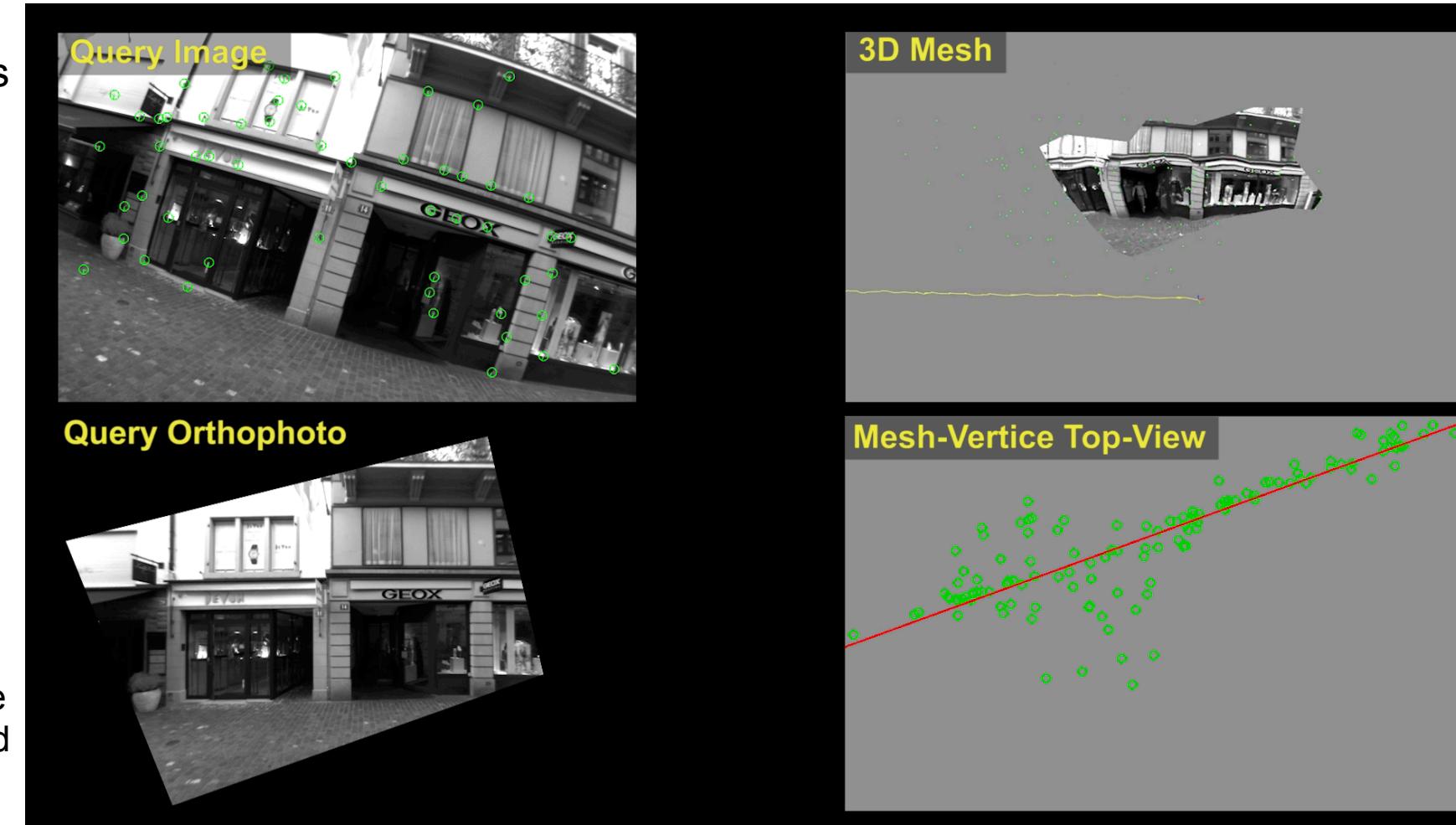
Place Recognition | using orthophotos

- Appearance tests:
BoBW with **BRISK** features



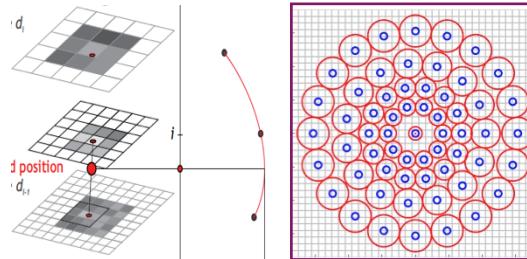
- Use feature detections in local scene to:
 - construct a local mesh, and
 - identify the scene's most dominant plane
- Query Orthophoto =
A projection of the query image to the dominant plane identified

[Maffra et al., 3DV 2017]: "Loop-closure detection in urban scenes for Autonomous Robot Navigation"



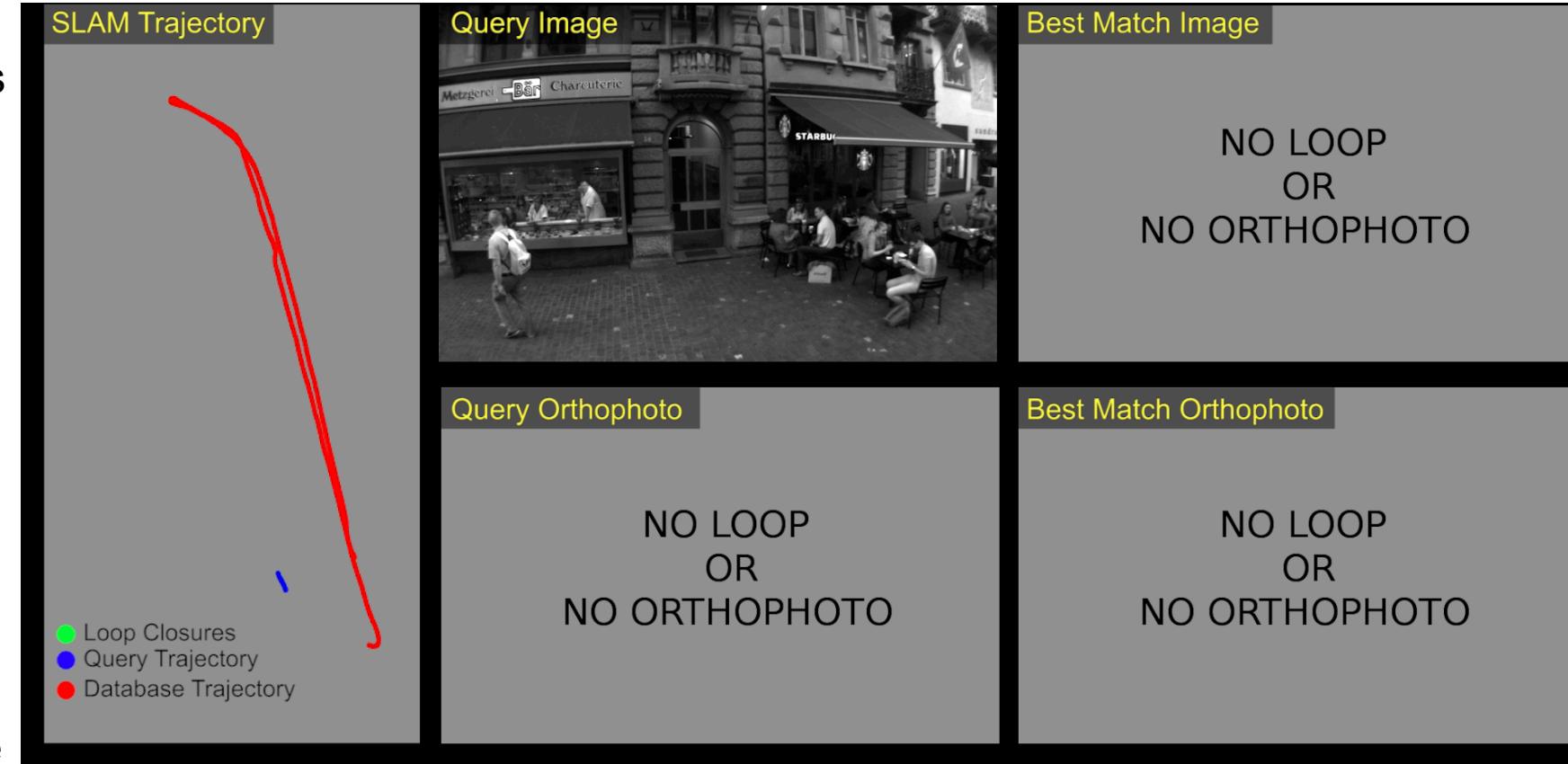
Place Recognition | using orthophotos

- Appearance tests:
BoBW with **BRISK** features



- Use feature detections in local scene to:
 - construct a local mesh, and
 - identify the scene's most dominant plane
- Query Orthophoto =
A projection of the query image to the dominant plane identified

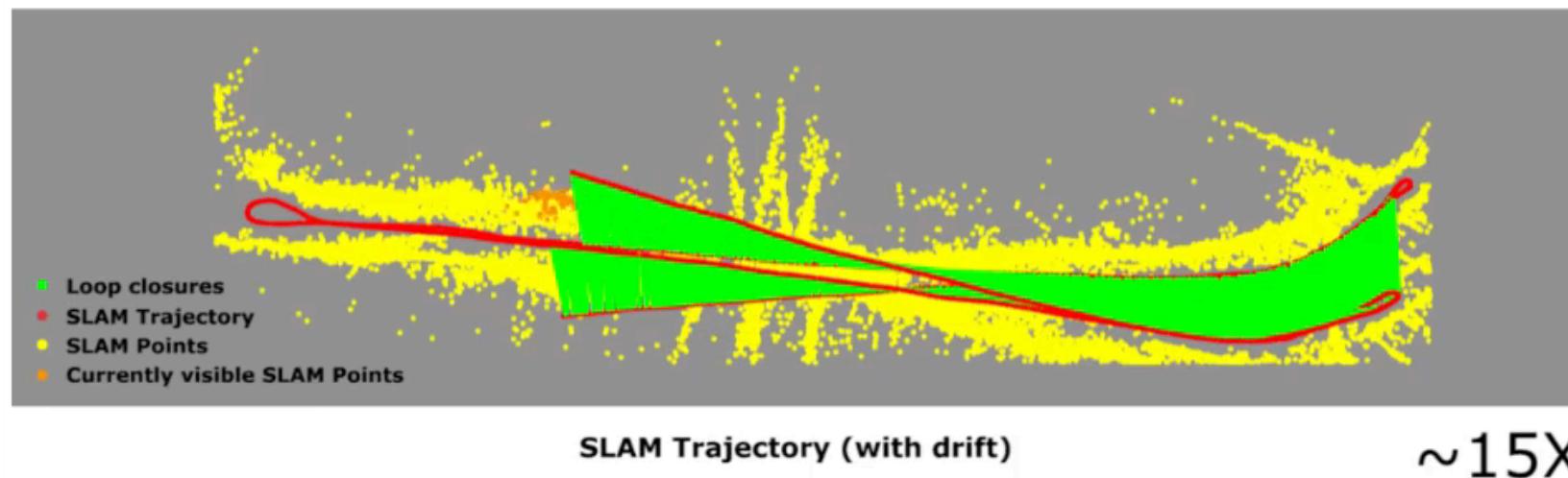
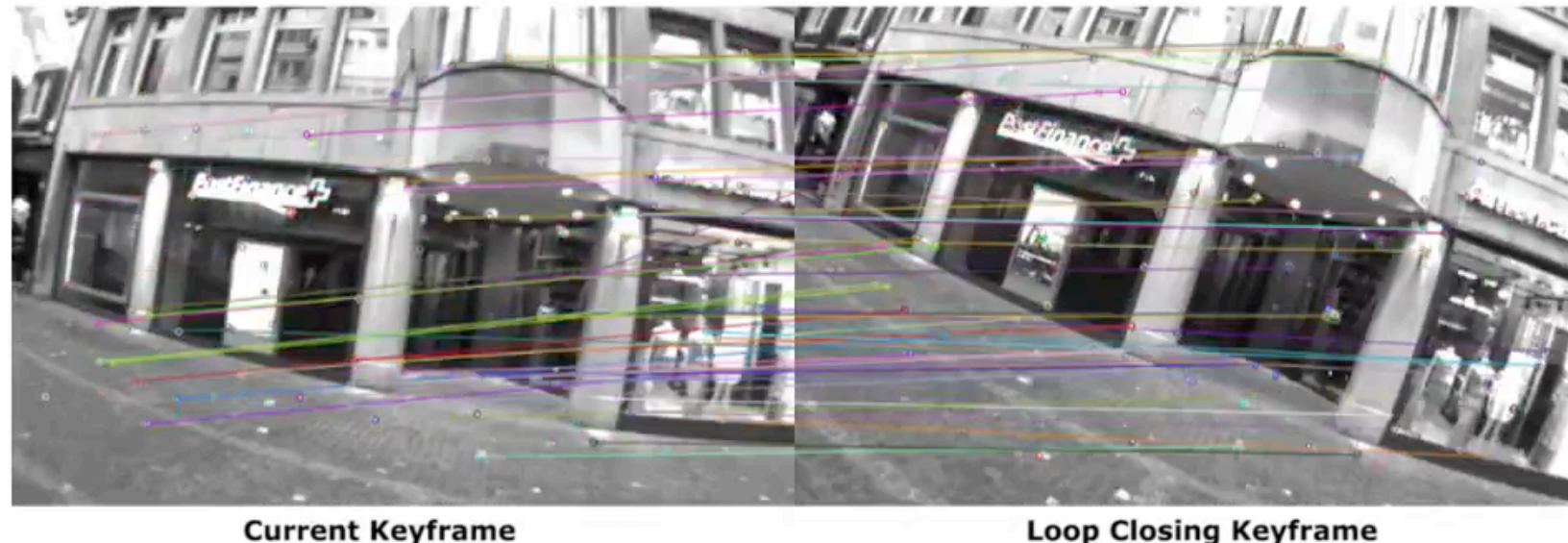
[Maffra et al., 3DV 2017]: "Loop-closure detection in urban scenes for Autonomous Robot Navigation"



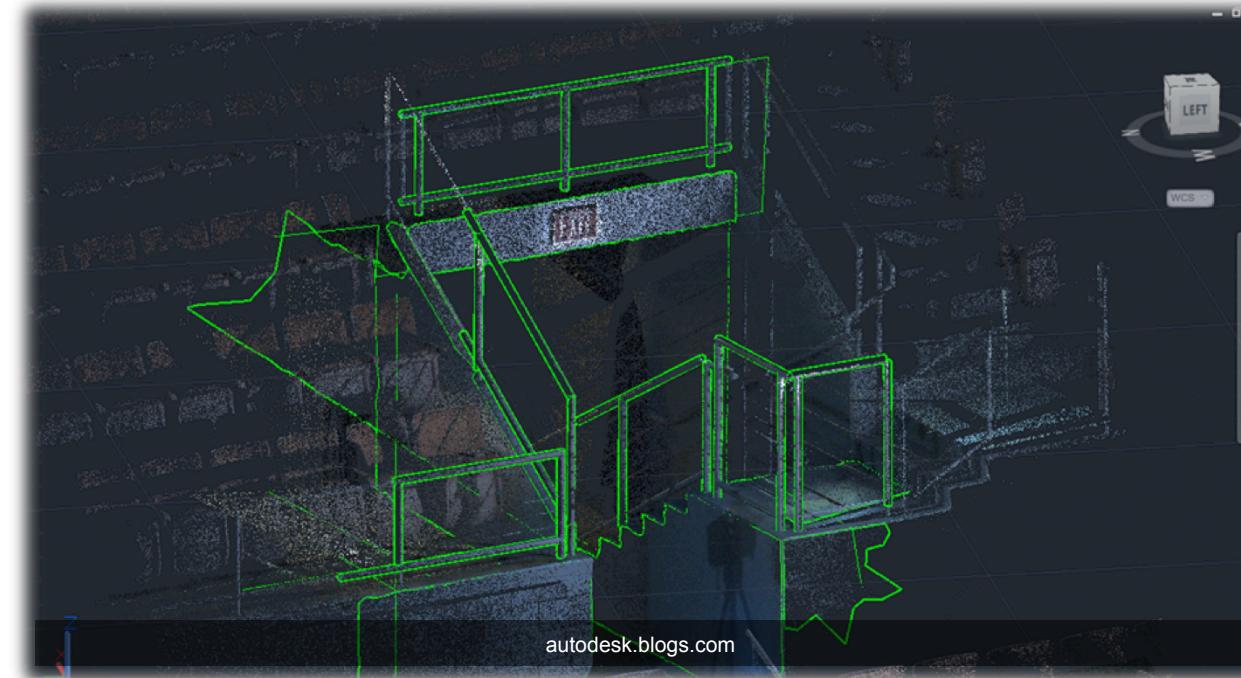
Place Recognition | with geometric checks

- Appearance tests:
BoBW with **BRISK** features
- Geometric tests:
compare local geometry of query to database in 2D (image) and in 3D (local map)
- If both tests are successful, loop-closure can be accepted
- Public datasets: www.v4rl.ethz.ch with challenging viewpoint, illumination & situational changes (e.g. different logos, pedestrians)

[Maffra et al., ICRA 2018]: “Viewpoint-tolerant Place Recognition combining 2D and 3D information for UAV navigation”



- Uncertainties – Representation & Propagation
- Line Extraction from point clouds



Uncertainty representation | importance

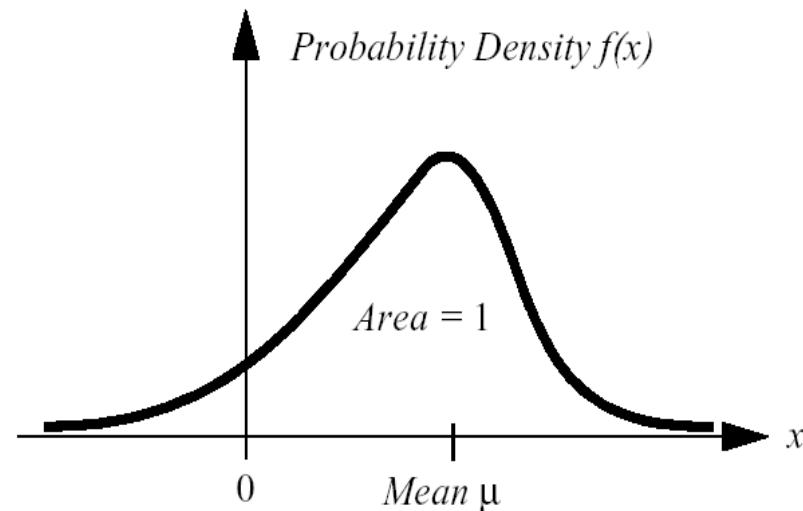
Section 4.1.3 of the book

- Sensing in the real world is **always uncertain**
 - How can uncertainty be **represented** or quantified?
 - How does uncertainty **propagate**?
i.e. given uncertain inputs into a system, what is the uncertainty in the output?
 - What is the merit of all this for mobile robotics?



