Robotics 2

AdaBoost for People and Place Detection

Kai Arras, Cyrill Stachniss,

Maren Bennewitz, Wolfram Burgard

v.1.1, Kai Arras, Jan 12, including material by Luciano Spinello and Oscar Martinez Mozos

Chapter Contents

- Machine Learning: A Survey
- Classification
- AdaBoost
- People Detection with Boosted Features
- Place Recognition with Boosted Features

Machine Learning: Survey

What is Machine Learning?

- **Learning a model from data**
- Fundamentally different than **model-based approaches** where the model is derived from domain knowledge, e.g. physics, social science
- Often it is too **complex**, too **costly**, or **impossible** to model a process in “closed form” (e.g. financial market, consumer behavior in on-line store)
- Thus, we can **collect data** and hope to **extract the process or pattern** that explains the observed data
- Even if we are unable to describe the complete process, an **approximate model** may be enough

Machine Learning: Survey

Machine Learning Taxonomy:

- **Supervised Learning:** Inferring a function from labelled training data
  - Examples: Classification, Regression
- **Unsupervised Learning:** Try to find hidden structures in unlabeled data
  - Examples: Clustering, Outlier Detection
- **Semi-supervised Learning:** Learn a function from both, labelled and unlabelled data
- **Reinforcement Learning:** Learn how to act guided by feedback (rewards) from the world

Machine Learning: Survey

Machine Learning Examples:

- **Classification**
  - Support Vector Machines (SVM), naive Bayes, LDA, Decision trees, k-nearest neighbor, ANNs, AdaBoost
- **Regression**
  - Gaussian Processes, Least Squares Estimation, Gauss-Newton
- **Clustering**
  - GMMs, Hierarchical clustering, k-means
- **Reinforcement Learning**
  - Q-Learning

Machine Learning: Survey

Machine Learning in Robotics Examples:

- **Perception**: people/object/speech recognition from sensory data, learning of dynamic objects
- **Modeling:** human behavior modeling and analysis
- **Planning**: on learned cost maps, e.g. for human-aware coverage
- **Action** (learning motions by imitating people, e.g. ping-pong playing)

Machine Learning has become a very **popular tool** for many robotics tasks

Can make systems **adaptive** to changing environments

Chapter Contents

- Machine Learning: A Survey
- **Classification**
- AdaBoost
- People Detection with Boosted Features
- Place Recognition with Boosted Features

Classification

- Classification algorithms are **supervised algorithms** to predict **categorical labels**
- Differs from **regression** which is a supervised technique to predict **real-valued labels**

Formal problem statement:

- **Produce a function that maps**

  $C : \mathcal{X} \to \mathcal{Y}$

- **Given a training set**

  $\{(\mathbf{x_1}, y_1), \ldots, (\mathbf{x_n}, y_n)\}$

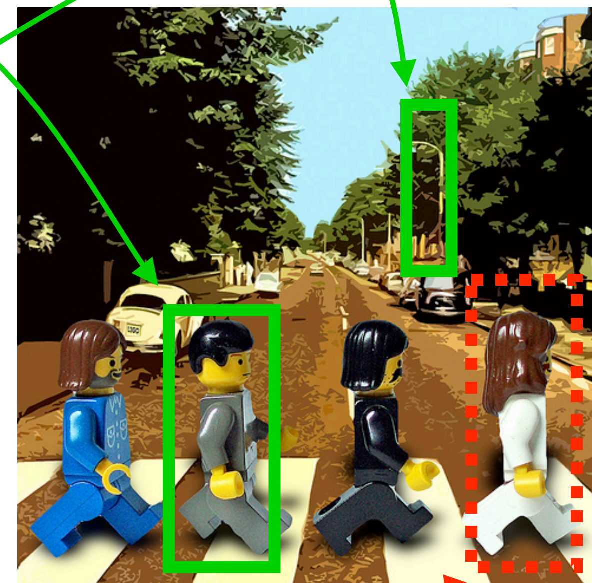
  $y \in \mathcal{Y}$ label

  $\mathbf{x} \in \mathcal{X}$ training sample

Classification

Error types

		True value	
		T	*N*
Predicted value of the classifier	*T'*	**True Positive**	**False Positive**
	N'	**False Negative**	**True Negative**