Uncertainty representation

- Use a Probability Density Function (PDF) to characterize the statistical properties of a variable x :



$$\int_{-\infty}^{\infty} f(x)dx = 1$$

- Expected value of variable x :

$$\mu = E[X] = \int_{-\infty}^{\infty} xf(x)dx$$

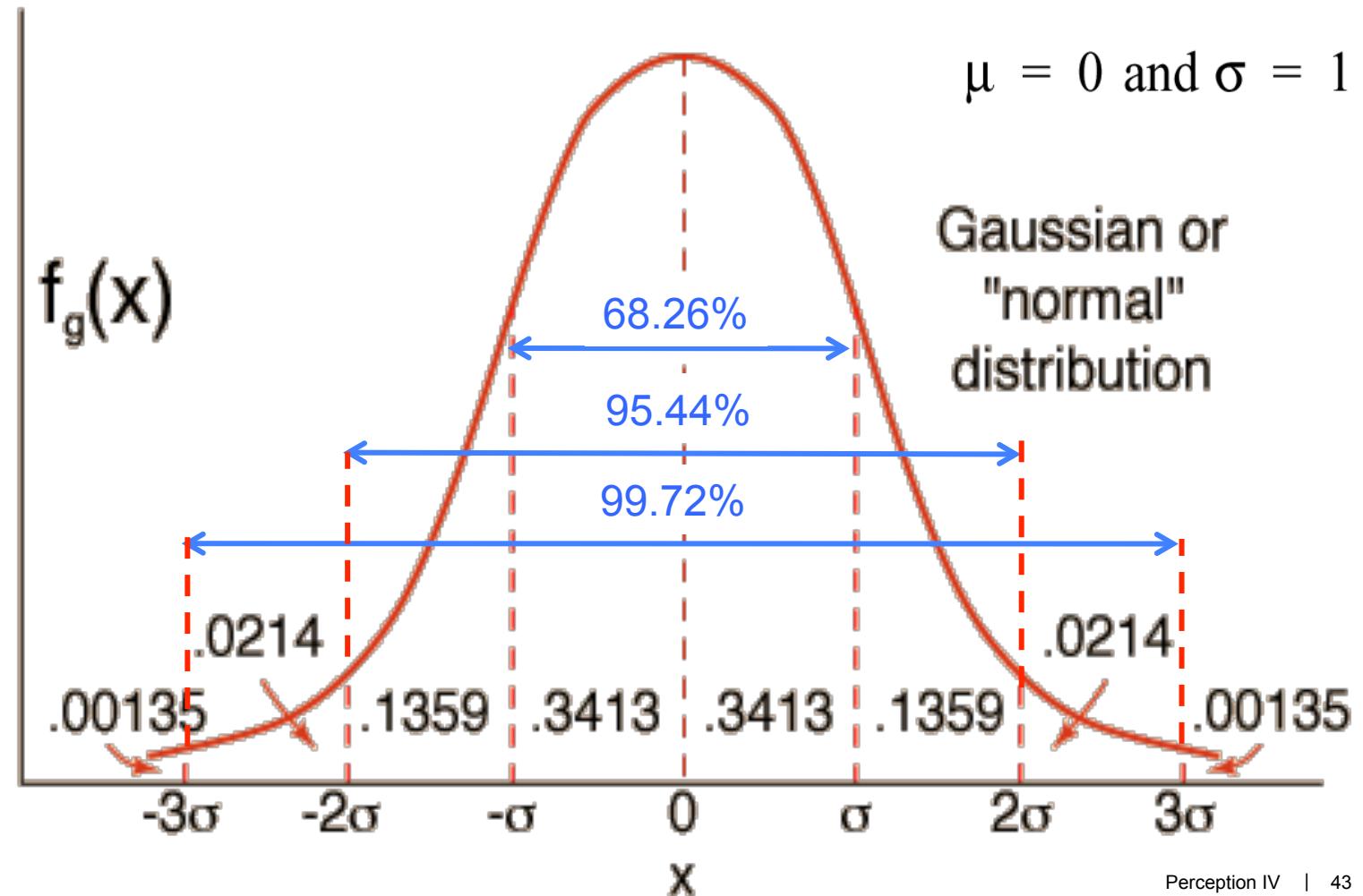
- Variance of variable x :

$$\sigma^2 = \int_{-\infty}^{\infty} (x - \mu)^2 f(x)dx$$

Uncertainty representation | gaussian distribution

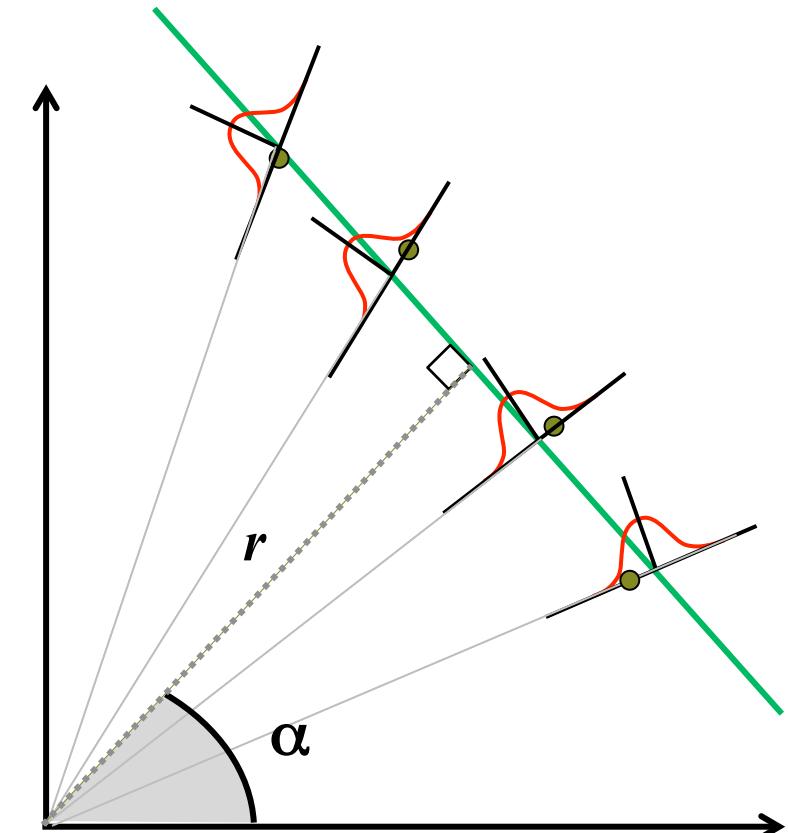
- Most common PDF for characterizing uncertainties: Gaussian

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$



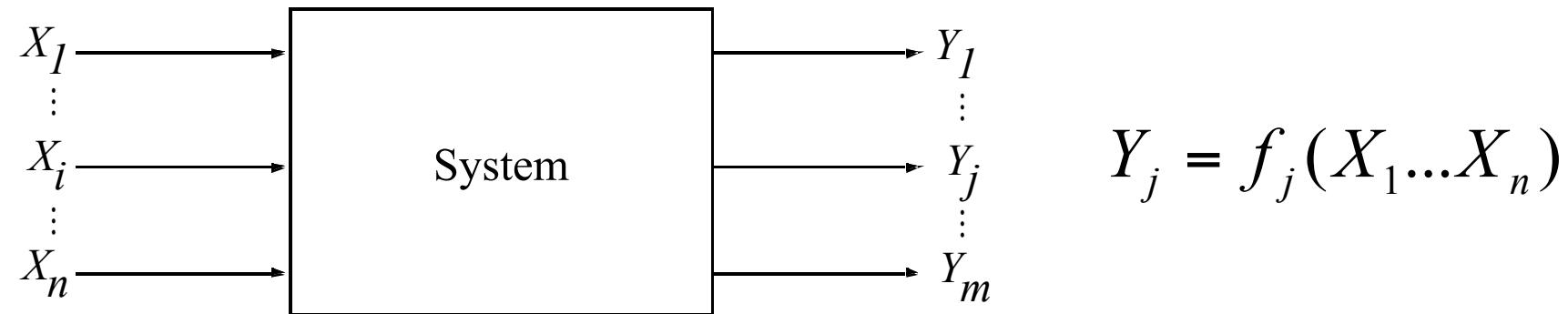
Uncertainty representation | the error propagation law

- Imagine extracting a line based on point measurements with uncertainties.
- Model parameters in polar coordinates
[(r, α) uniquely identifies a line]
- What is the **uncertainty** of the extracted line knowing the uncertainties of the measurement points that contribute to it ?



Uncertainty representation | the error propagation law

Error propagation in a multiple-input multiple-output system with n inputs and m outputs



Uncertainty representation | the error propagation law

- 1D case of a nonlinear error propagation problem:
- It can be shown that the output covariance matrix C_{YY} is given by the error propagation law:

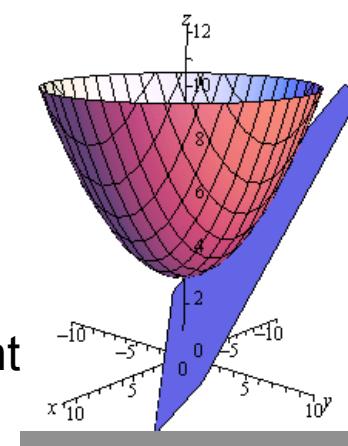
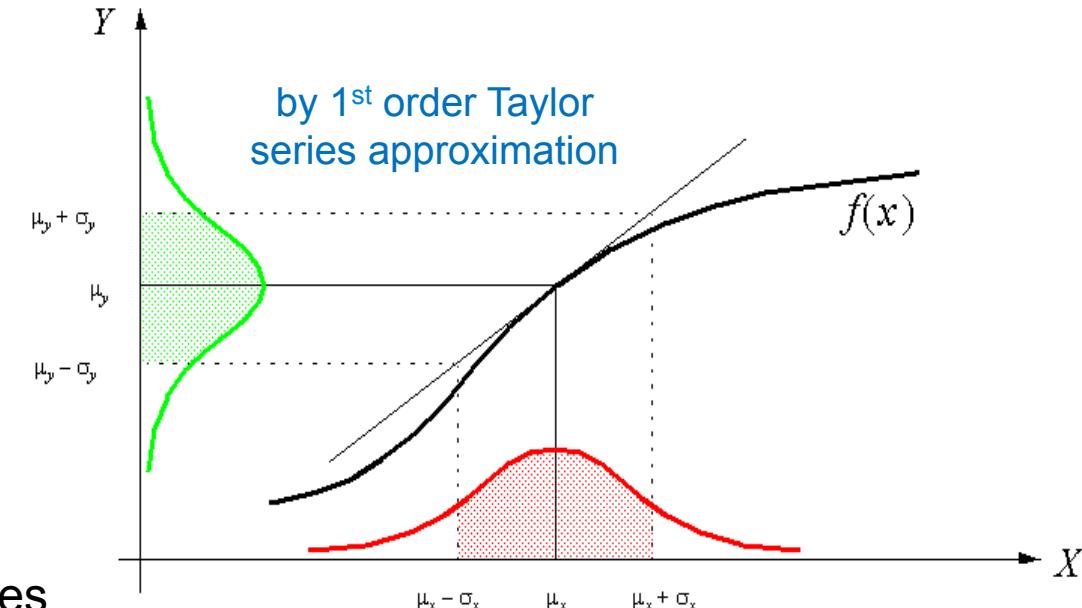
$$C_{YY} = F_{YX} C_{XX} F_{XY} \quad , F_{XY} = F_{YX}^T$$

where

- C_{XX} : covariance matrix representing the input uncertainties
- C_{YY} : covariance matrix representing the propagated uncertainties for the outputs.
- F_{YX} : is the **Jacobian** matrix defined as:

$$F_{YX} = \begin{pmatrix} \frac{\partial f_1}{\partial X_1} & \dots & \frac{\partial f_m}{\partial X_1} \\ \vdots & \dots & \vdots \\ \frac{\partial f_m}{\partial X_1} & \dots & \frac{\partial f_m}{\partial X_n} \end{pmatrix}$$

- Defines the orientation of the **tangent** line/plane/hyper-plane at a given point



Uncertainty representation | line extraction

- Point-Line distance

$$\rho_i \cos(\theta_i - \alpha) - r = d_i$$

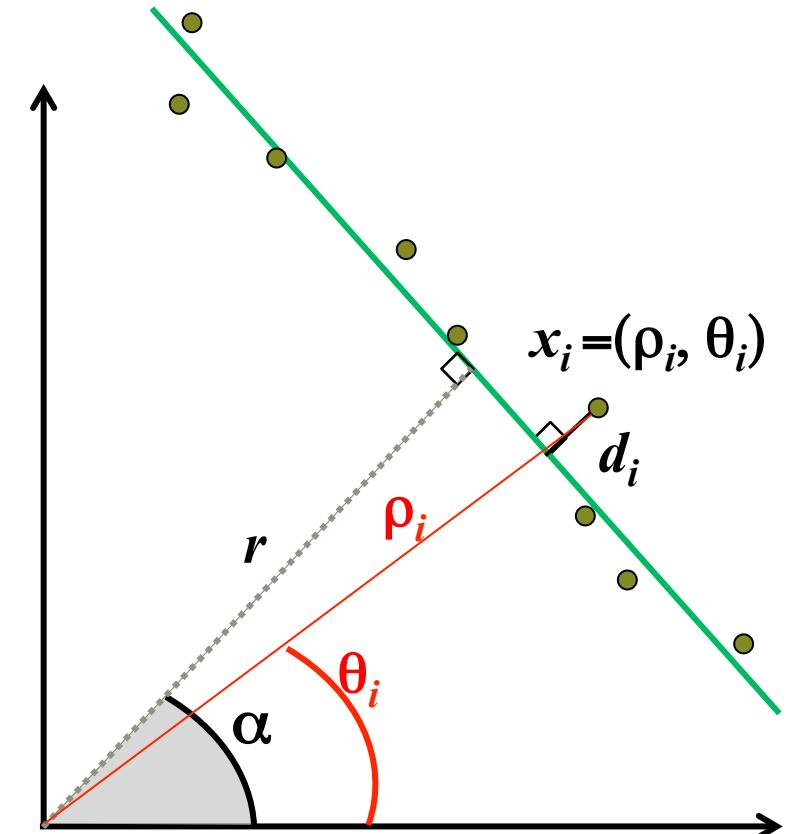
- If each measurement is equally uncertain then sum of sq. errors:

$$S = \sum_i d_i^2 = \sum_i (\rho_i \cos(\theta_i - \alpha) - r)^2$$

- Goal: minimize S when selecting (r, α) \Rightarrow solve the system

$$\frac{\partial S}{\partial \alpha} = 0 \quad \frac{\partial S}{\partial r} = 0$$

- “Unweighted Least Squares”**



Uncertainty representation | line extraction

- Point-Line distance

$$\rho_i \cos(\theta_i - \alpha) - r = d_i$$

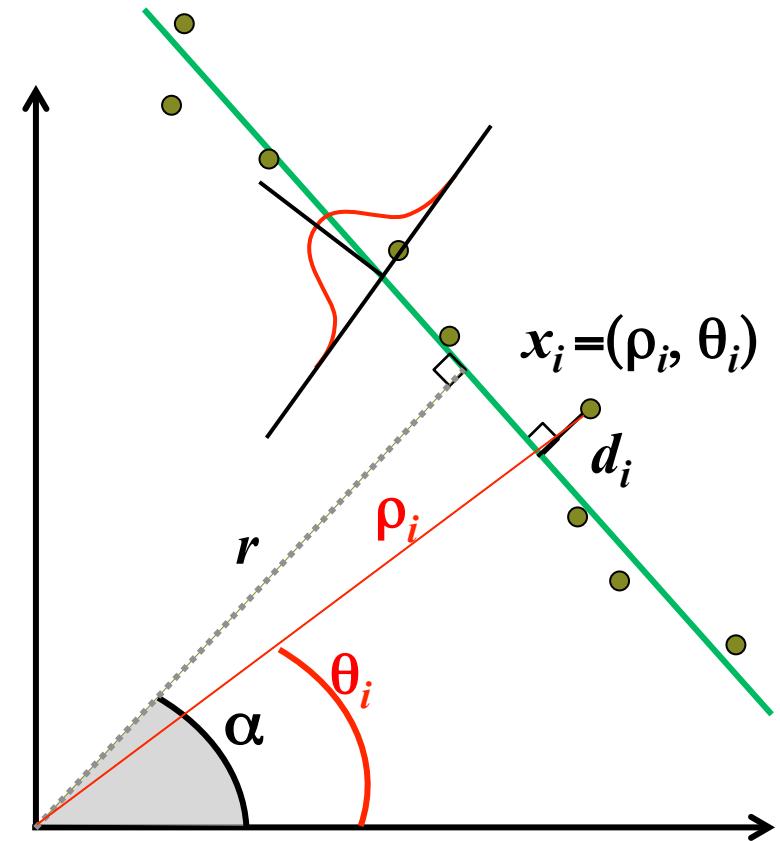
- Each sensor measurement, may have its own, unique uncertainty σ_i^2

$$S = \sum w_i d_i^2 = \sum w_i (\rho_i \cos(\theta_i - \alpha) - r)^2$$

$$w_i = 1/\sigma_i^2$$

- “Weighted Least Squares”**

$$\frac{\partial S}{\partial \alpha} = 0 \quad \frac{\partial S}{\partial r} = 0$$



Uncertainty representation | line extraction

- Weighted least squares and solving the system:

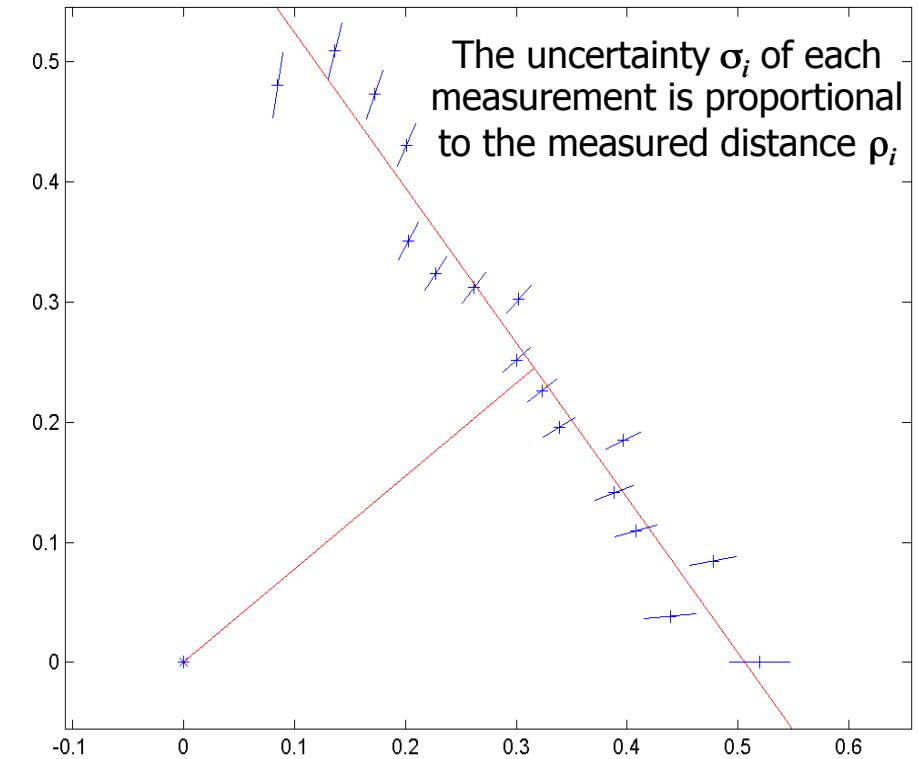
$$\frac{\partial S}{\partial \alpha} = 0 \quad \frac{\partial S}{\partial r} = 0$$

- Gives the line parameters:

$$\alpha = \frac{1}{2} \text{atan} \left(\frac{\sum w_i \rho_i^2 \sin 2\theta_i - \frac{2}{\sum w_i} \sum \sum w_i w_j \rho_i \rho_j \cos \theta_i \sin \theta_j}{\sum w_i \rho_i^2 \cos 2\theta_i - \frac{1}{\sum w_i} \sum \sum w_i w_j \rho_i \rho_j \cos(\theta_i + \theta_j)} \right)$$

$$r = \frac{\sum w_i \rho_i \cos(\theta_i - \alpha)}{\sum w_i}$$

- If $\begin{cases} \rho_i \sim N(\hat{\rho}_i, \sigma_{\rho_i}^2) \\ \theta_i \sim N(\hat{\theta}_i, \sigma_{\theta_i}^2) \end{cases}$ what is the uncertainty in the line (r, α) ?



Uncertainty representation | line extraction

The uncertainty of **each measurement** $x_i = (\rho_i, \theta_i)$ is described by the covariance matrix:

Assuming that ρ_i, θ_i are independent

$$C_{xx_i} = \begin{bmatrix} \sigma_{\rho_i}^2 & 0 \\ 0 & \sigma_{\theta_i}^2 \end{bmatrix}$$

The uncertainty in the **line** (r, α) is described by the covariance matrix: $C_{ll} = \begin{bmatrix} \sigma_\alpha^2 & \sigma_{\alpha r} \\ \sigma_{r\alpha} & \sigma_r^2 \end{bmatrix} = ?$

Define: $C_{xx} = \begin{bmatrix} \text{diag}(\sigma_\rho^2) & 0 \\ 0 & \text{diag}(\sigma_\theta^2) \end{bmatrix} = \begin{bmatrix} \dots & 0 & 0 & \dots & 0 & 0 & \dots \\ \dots & \sigma_{\rho_i}^2 & 0 & \dots & 0 & 0 & \dots \\ \dots & 0 & \sigma_{\rho_{i+1}}^2 & \dots & 0 & 0 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & 0 & 0 & \dots & \sigma_{\theta_i}^2 & 0 & \dots \\ \dots & 0 & 0 & \dots & 0 & \sigma_{\theta_{i+1}}^2 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \end{bmatrix}$

Error Propagation Law

$$C_{ll} = F_{lx} C_{xx} F_{xl}$$

Jacobian:

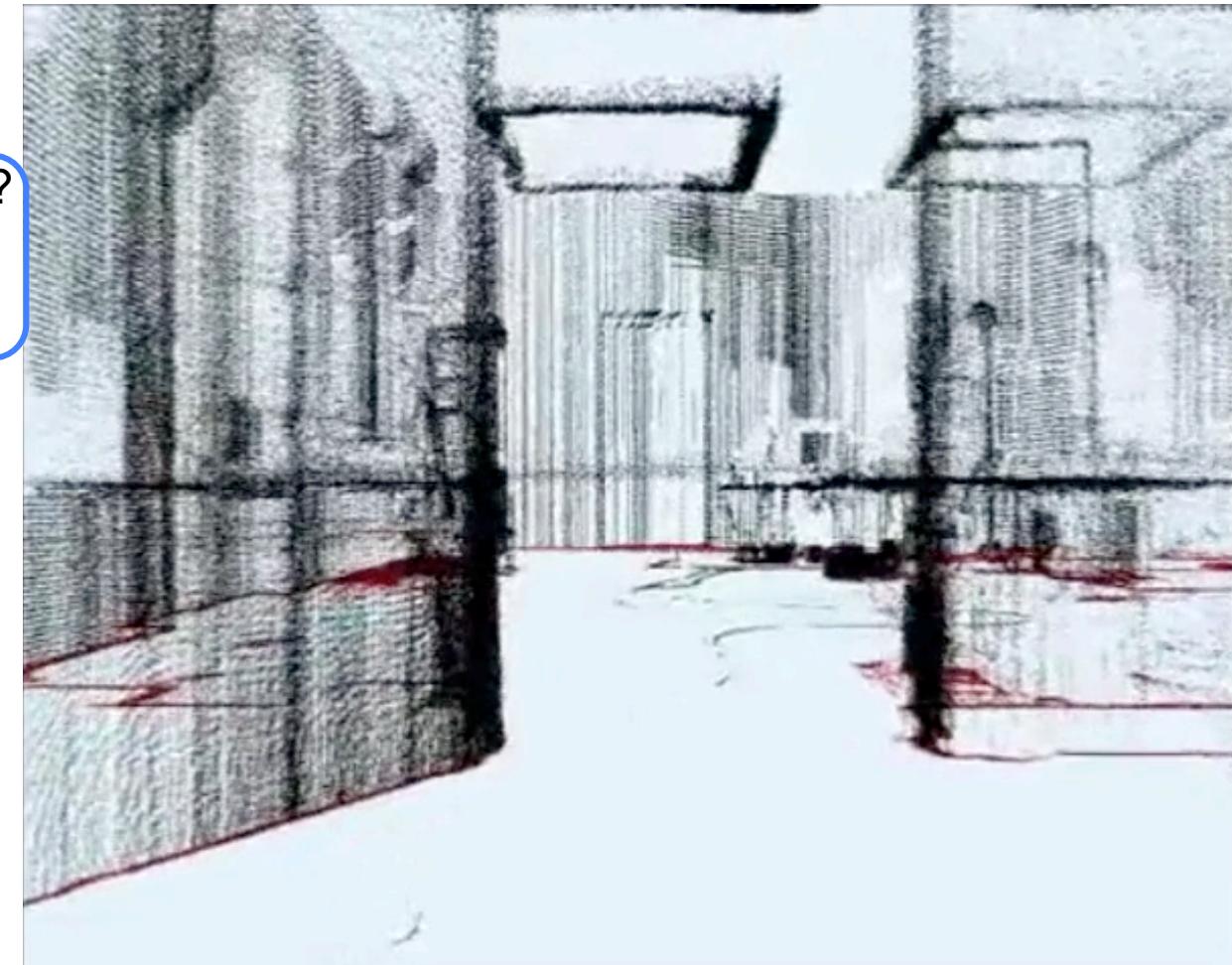
$$F_{lx} = \begin{bmatrix} \dots & \frac{\partial \alpha}{\partial \rho_i} & \frac{\partial \alpha}{\partial \rho_{i+1}} & \dots & \frac{\partial \alpha}{\partial \theta_i} & \frac{\partial \alpha}{\partial \theta_{i+1}} & \dots \\ \dots & \frac{\partial r}{\partial \rho_i} & \frac{\partial r}{\partial \rho_{i+1}} & \dots & \frac{\partial r}{\partial \theta_i} & \frac{\partial r}{\partial \theta_{i+1}} & \dots \end{bmatrix}^{2n \times 2n}$$

Line extraction from a point cloud

Extract lines from a point cloud (e.g. range scan)

<http://www.youtube.com/watch?v=wV8frjLqtIA&feature=related>

- Three main problems:
 - How many lines are there?
 - **Segmentation:** Which points belong to which line?
 - **Line Fitting/Extraction:** Given points that belong to a line, how to estimate the line parameters?
- Algorithms we will see:
 1. Split-and-merge
 2. Linear regression
 3. RANSAC
 4. Hough-Transform



Line extraction | 1. split-and-merge (standard)

- Popular algorithm, originates from Computer Vision.
- A recursive procedure of fitting and splitting.

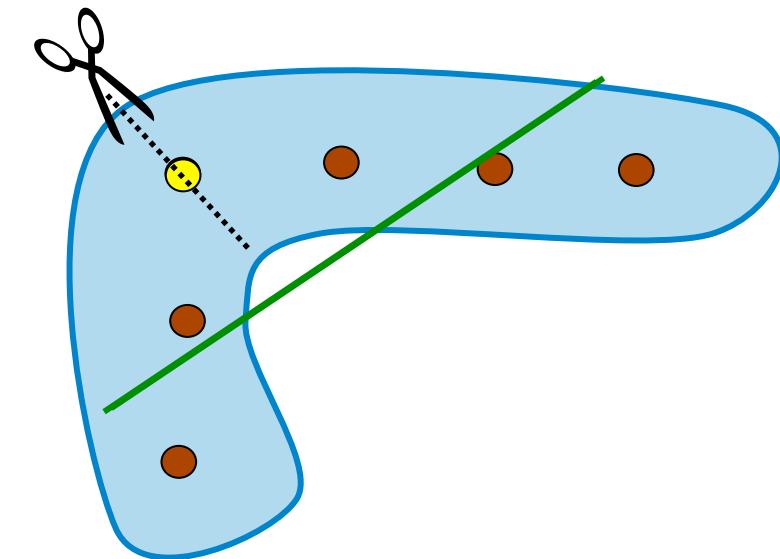
Let **S** be the set of all data points

Split

- Fit a line to points in current set **S**
- Find the most distant point to the line
- If distance > threshold \Rightarrow split set & repeat with left and right point sets

Merge

- If two consecutive segments are collinear enough, obtain the common line and find the most distant point
- If distance \leq threshold, merge both segments



Line extraction | 1. split-and-merge (standard)

- Popular algorithm, originates from Computer Vision.
- A recursive procedure of fitting and splitting.

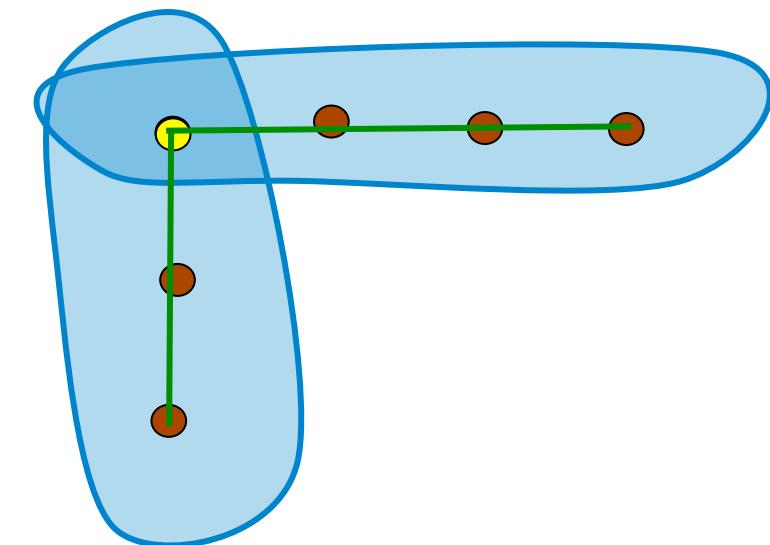
Let \mathbf{S} be the set of all data points

Split

- Fit a line to points in current set \mathbf{S}
- Find the most distant point to the line
- If distance > threshold \Rightarrow split set & repeat with left and right point sets

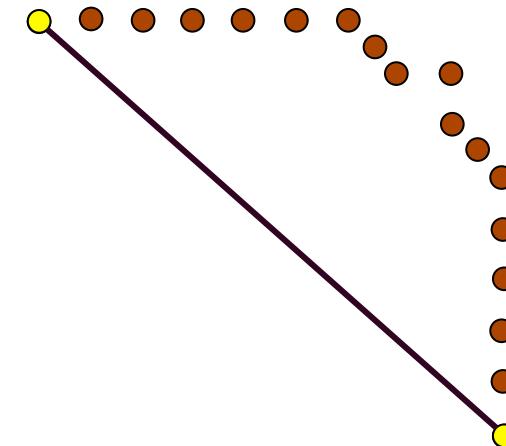
Merge

- If two consecutive segments are collinear enough, obtain the common line and find the most distant point
- If distance \leq threshold, merge both segments