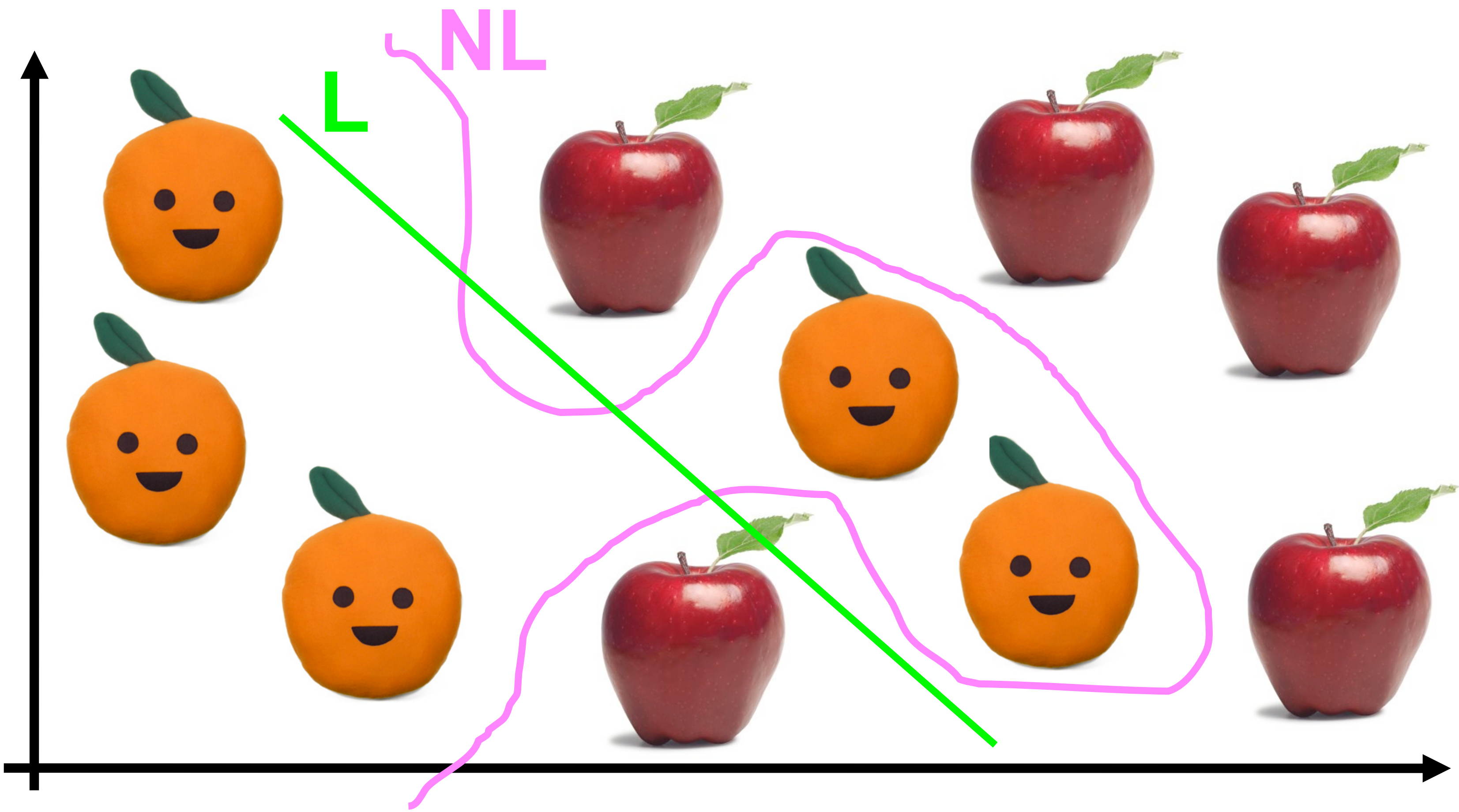
Detected
Not Detected

- **Precision** = TP / (TP + FP)
- **Recall** = TP / (TP + FN)

Many more measures...

Classification

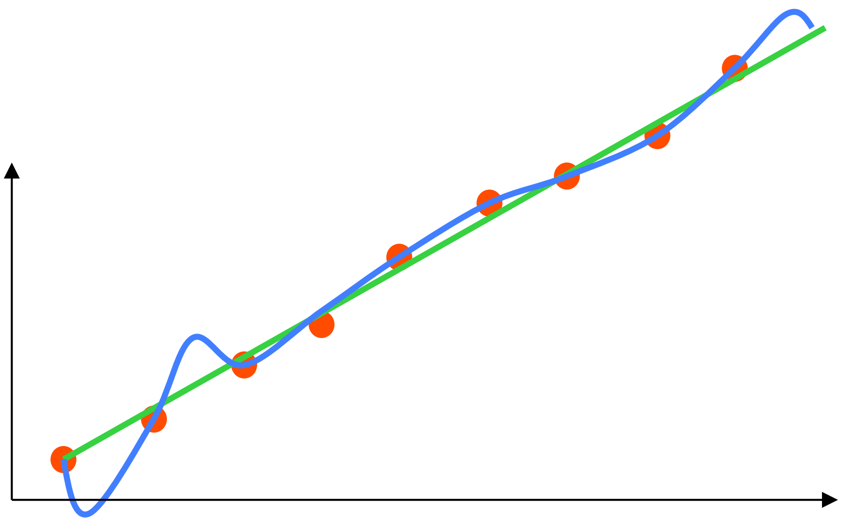
Linear vs. **Non-Linear Classifier, Margin**



Classification

Overfitting

- Overfitting occurs when a model begins to memorize the **training data** rather than learning the underlying relationship
- Occurs typically when fitting a statistical model with too many parameters
- Overfitted models explain training data perfectly but they **do not generalize!**
- There are techniques to avoid overfitting such as regularization or cross-validation



Chapter Contents

- Machine Learning: A Survey
- Classification
- **AdaBoost**
- People Detection with Boosted Features
- Place Recognition with Boosted Features

Boosting

- An **ensemble technique** (a.k.a. committee method)
- Supervised learning: given <samples *x*, labels *y*>

- Learns an accurate **strong classifier** by combining an ensemble of inaccurate "rules of thumb"
- **Inaccurate rule** $h(x_i)$: "weak" classifier, weak learner, basis classifier, feature
- **Accurate rule** $H(x_i)$: "strong" classifier, final classifier

- Other **ensemble techniques** exist: Bagging, Voting, Mixture of Experts, etc.

AdaBoost

- Most popular algorithm: **AdaBoost**
  [Freund et al. 95], [Schapire et al. 99]

- Given an ensemble of weak classifiers $h(x_i)$, the combined strong classifier $H(x_i)$ is obtained by a **weighted majority voting scheme**

$$f(x_i) = \sum_{t=1}^{T} \alpha_t h_t(x_i) \qquad H(x_i) = \operatorname{sgn}\left(f(x_i) \right)$$

- AdaBoost in Robotics:
  [Viola et al. 01], [Treptow et al. 04], [Martínez-Mozos et al. 05], [Rottmann et al. 05] , [Monteiro et al. 06] , [Arras et al. 07]

AdaBoost

Why is AdaBoost interesting?

1. It tells you what the **best "features"** are
2. What the **best thresholds** are, and
3. How to **combine them to a classifier**

- AdaBoost can be seen as a **principled feature selection strategy**

- Classifier design becomes **science**, not art

AdaBoost

- AdaBoost is a **non-linear classifier**
- Has **good generalization properties**: can be proven to maximize the margin
- Quite robust to **overfitting**
- Very **simple** to implement

- **Prerequisite:**
  weak classifier must be better than chance:
  error < 0.5 in a binary classification problem

AdaBoost

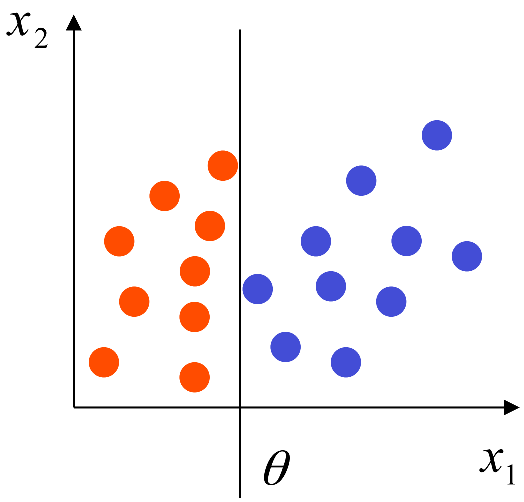
- **Possible Weak Classifiers:**
  - **Decision stump**:
    Single axis-parallel partition of space
  - **Decision tree**:
    Hierarchical partition of space
  - **Multi-layer perceptron**:
    General non-linear function approximators
  - **Support Vector Machines** (SVM):
    Linear classifier with RBF Kernel
- Trade-off between diversity among weak learners versus their accuracy. Can be complex, see literature
- Decision stumps are **a popular choice**

AdaBoost: Weak Classifier

Decision stump

- Simple-most type of **decision tree**
- Equivalent to linear classifier defined by affine hyperplane
- Hyperplane is orthogonal to axis with which it intersects in threshold θ
- Commonly not used on its own
- Formally,

$$h(x; j, \theta) = \begin{cases} +1 & x_j > \theta \\ -1 & \text{else} \end{cases}$$



where x is (d-dim.) training sample, j is dimension

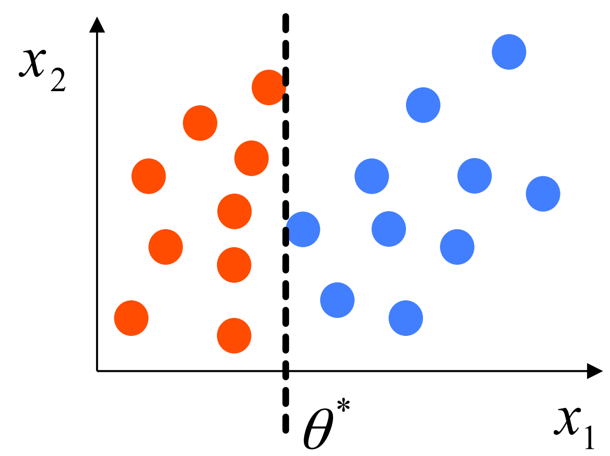
AdaBoost: Weak Classifier

- **Train a decision stump on weighted data**

$$(j^*, \theta^*) = \mathrm{argmin}_{j,\theta} \left\{ \sum_{i=1}^{n} w_t(i) \, \mathrm{I}(y_i \neq h_t(x_i)) \right\}$$