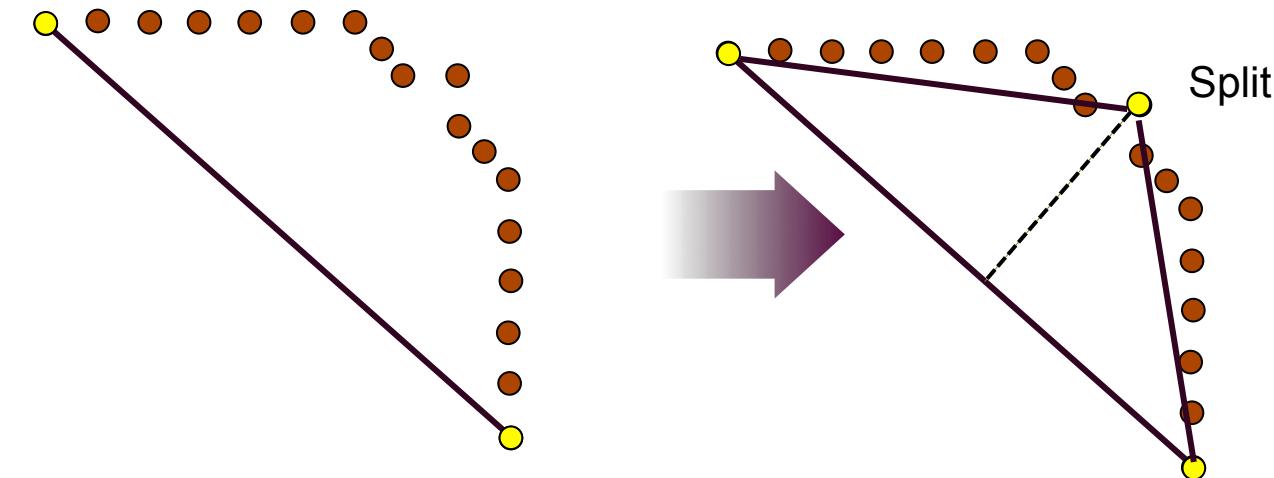
Line extraction | 1. split-and-merge (iterative end-point-fit)

Iterative end-point-fit:
simply connects the
end points for line
fitting



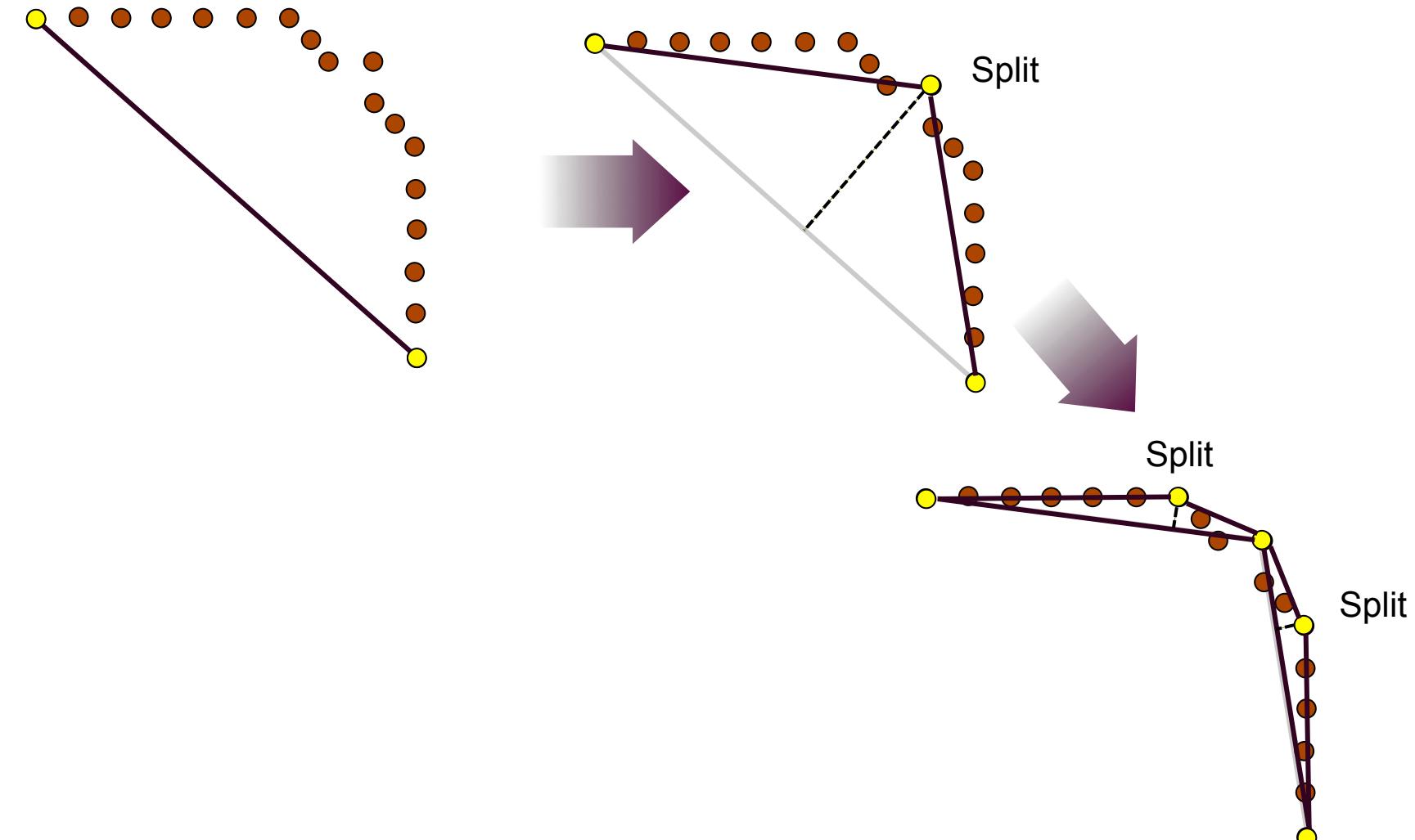
Line extraction | 1. split-and-merge (iterative end-point-fit)

Iterative end-point-fit:
simply connects the
end points for line
fitting



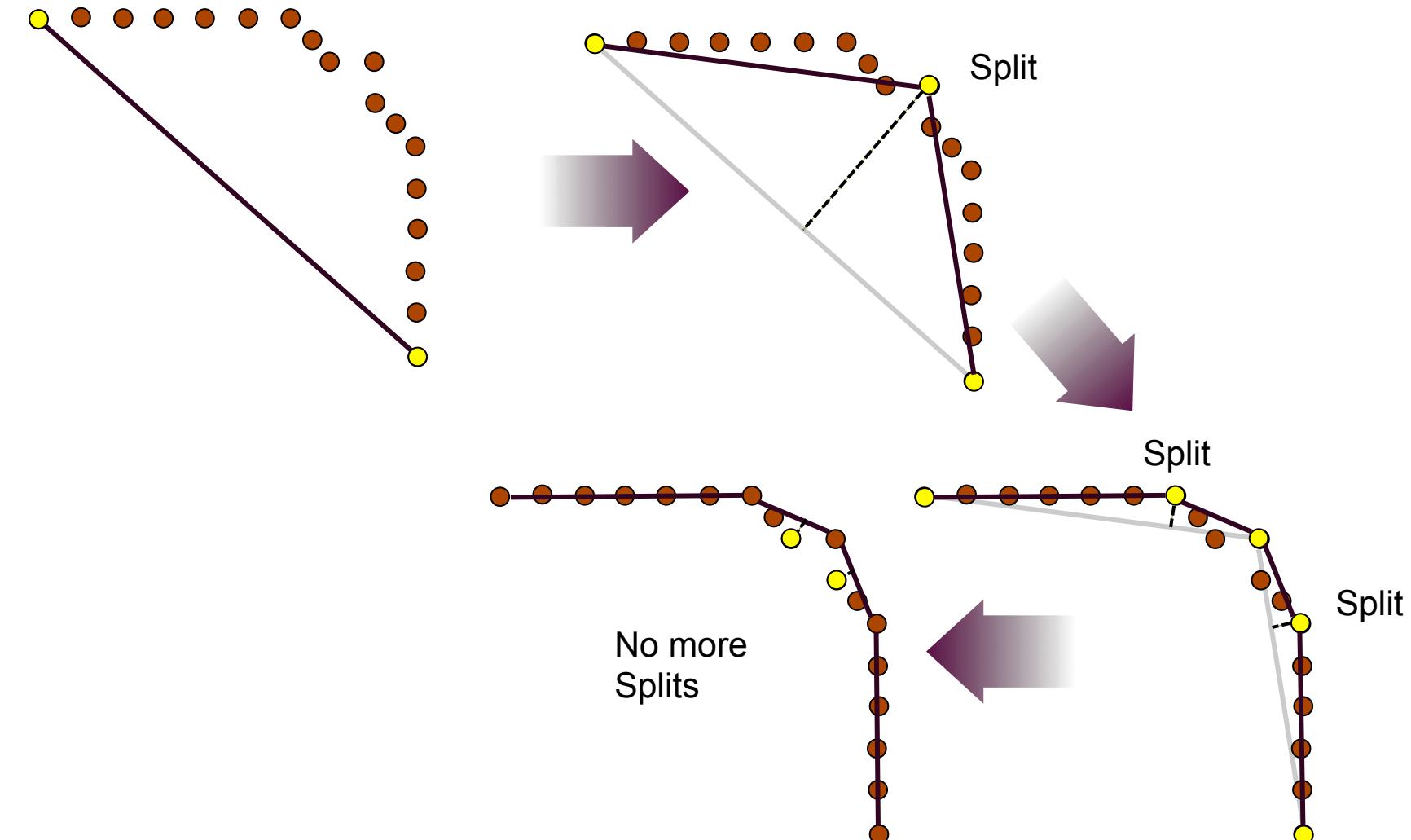
Line extraction | 1. split-and-merge (iterative end-point-fit)

Iterative end-point-fit:
simply connects the
end points for line
fitting



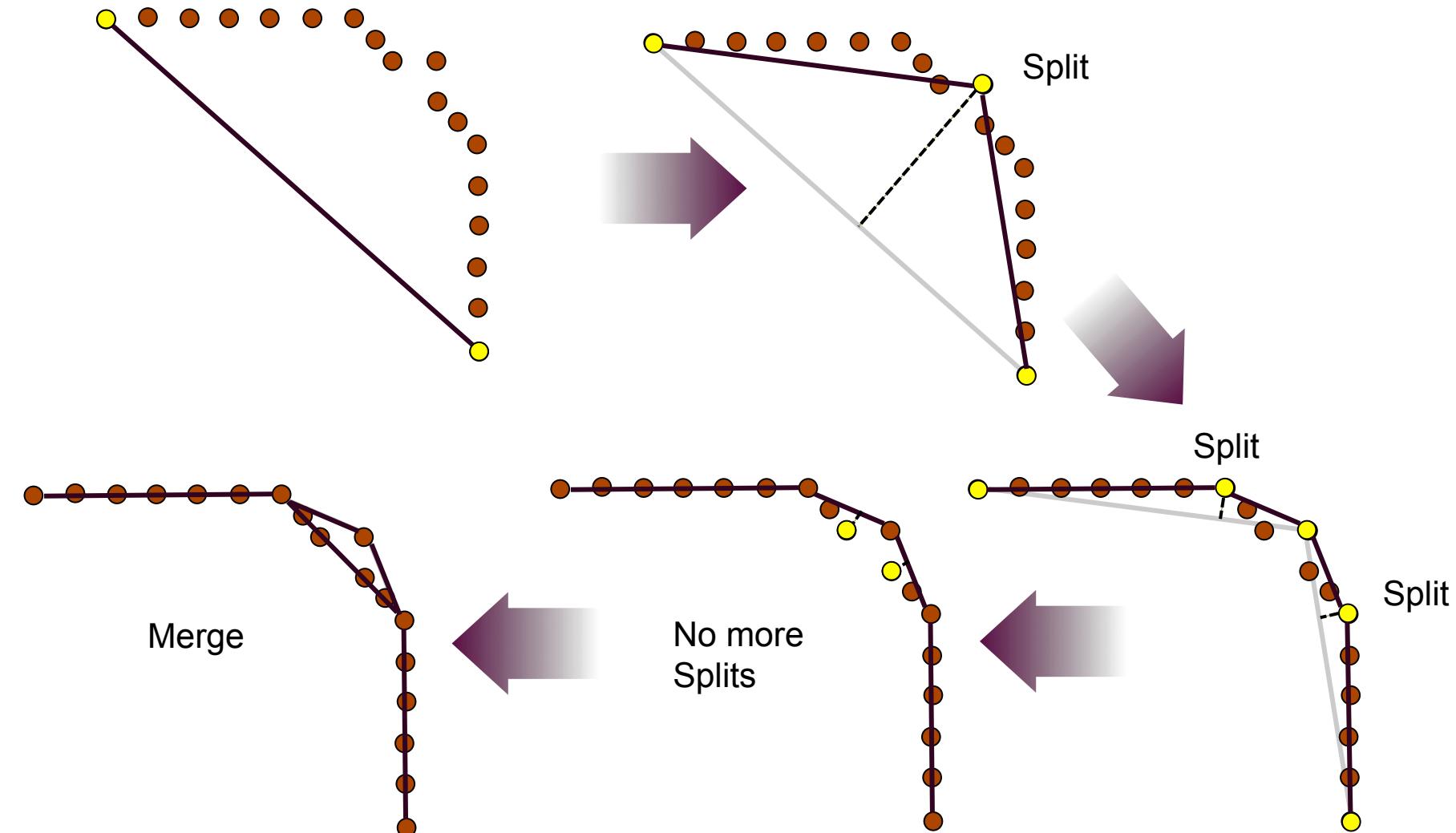
Line extraction | 1. split-and-merge (iterative end-point-fit)

Iterative end-point-fit:
simply connects the
end points for line
fitting



Line extraction | 1. split-and-merge (iterative end-point-fit)

Iterative end-point-fit:
simply connects the
end points for line
fitting

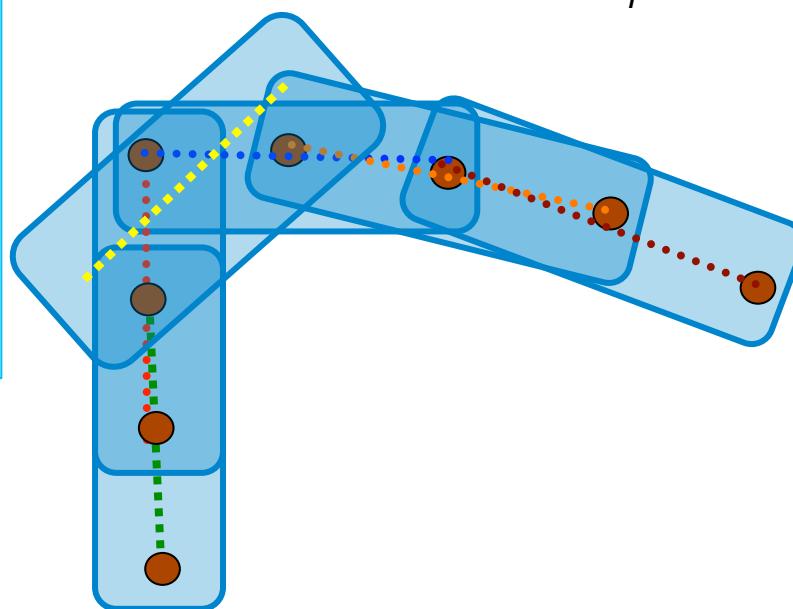


Line extraction | 2. line-regression

- “Sliding window” of size N_f points
- Fit line-segment to all points in each window

Line-Regression

- Initialize sliding window size N_f
- Fit a line to every N_f consecutive points (i.e. in each window)
- Merge overlapping line segments + re-compute line parameters for each segment

 $N_f = 3$ 

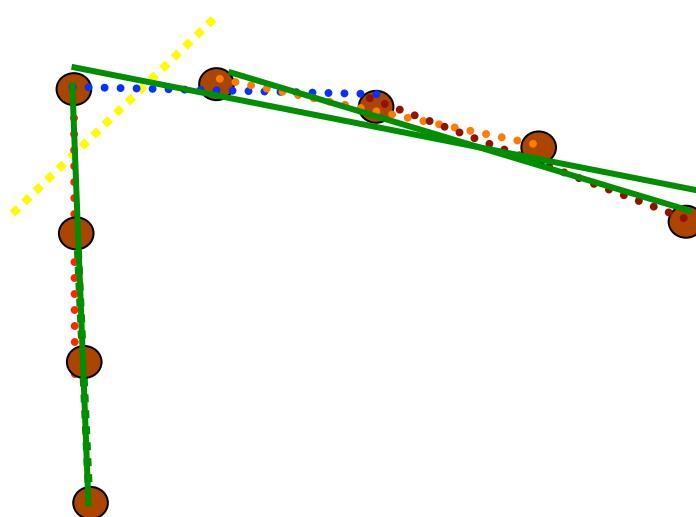
Line extraction | 2. line-regression

- “Sliding window” of size N_f points
- Fit line-segment to all points in each window