- **This consists in...**

Finding an optimum parameter θ^* for each dimension $j = 1 \ldots d$ and then select the j^* for which the weighted error is minimal.



AdaBoost: Weak Classifier

A simple training algorithm for stumps:

$\forall\ j = 1 \ldots d$

Sort samples x_i in ascending order along dimension j

$\forall\ i = 1 \ldots n$

Compute n cumulative sums $w_{cum}^{j}(i) = \sum_{k=1}^{i} w_k\, y_k$

end

Threshold θ_j is at extremum of w_{cum}^{j}

Sign of extremum gives direction p_j of inequality

end

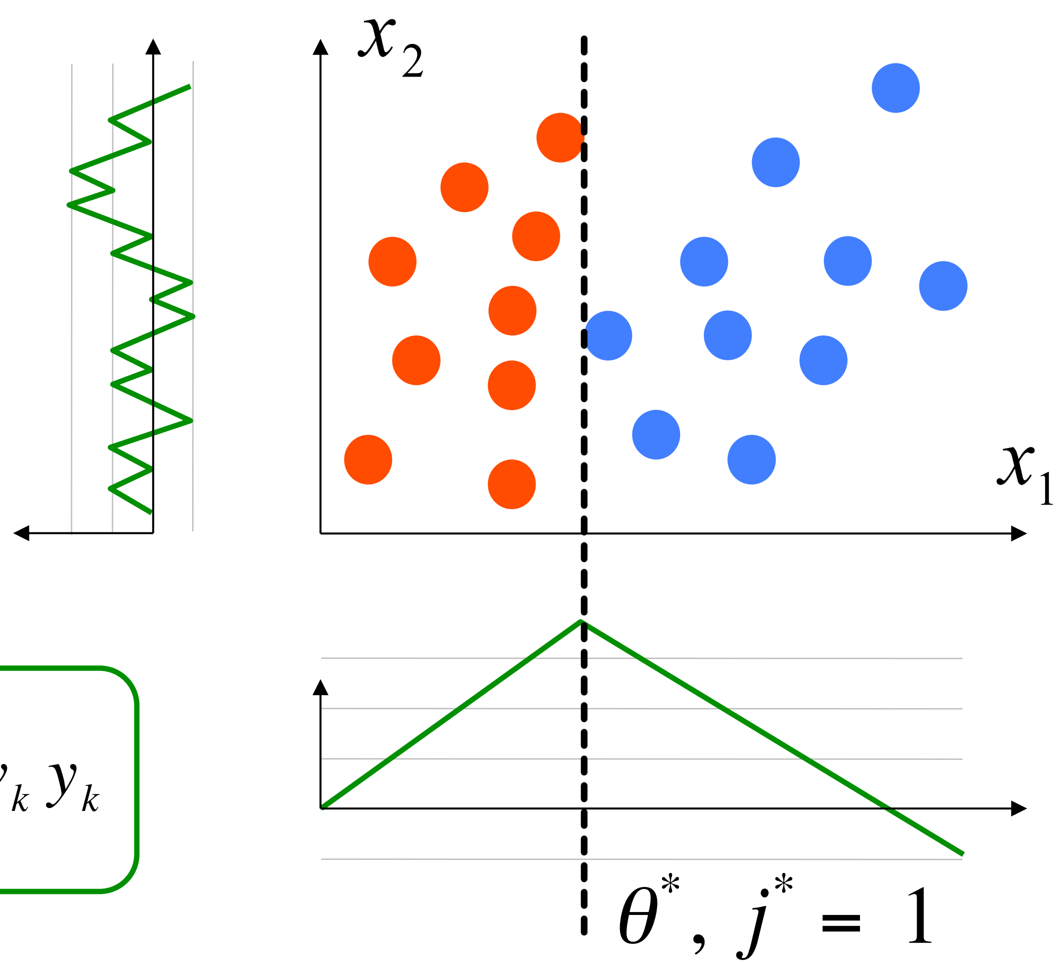
Global extremum in all d sums w_{cum} gives
threshold θ^* and **dimension** j^*

AdaBoost: Weak Classifier

Training algorithm for stumps: Intuition

- **Label** y **:**
  red: +
  blue: –

- Assuming all weights = 1



$$w_{cum}^{j}(i) = \sum_{k=1}^{i} w_k\, y_k$$

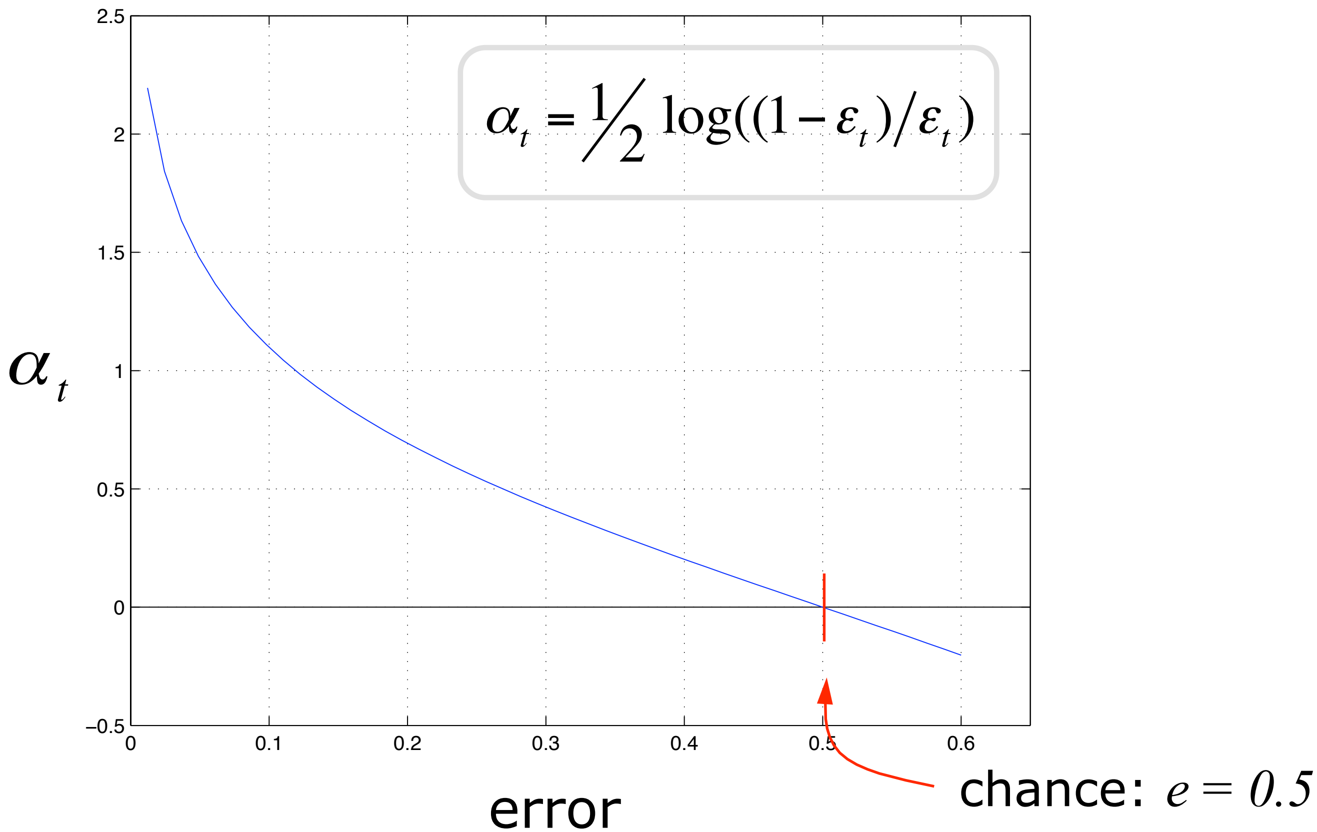
AdaBoost: Algorithm

Given the **training data** $\{(\mathbf{x_1}, y_1), \ldots, (\mathbf{x_n}, y_n)\}$ $\quad \mathbf{x} \in \mathcal{X} \quad y \in \mathcal{Y}$

1. Initialize weights $w_t(i) = 1/n$
2. For $t = 1,\ldots,T$
   - Train a **weak classifier** $h_t(x)$ on weighted training data minimizing the error $\varepsilon_t = \sum_{i=1}^{n} w_t(i)\, \mathrm{I}(y_i \neq h_t(x_i))$
   - Compute voting weight of $h_t(x)$: $\alpha_t = \frac{1}{2} \log((1-\varepsilon_t)/\varepsilon_t)$
   - Recompute weights: $w_{t+1}(i) = w_t(i) \exp\left\{-\alpha_t\, y_i\, h_t(x_i)\right\} / Z_t$
3. Make predictions using the final **strong classifier**

AdaBoost: Voting Weight

- Computing the **voting weight** α_t of a weak classifier
- α_t measures the **importance** assigned to $h_t(x_i)$