Line-Regression

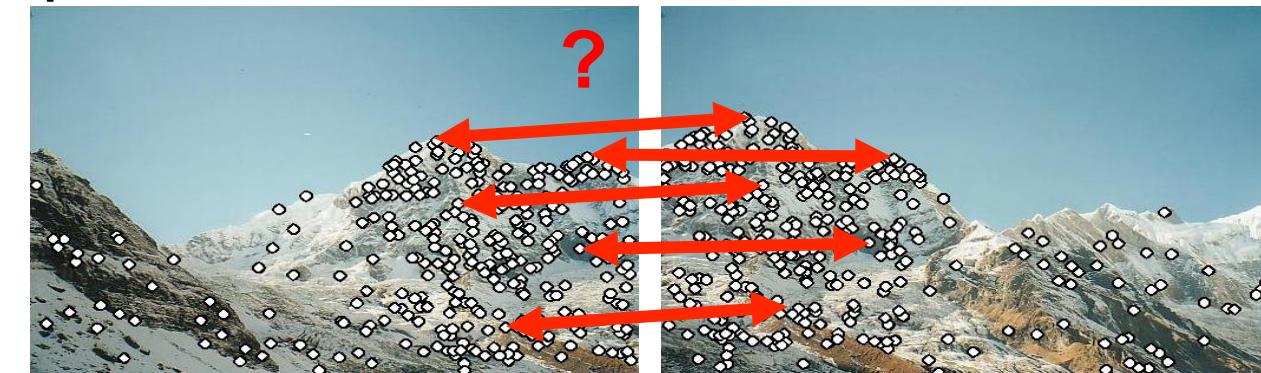
- Initialize sliding window size N_f
- Fit a line to every N_f consecutive points (i.e. in each window)
- Merge overlapping line segments + re-compute line parameters for each segment

$N_f = 3$

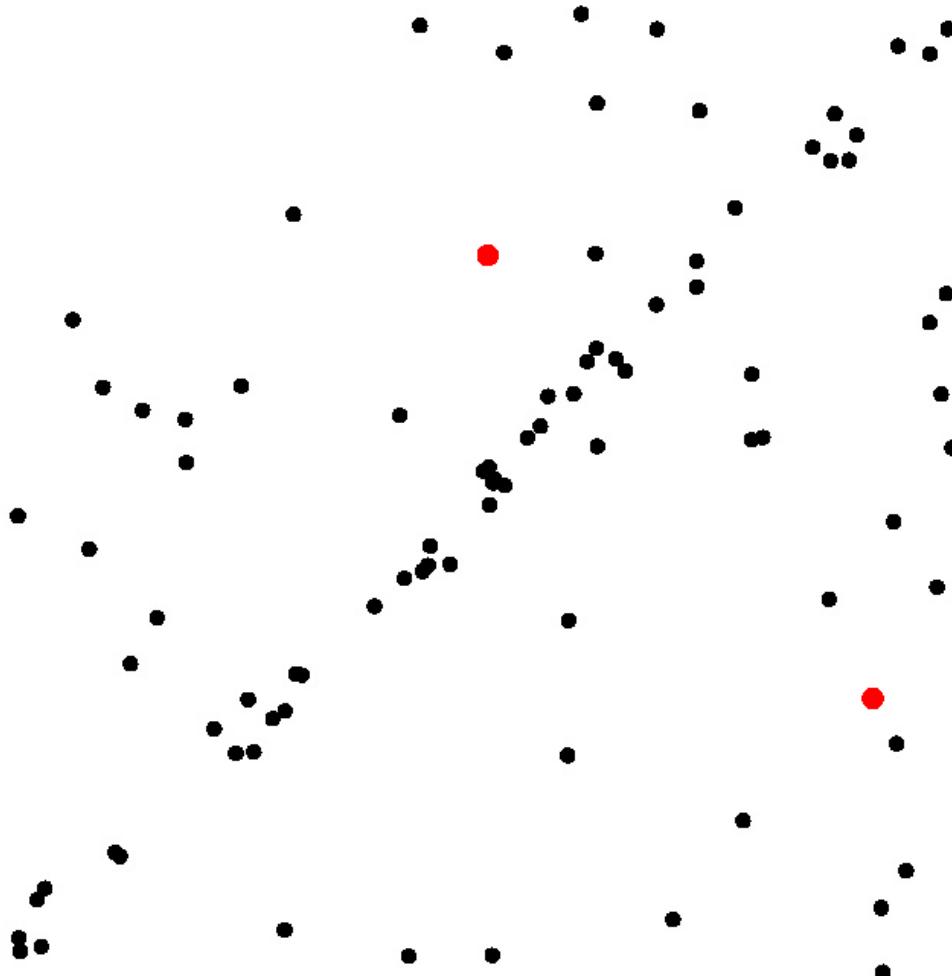


Line extraction | 3. RANSAC

- **RANSAC** = **RAN**dom **S**Ample **C**onsensus.
- A generic & robust fitting algorithm of models **in the presence of outliers** (i.e. points which do not satisfy a model)
- Can be applied in general to any problem, where the goal is to **identify the inliers which satisfy a predefined model**.
- Typical applications in robotics are:
line extraction from 2D range data, plane extraction from 3D data, feature matching, structure from motion, camera calibration, homography estimation, etc.
- RANSAC is **iterative** and **non-deterministic** \Rightarrow the probability to find a set free of outliers increases as more iterations are used
- Drawback: a non-deterministic method, results are different between runs.

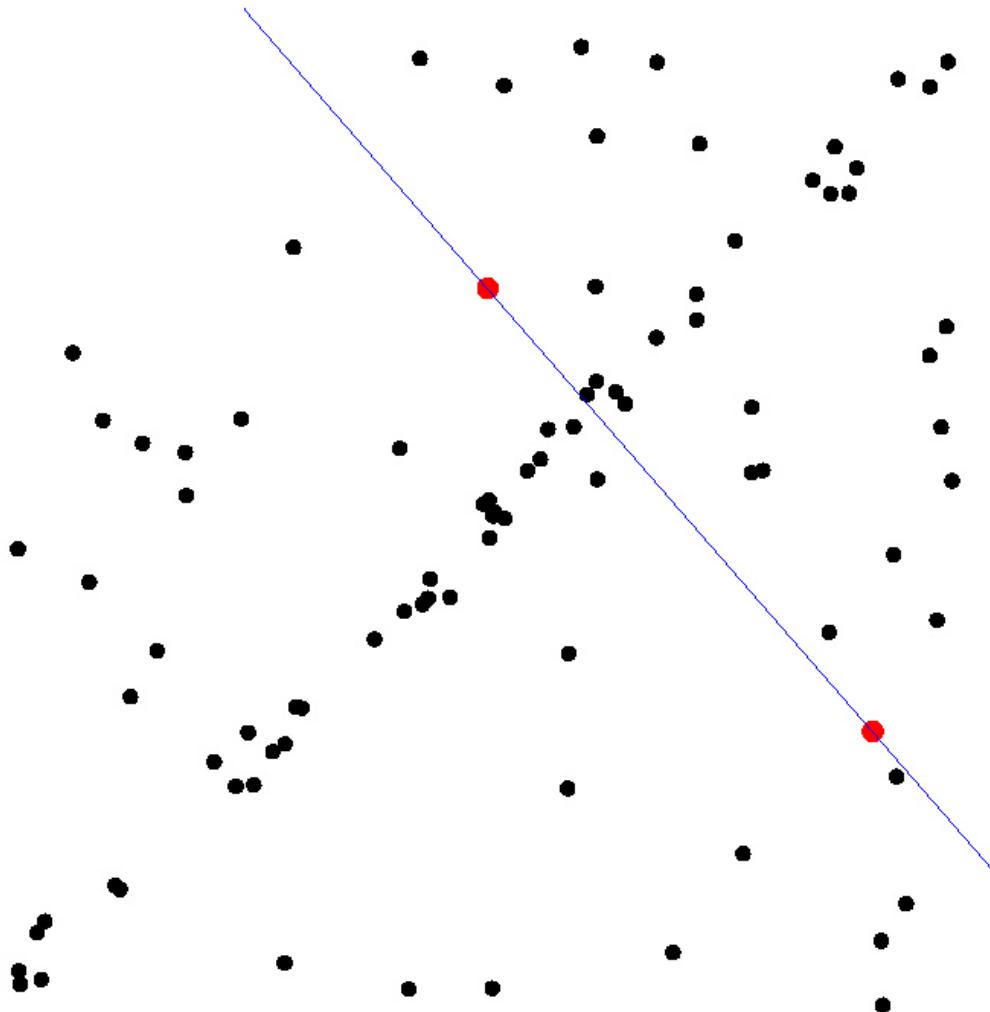


Line extraction | 3. RANSAC



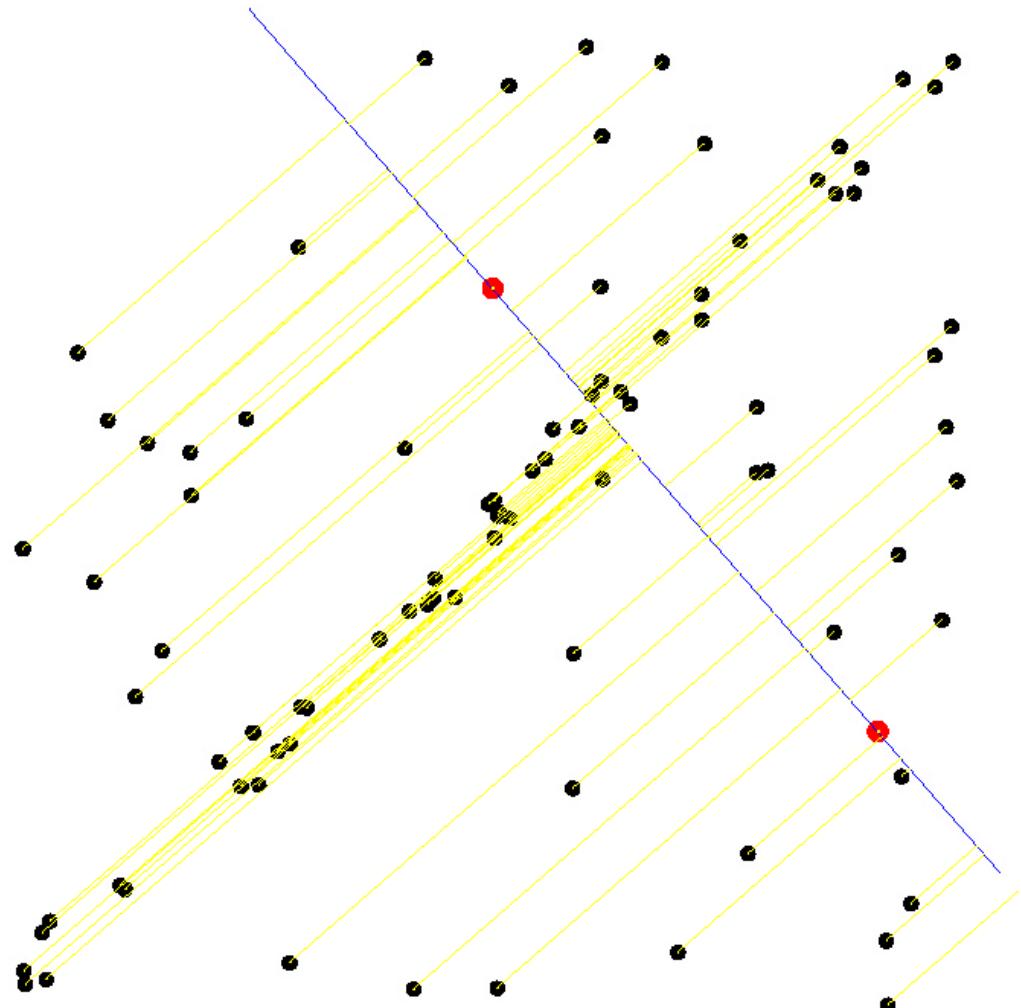
- Select sample of 2 points at random

Line extraction | 3. RANSAC



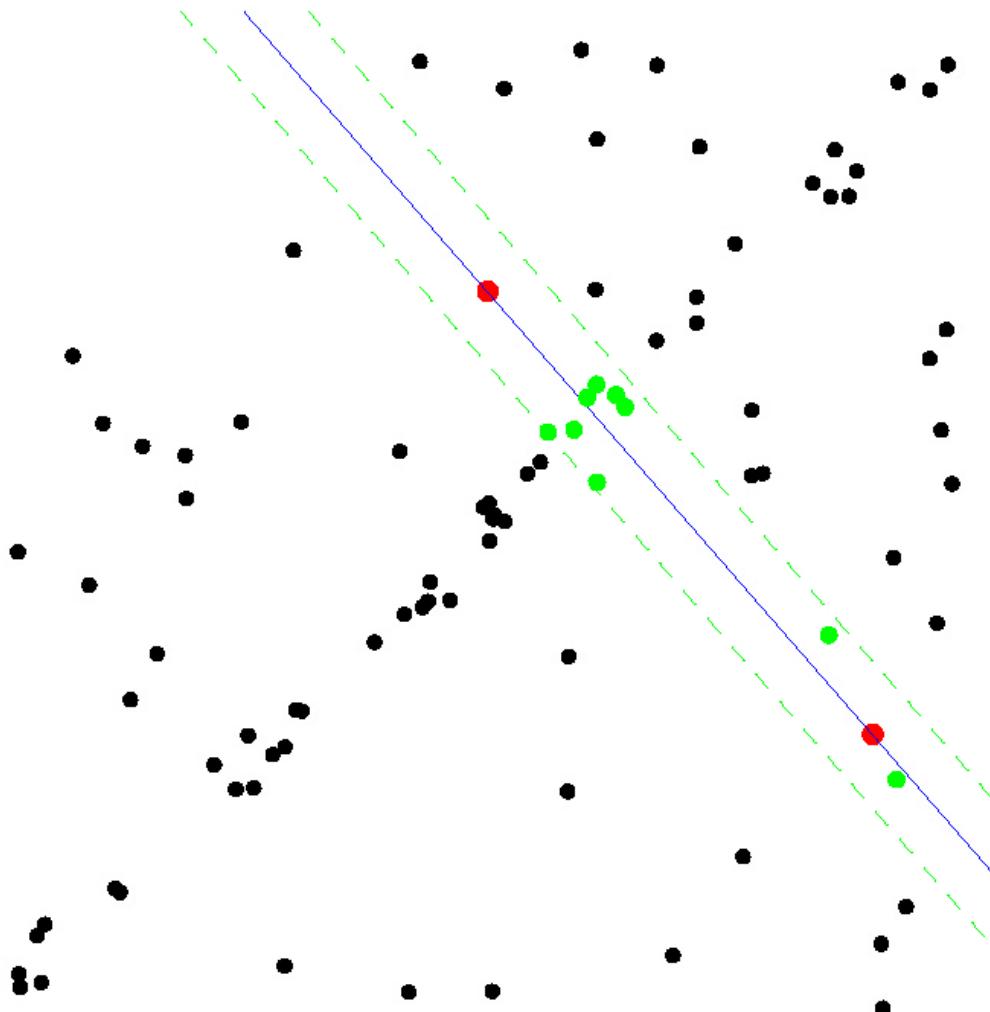
- Select sample of 2 points at random
- **Calculate model parameters that fit the data in the sample**

Line extraction | 3. RANSAC



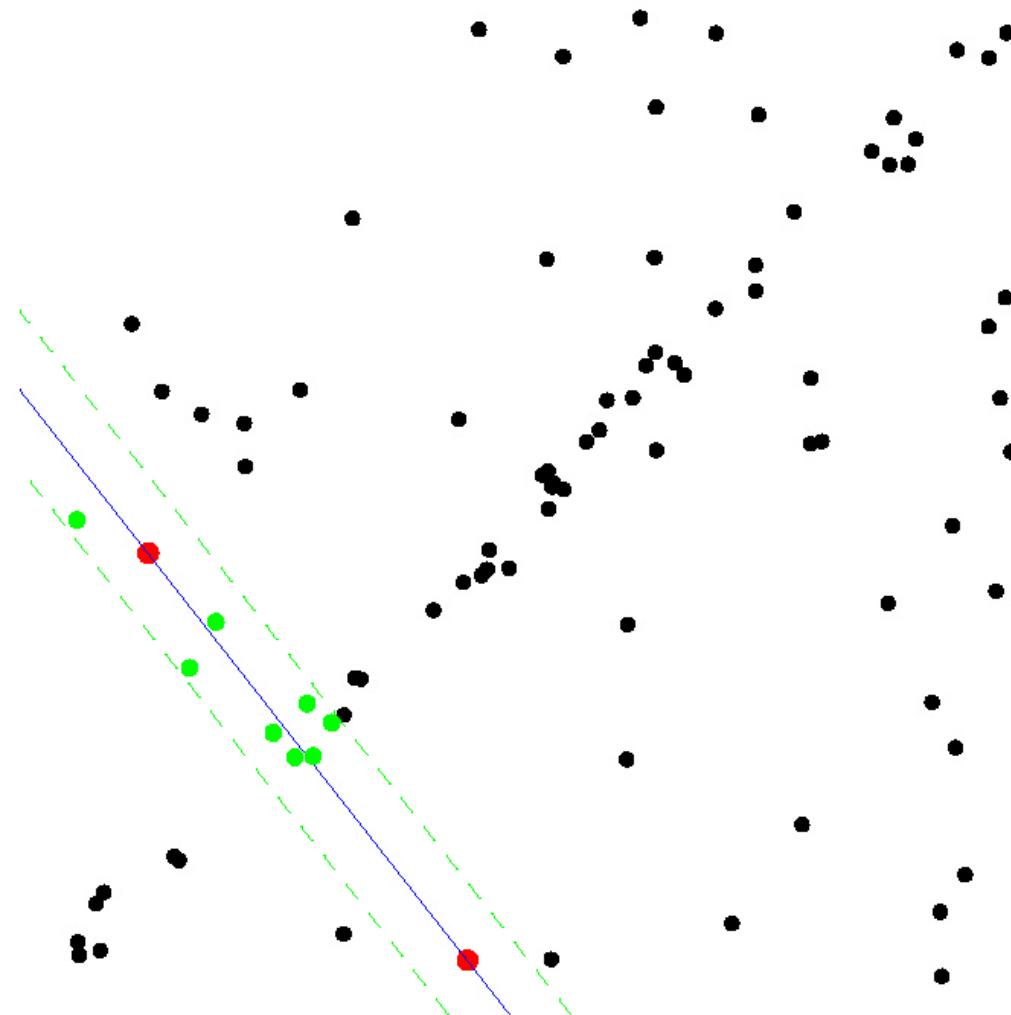
- Select sample of 2 points at random
- Calculate model parameters that fit the data in the sample
- **Calculate error function for each data point**