AdaBoost: Weight Update

- Looking at the weight update step:

$$w_{t+1}(i) = w_t(i) \exp\left\{-\alpha_t\, y_i\, h_t(x_i)\right\} / Z_t$$

Z_t : **Normalizer** such that w_{t+1} is a prob. distribution

$$\exp\left\{-\alpha_t\, y_i\, h_t(x_i)\right\} = \begin{cases} <1, & y_i = h_t(x_i) \\ >1, & y_i \neq h_t(x_i) \end{cases}$$

→ Weights of misclassified training samples are **increased**

→ Weights of correctly classified samples are **decreased**

- Algorithm generates weak classifier by training the next learner on the **mistakes of the previous one**
- Now we understand the name: **AdaBoost** comes from **adaptive Boosting**

AdaBoost: Strong Classifier

- **Training is completed...**

  The weak classifiers $h_{1\ldots T}(x)$ and their voting weight $\alpha_{1\ldots.T}$ are now fix

- **The resulting strong classifier is**

$$H(x_i) = \operatorname{sgn}\left(\sum_{t=1}^{T} \alpha_t h_t(x_i) \right) \longrightarrow \text{Class Result } \{+1, -1\}$$

Put your data here

Weighted majority voting scheme

AdaBoost: Algorithm

Given the **training data** $\{(\mathbf{x_1}, y_1), \ldots, (\mathbf{x_n}, y_n)\}$ $\quad \mathbf{x} \in \mathcal{X} \quad y \in \mathcal{Y}$

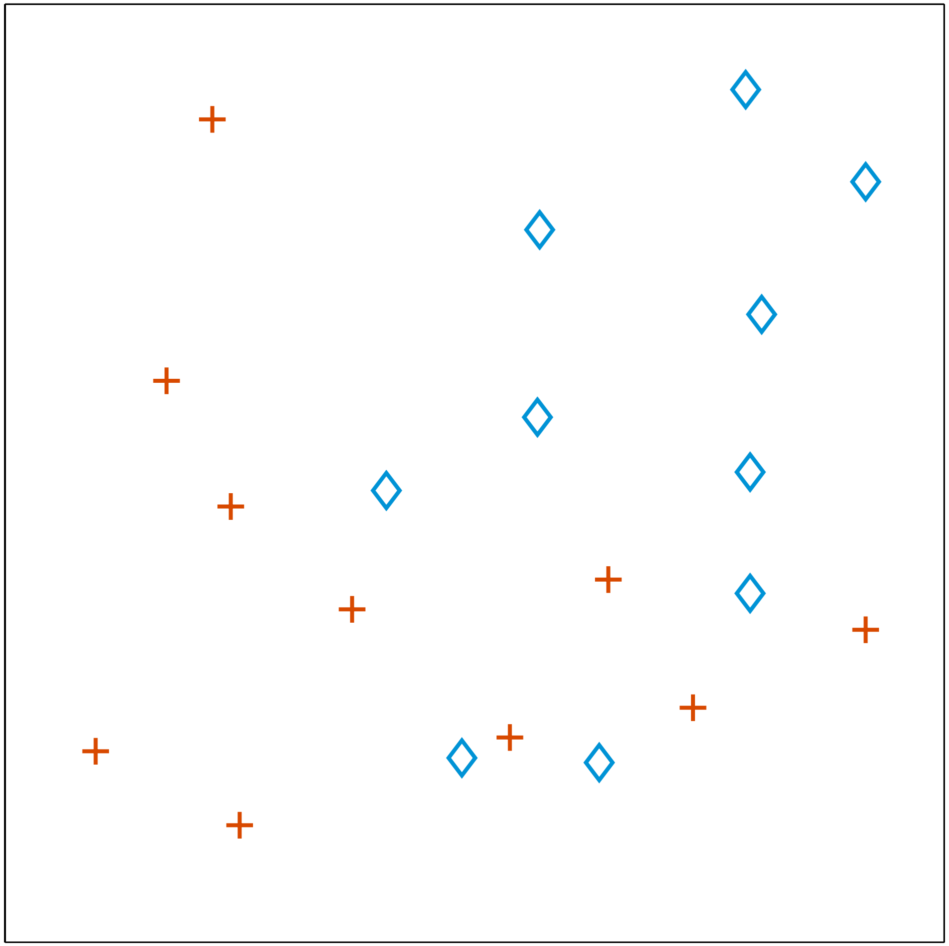
1. Initialize weights $w_t(i) = 1/n$

2. For $t = 1,\ldots,T$
   - Train a **weak classifier** $h_t(x)$ on weighted training data minimizing the error
     $$\varepsilon_t = \sum_{i=1}^{n} w_t(i)\, \mathrm{I}(y_i \neq h_t(x_i))$$
   - Compute voting weight of $h_t(x)$: $\alpha_t = \frac{1}{2} \log((1-\varepsilon_t)/\varepsilon_t)$
   - Recompute weights: $w_{t+1}(i) = w_t(i) \exp\{-\alpha_t y_i h_t(x_i)\} / Z_t$

3. Make predictions using the final **strong classifier**

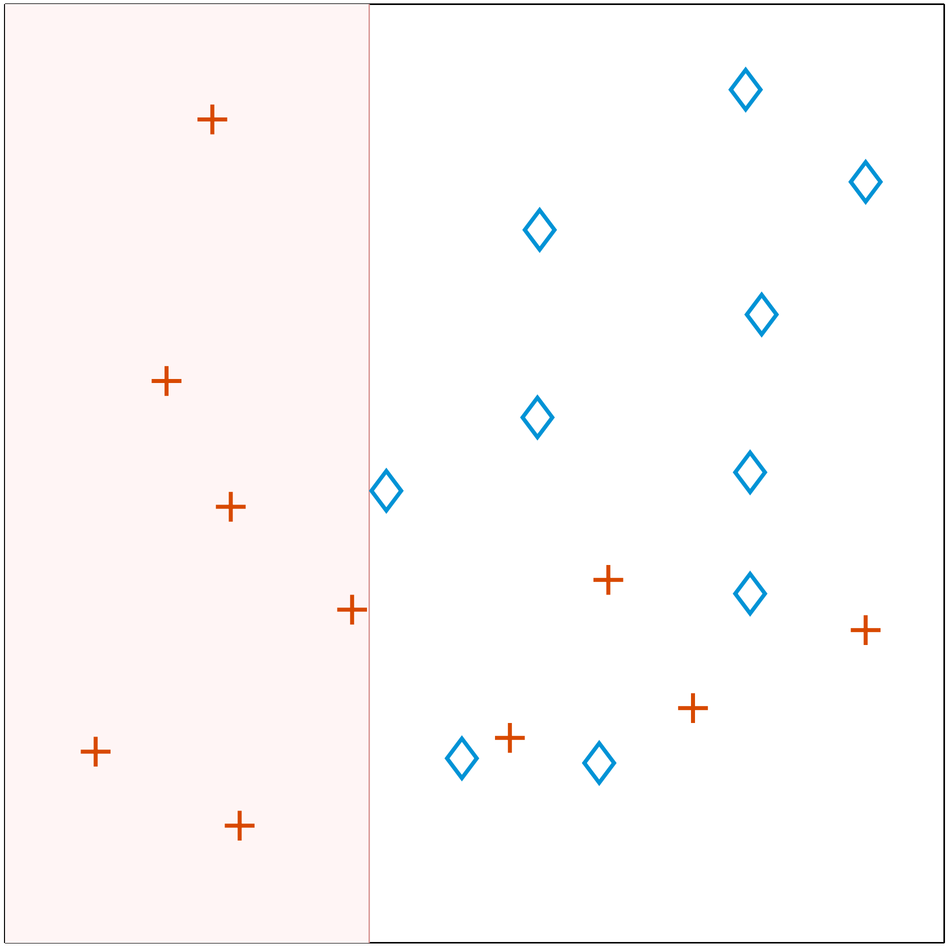
AdaBoost: Step-By-Step

- **Training data**



AdaBoost: Step-By-Step

- **Iteration 1, train weak classifier 1**



Threshold
$\theta^* = 0.37$

Dimension
$j^* = 1$

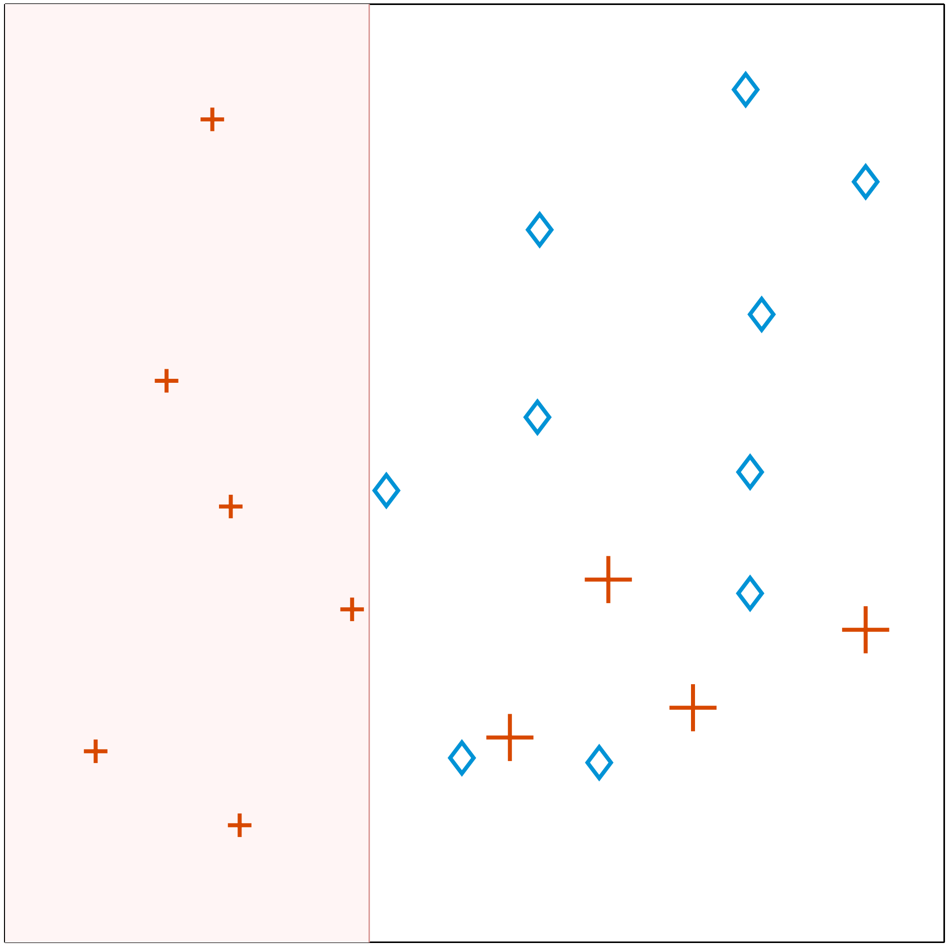