Line extraction | 3. RANSAC



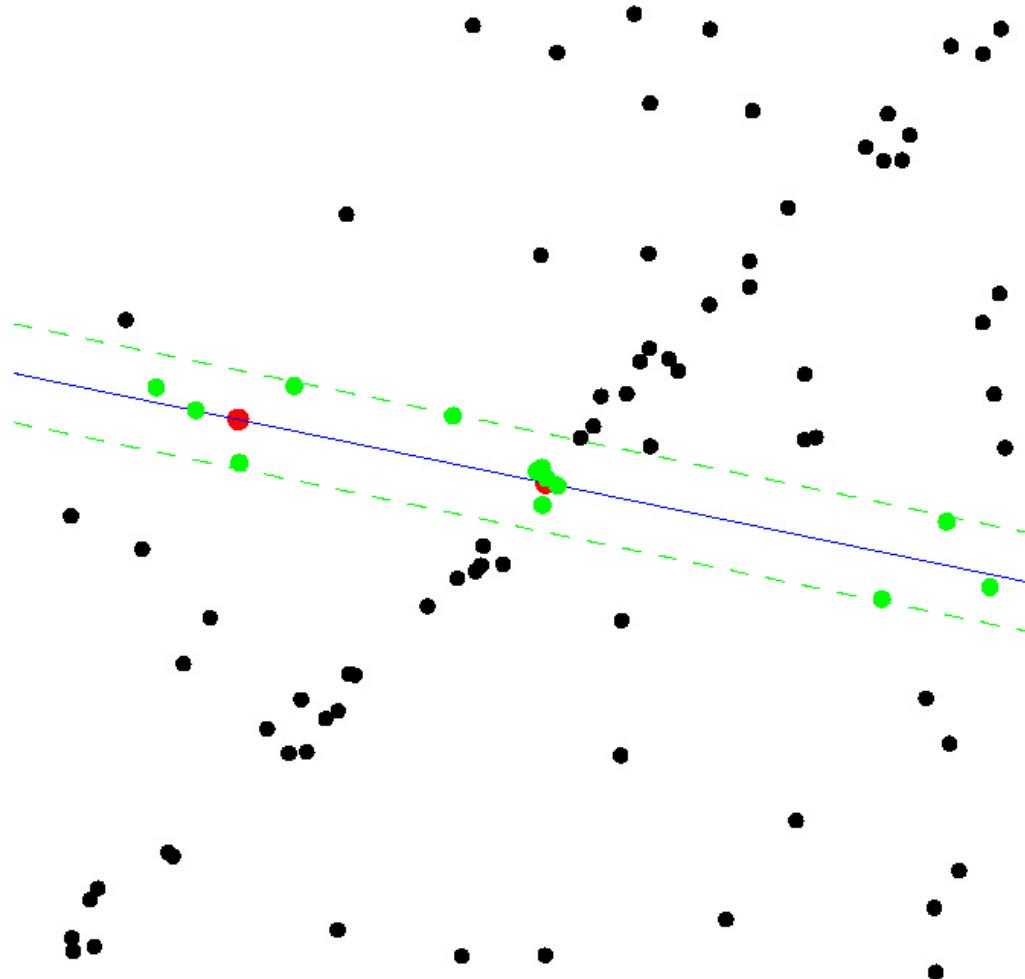
- Select sample of 2 points at random
- Calculate model parameters that fit the data in the sample
- Calculate error function for each data point
- **Select data that supports current hypothesis**

Line extraction | 3. RANSAC



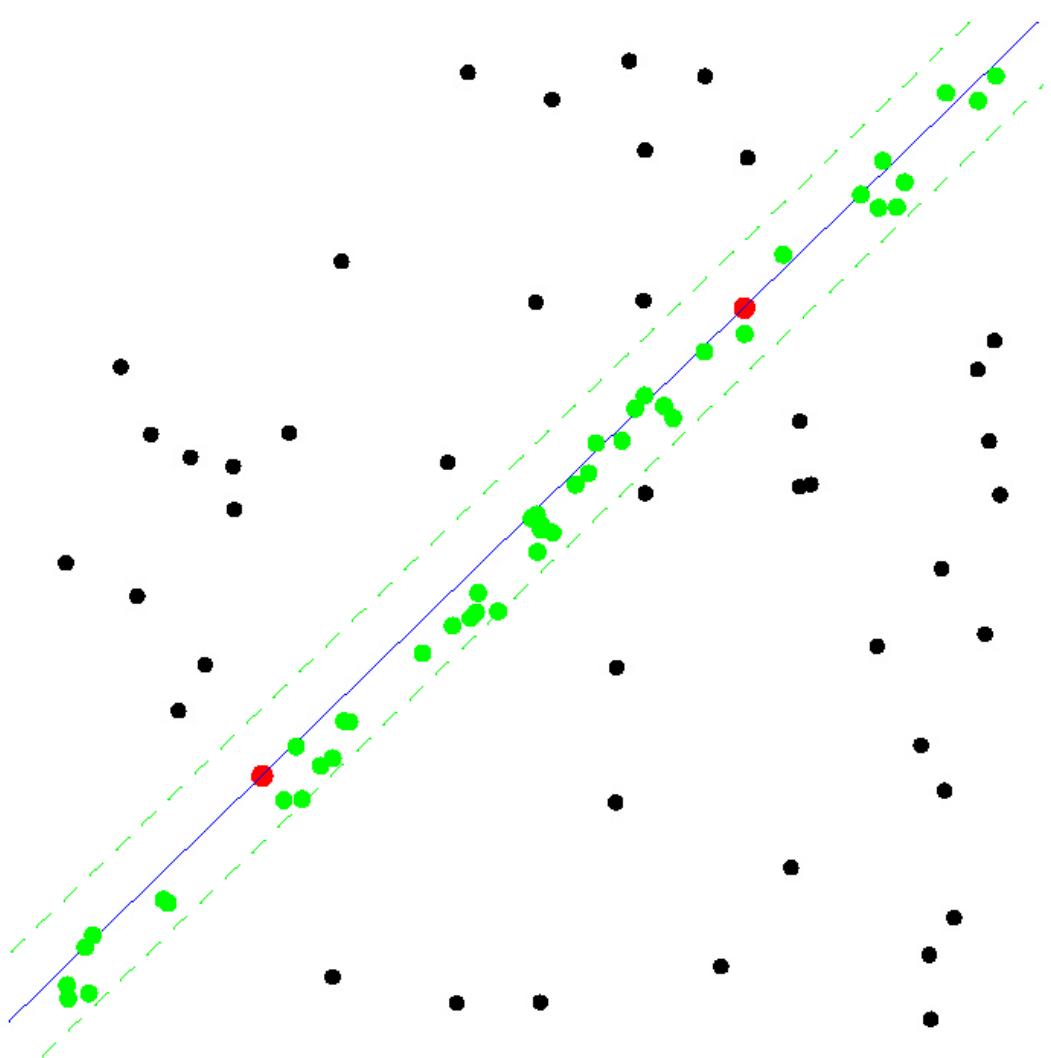
- Select sample of 2 points at random
- Calculate model parameters that fit the data in the sample
- Calculate error function for each data point
- Select data that supports current hypothesis
- **Repeat sampling**

Line extraction | 3. RANSAC



- Select sample of 2 points at random
- Calculate model parameters that fit the data in the sample
- Calculate error function for each data point
- Select data that supports current hypothesis
- **Repeat sampling**

Line extraction | 3. RANSAC



- Select sample of 2 points at random
- Calculate model parameters that fit the data in the sample
- Calculate error function for each data point
- Select data that supports current hypothesis
- **Repeat sampling**

Set with the maximum number of inliers obtained after k iterations

Line extraction | 3. RANSAC

How many iterations does RANSAC need? $k = ?$

- Ideally: check all possible combinations of **2** points in a dataset of **N** points.
- Number of all pairwise combinations: $N(N-1)/2$
⇒ computationally infeasible if **N** is too large.

example:

10'000 points to fit a line through ⇒ need to check all $10'000 \times 9'999/2 = 50$ million combinations!

- Do we really need to check all combinations or can we stop after some iterations?

Checking a **subset** of combinations is enough if we have a **rough estimate of the percentage of inliers** in our dataset

- This can be done in a probabilistic way

Line extraction | 3. RANSAC

How many iterations does RANSAC need? $k = ?$

- $N :=$ tot. no. data points
 $w :=$ number of inliers / N
 $\Rightarrow w$: fraction of inliers in the dataset $\Rightarrow w = P(\text{selecting an inlier-point out of the dataset})$
- Let $p := P(\text{selecting a minimal set of points free of outliers})$
- Assumption: the 2 points necessary to estimate a line are selected independently
 - $\Rightarrow w^2 = P(\text{both selected points are inliers})$
 - $\Rightarrow 1-w^2 = P(\text{at least one of these two points is an outlier})$
- Let $k :=$ no. RANSAC iterations executed so far
 - $\Rightarrow (1-w^2)^k = P(\text{RANSAC never selects two points that are both inliers})$
 - $\Rightarrow 1-p = (1-w^2)^k$ and therefore :
$$k = \frac{\log(1-p)}{\log(1-w^2)}$$

Line extraction | 3. RANSAC

How many iterations does RANSAC need? $k = ?$

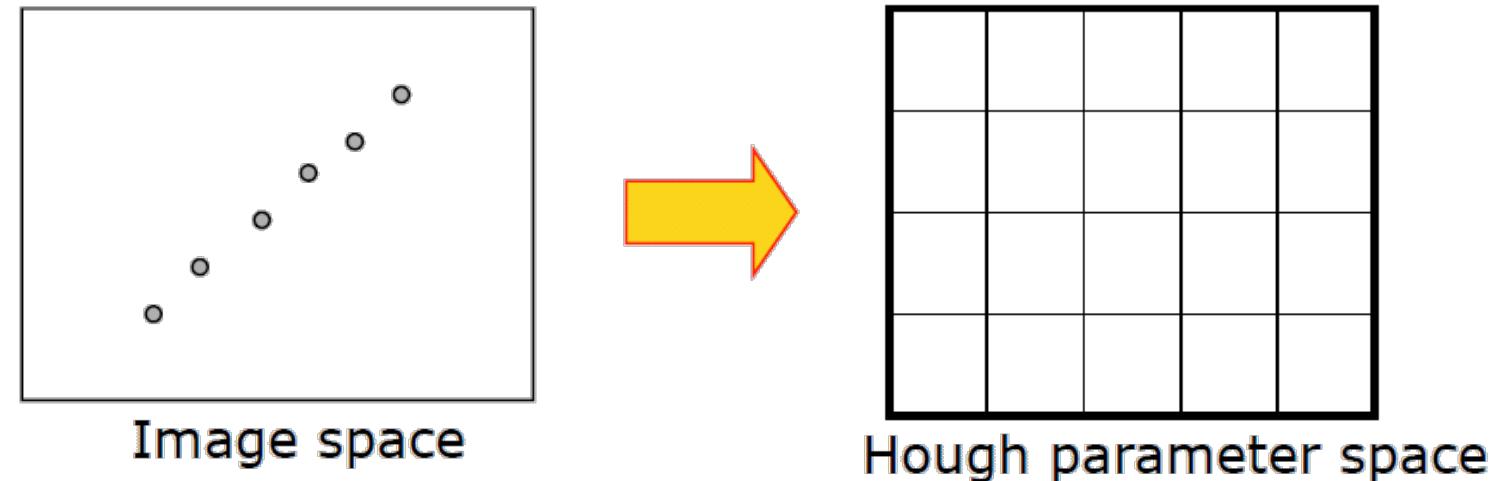
- The number of iterations k is:
$$k = \frac{\log(1 - p)}{\log(1 - w^2)}$$

⇒ knowing the fraction of inliers w , after k RANSAC iterations we will have a probability p of finding a set of points free of outliers

- Example:** if we want a probability of success $p=99\%$ and we know that $w=50\%$ ⇒ $k=16$ iterations
 - these are dramatically fewer than the number of all possible combinations!
- Notice: the number of points does not influence the estimated number of iterations, only w does!
- In practice we need only a rough estimate of w .
More advanced variants of RANSAC estimate the fraction of inliers and adaptively change it on every iteration

Line extraction | 4. Hough-Transform

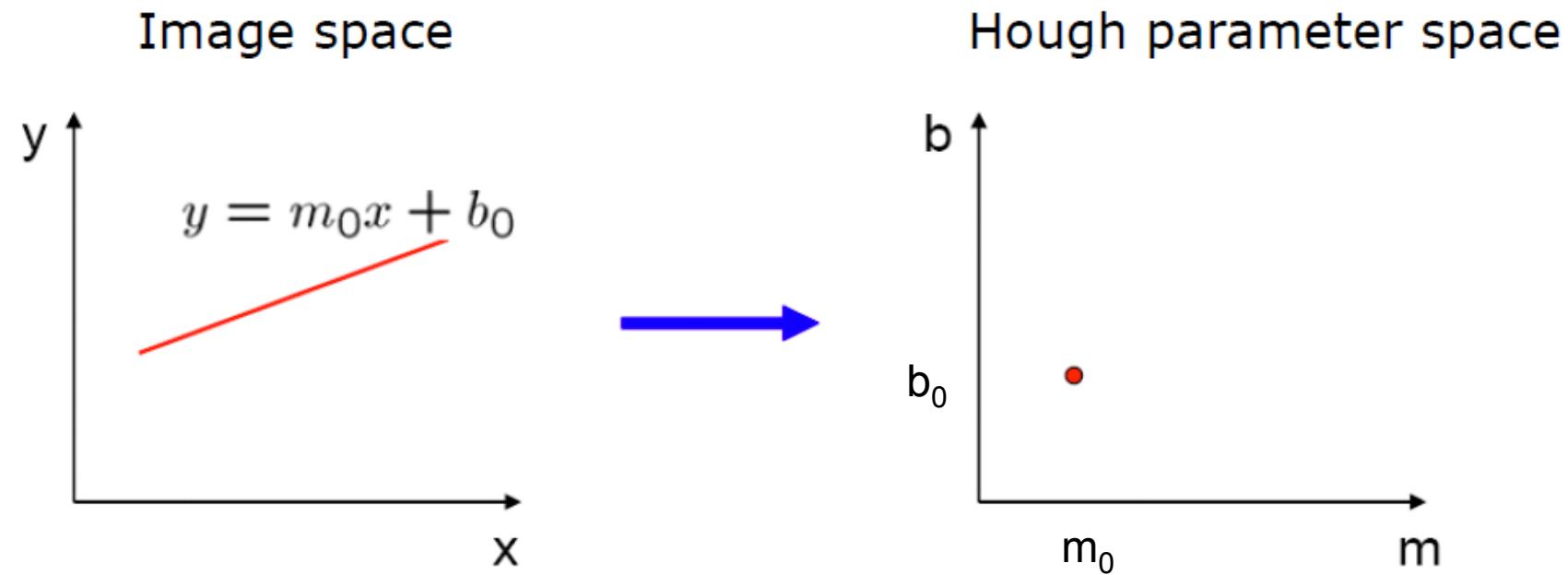
- Points vote for plausible line parameters
- Hough-Transform: maps image-space into Hough-space
- **Hough-space**: voting accumulator, parametrized w.r.t. line characteristics



1. P. Hough, [Machine Analysis of Bubble Chamber Pictures](#), Proc. Int. Conf. High Energy Accelerators and Instrumentation, 1959
2. J. Richard, O. Duda, P.E. Hart (April 1971). "[Use of the Hough Transformation to Detect Lines and Curves in Pictures](#)". Artificial Intelligence Center (SRI International)

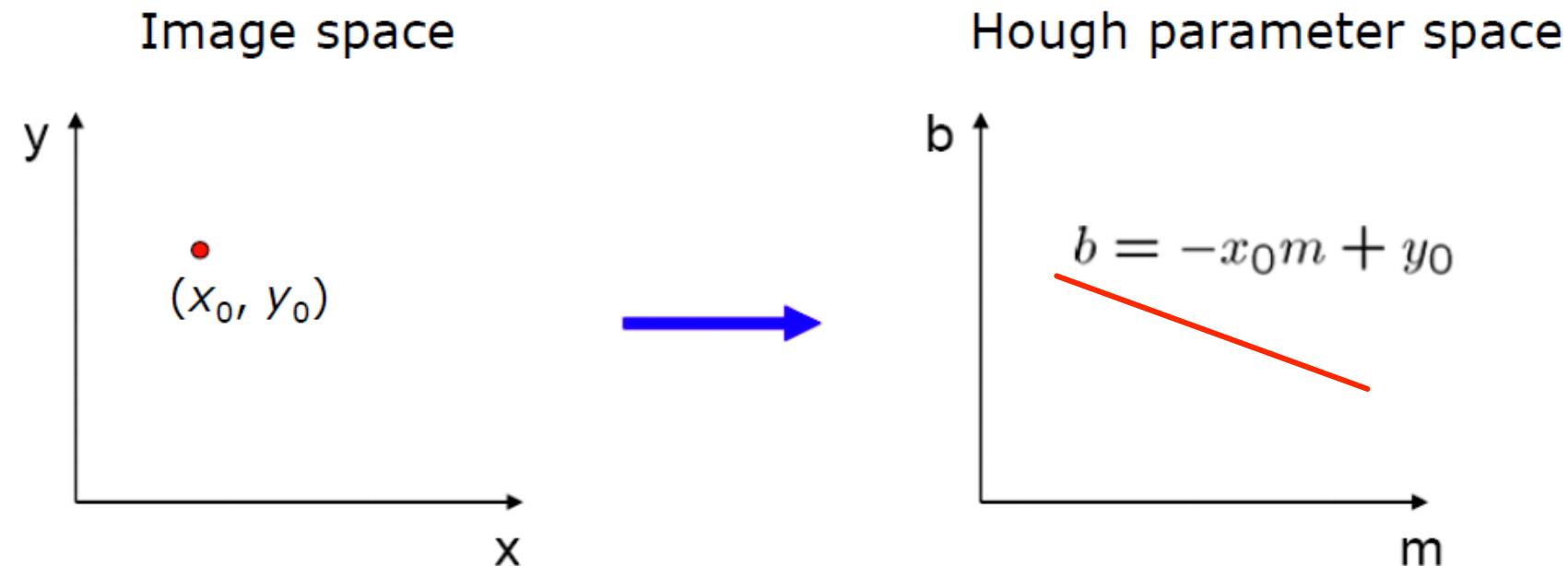
Line extraction | 4. Hough-Transform

- A line in the image corresponds to a point in Hough space



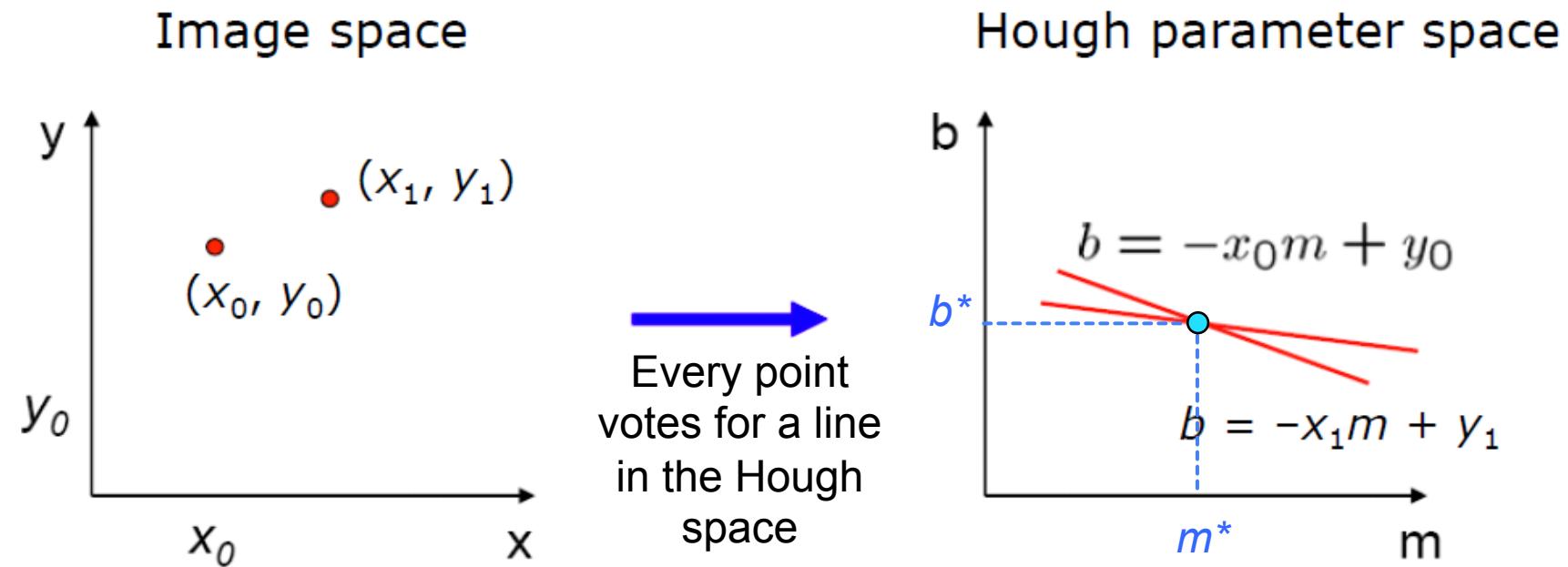
Line extraction | 4. Hough-Transform

- What does a point (x_0, y_0) in the image space map to in the Hough space?

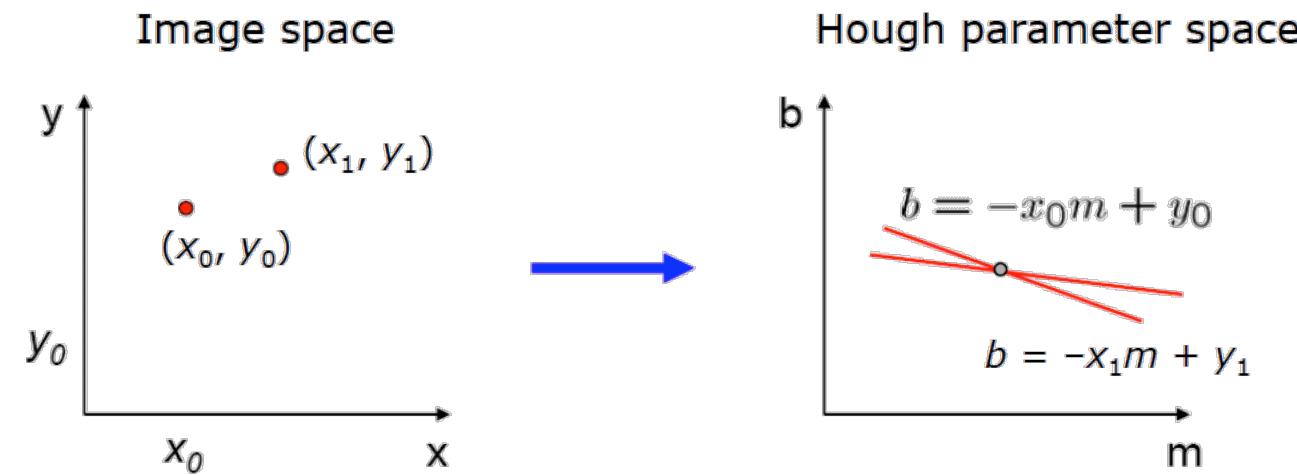


Line extraction | 4. Hough-Transform

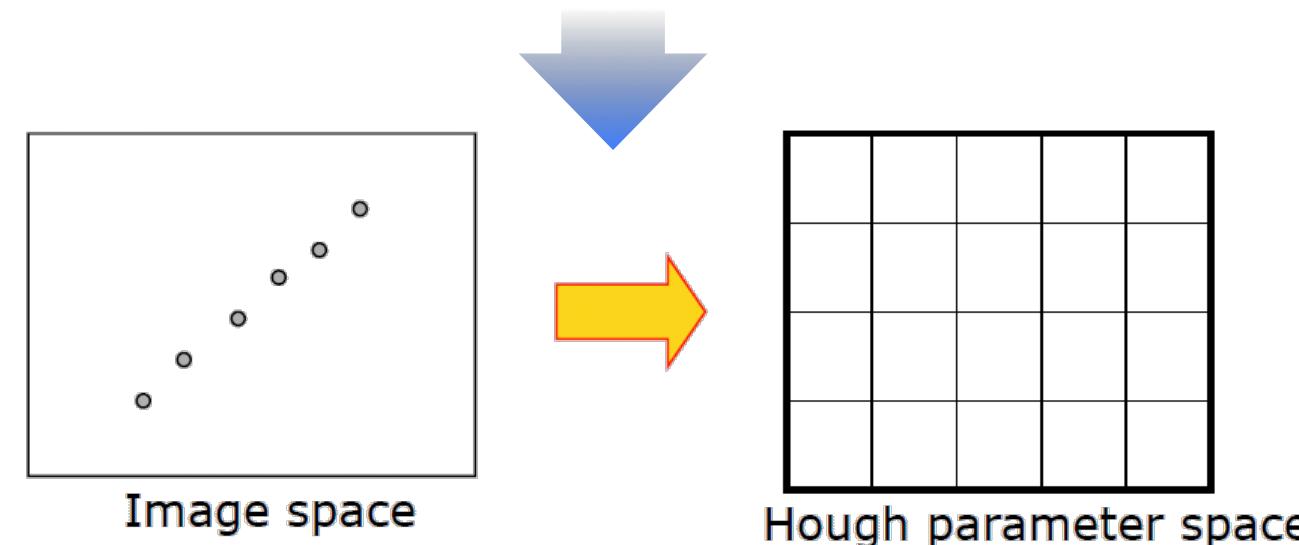
- Where is the line that contains both (x_0, y_0) and (x_1, y_1) ?
 - It is the intersection of the lines $b = -x_0m + y_0$ and $b = -x_1m + y_1$



Line extraction | 4. Hough-Transform



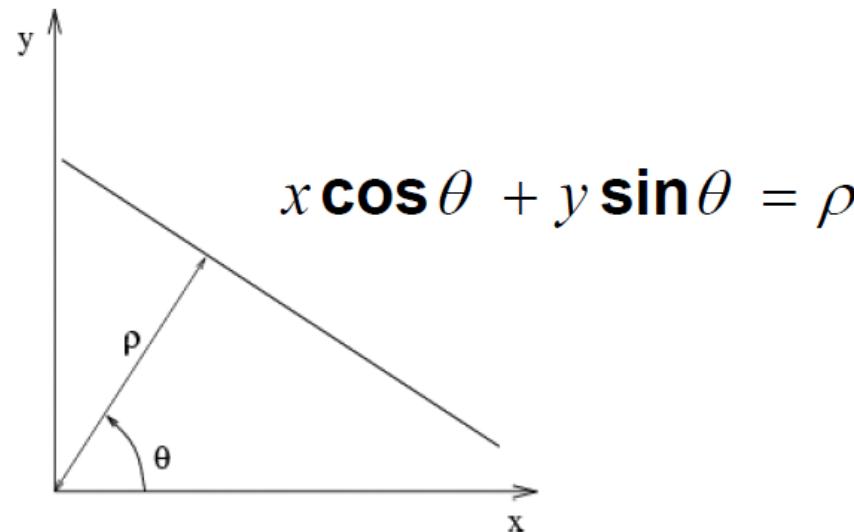
- Each point in image space, votes for line-parameters in Hough parameter space



Line extraction | 4. Hough-Transform

- Problems with the (m, b) space:
 - Unbounded parameter domain
 - How to represent lines aligned with the axes?

- Alternative: **polar representation**

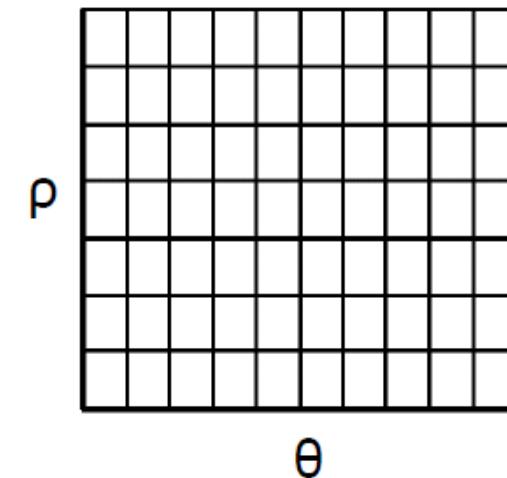


Each point in image space will map to a **sinusoid** in the (ρ, θ) parameter space

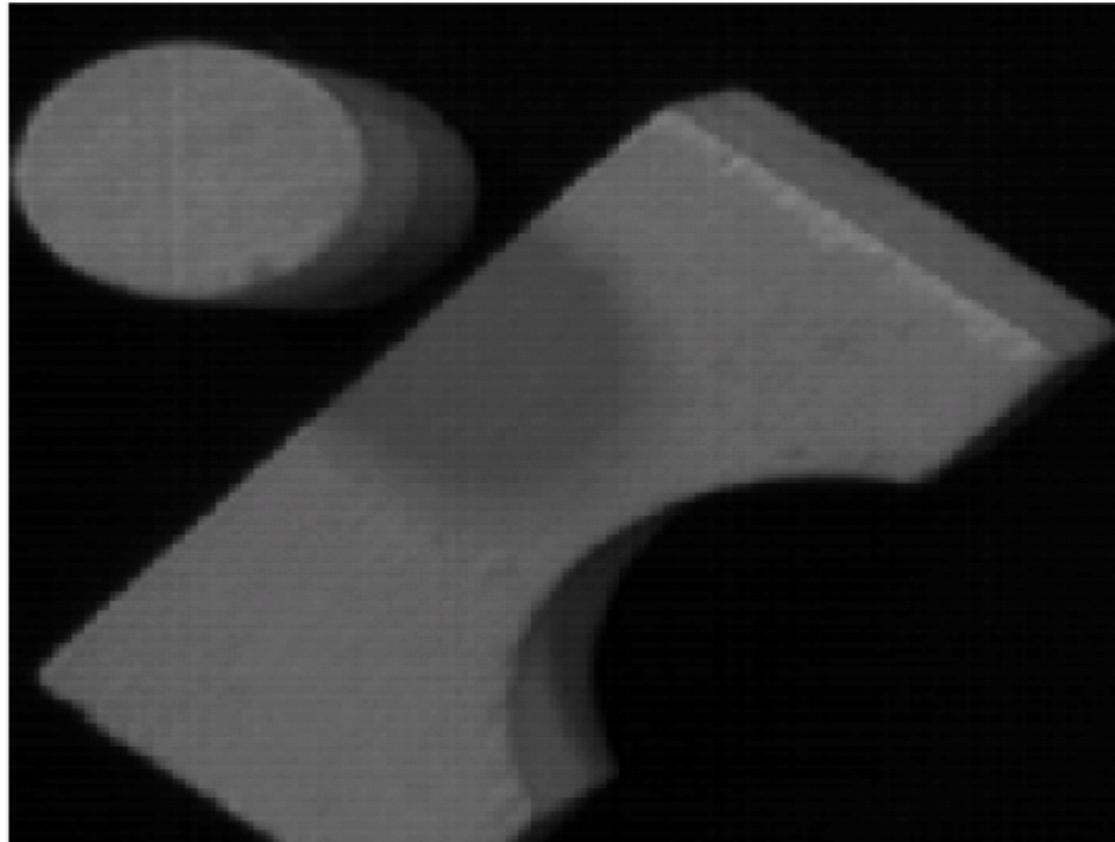
Line extraction | 4. Hough-Transform

1. Initialize accumulator H to all zeros
2. **for** each edge point (x,y) in the image
 - for** all θ in $[0,180]$
 - Compute $\rho = x \cos \theta + y \sin \theta$
 - $H(\theta, \rho) = H(\theta, \rho) + 1$
 - end**
3. Find the values of (θ, ρ) where $H(\theta, \rho)$ is a local maximum
4. The detected line in the image is given by: $\rho = x \cos \theta + y \sin \theta$

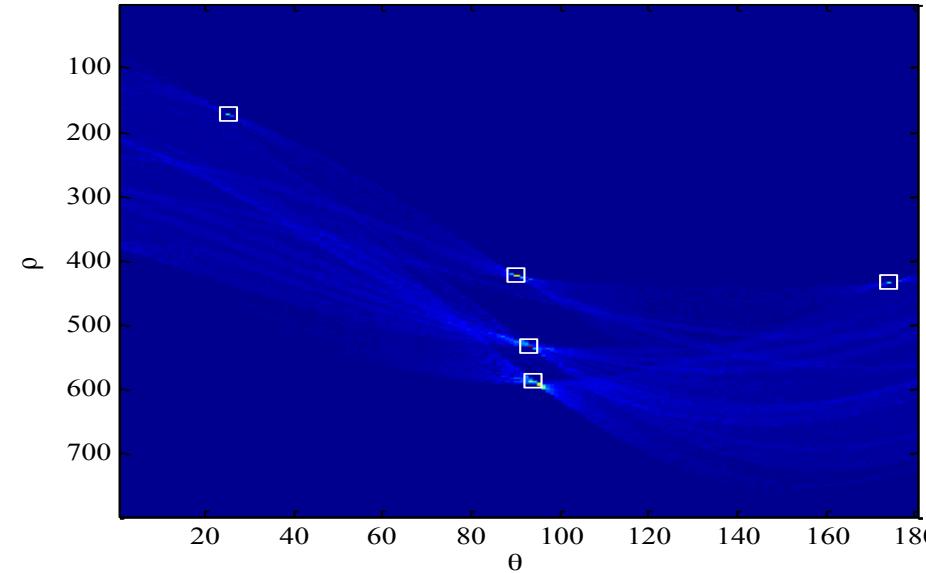
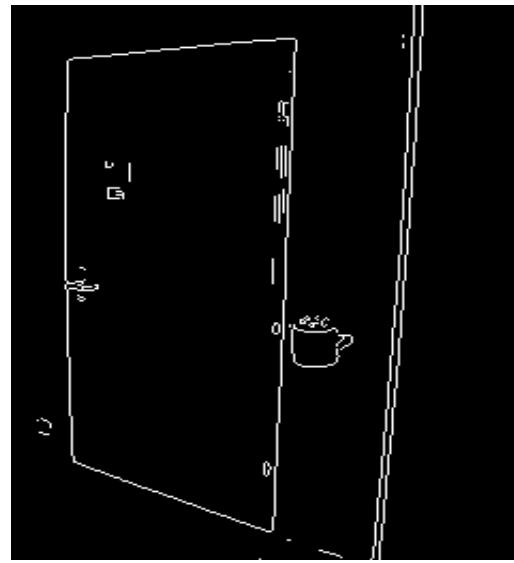
H : accumulator array (votes)



Line extraction | 4. Hough-Transform: examples



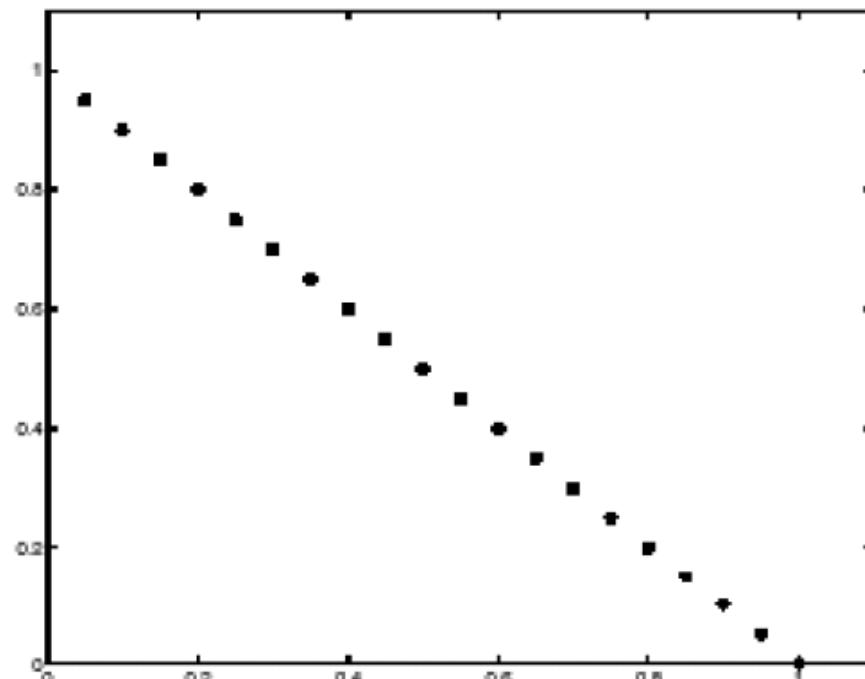
Line extraction | 4. Hough-Transform: examples



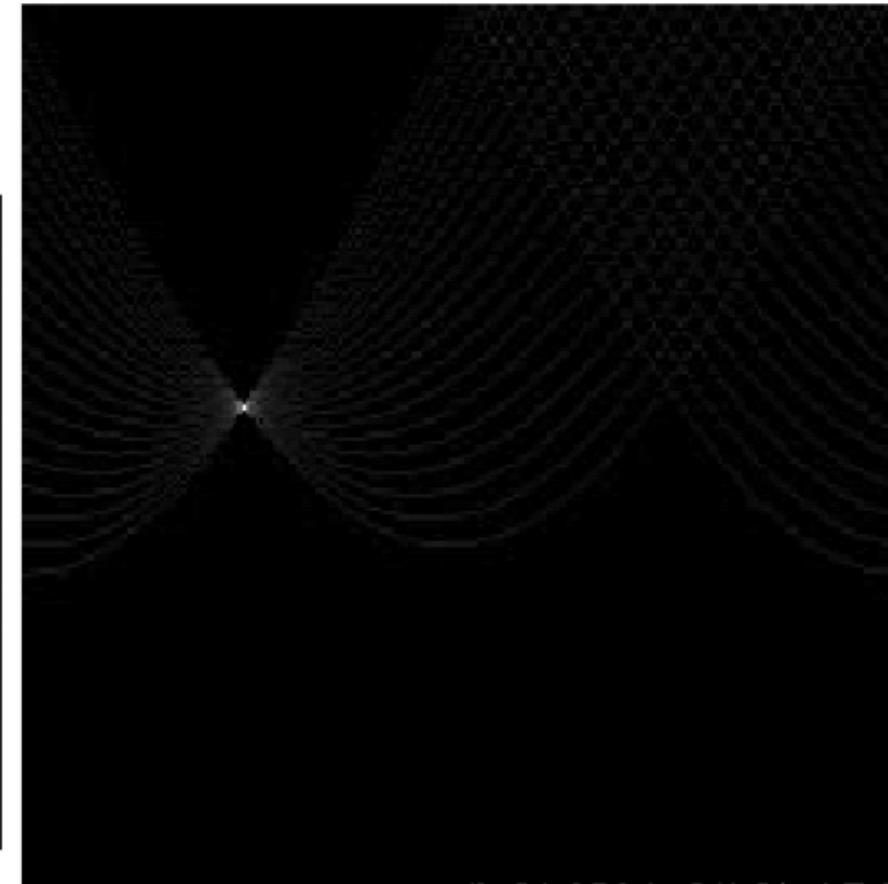
Hough Transform



Line extraction | 4. Hough-Transform: examples



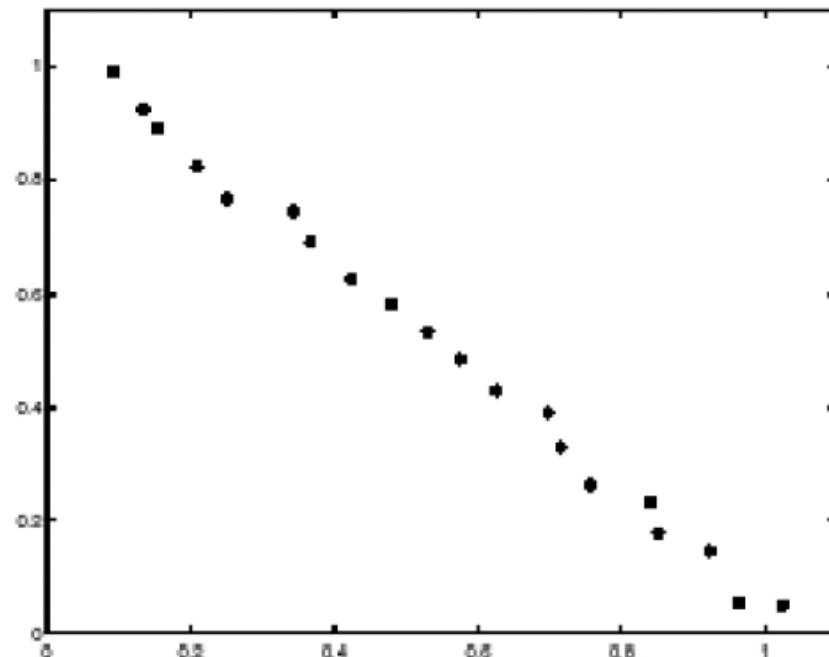
features



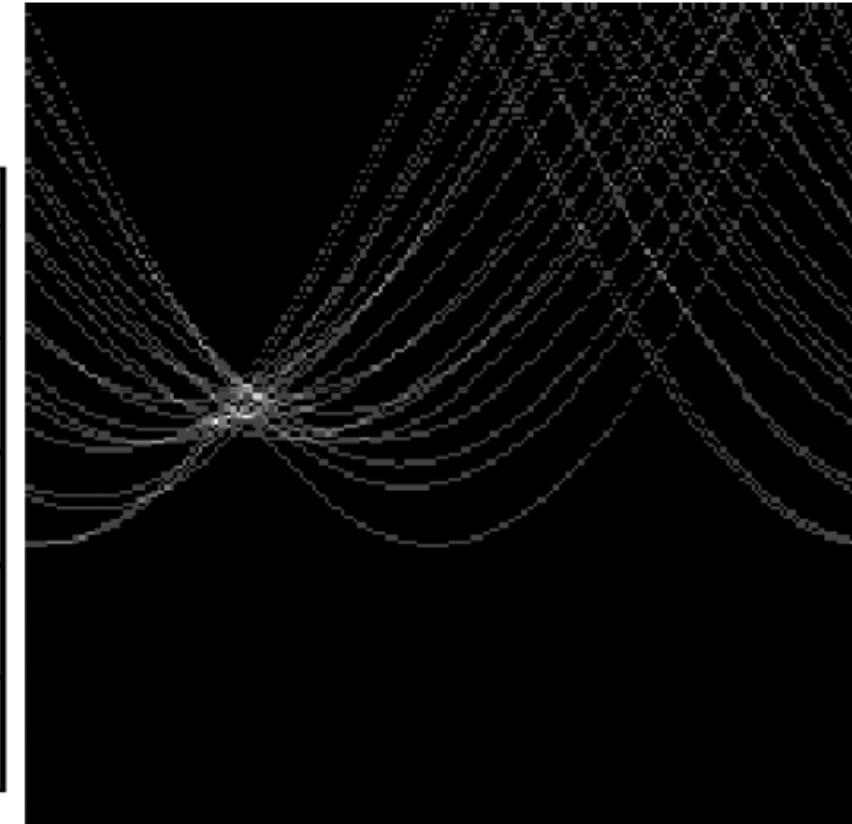
votes

Line extraction | 4. Hough-Transform: examples

Effect of Noise:



features



votes

- Peak gets fuzzy and hard to locate

Line extraction | comparison

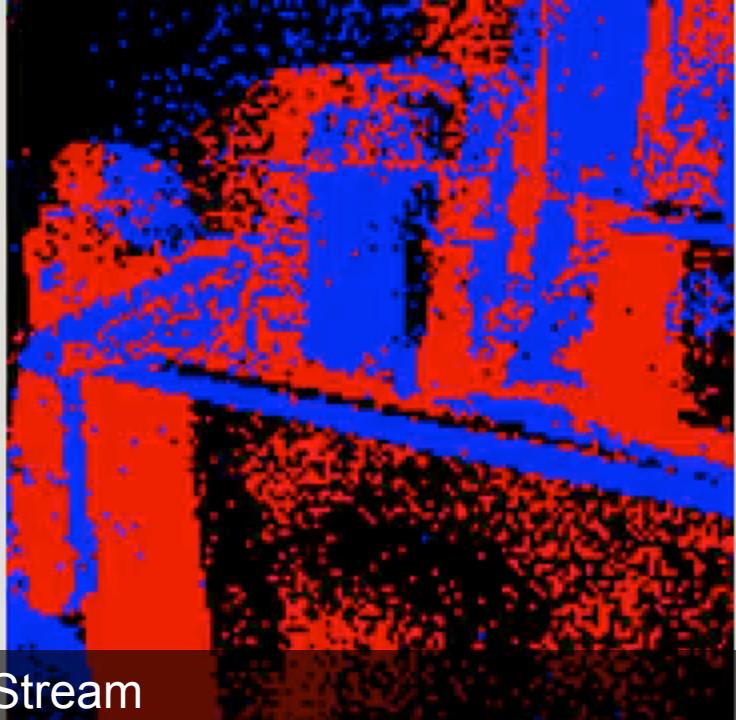
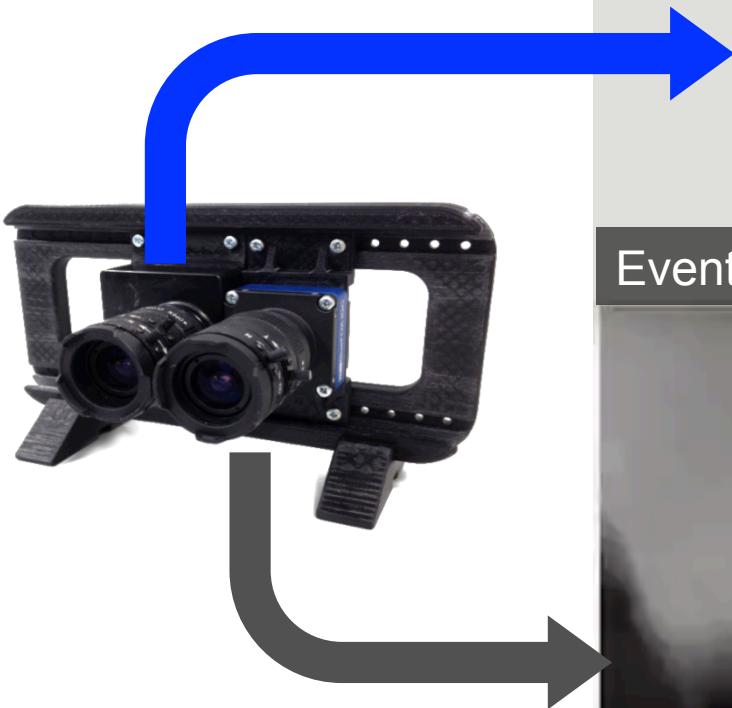
- Split-and-merge, Incremental and Line-Regression: fastest – best applied on laser scans
- Deterministic & make use of the sequential ordering of raw scan points (: points captured according to the rotation direction of the laser beam)
- If applied on randomly captured points only last 3 algorithms would segment all lines.
- RANSAC, Hough-Transform and EM produce greater precision \Rightarrow more robust to outliers

N : no. points considered
 N_f : no. points in window
 S : no. line-segments to be found
 k : no. iterations
 N_C, N_R : no columns, rows of the accumulator array

	Complexity	Speed (Hz)	False positives	Precision
Split-and-Merge	$N \log N$	1500	10%	+++
Incremental	$S N$	600	6%	+++
Line-Regression	$N N_f$	400	10%	+++
RANSAC	$S N k$	30	30%	++++
Hough-Transform	$S N N_C + S N_R N_C$	10	30%	++++
Expectation Maximization	$S N_1 N_2 N$	1	50%	++++

Comparison by
[Nguyen et al.
IROS 2005]

Detecting lines in the DVS stream



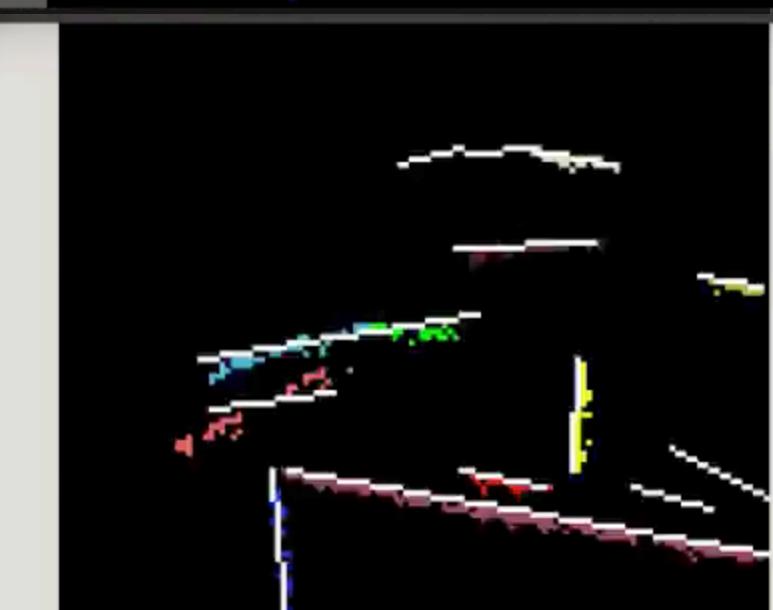
Event Stream



Visible-light image



Clusters of Events



Lines fitted on Events