Weighted error
$e_t = 0.2$

Voting weight
$\alpha_t = 1.39$

Total error = 4

AdaBoost: Step-By-Step

- **Iteration 1, recompute weights**



Threshold
$\theta^* = 0.37$

Dimension
$j^* = 1$

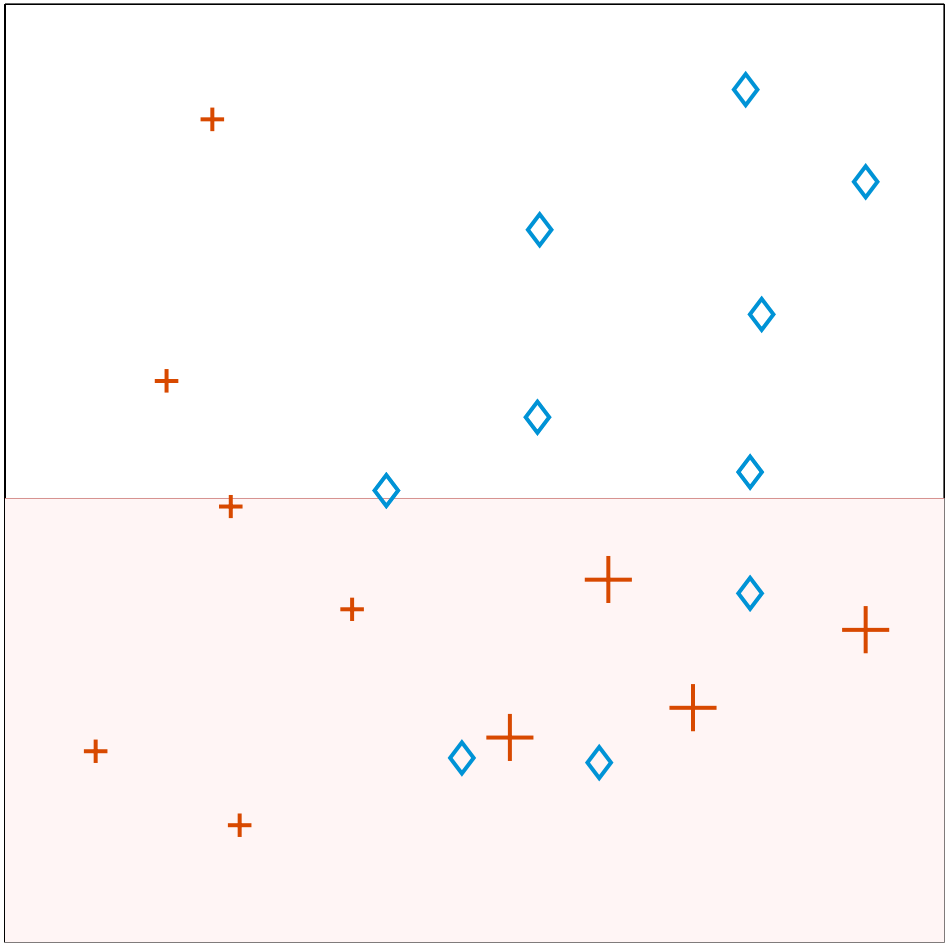
Weighted error
$e_t = 0.2$

Voting weight
$\alpha_t = 1.39$

Total error = 4

AdaBoost: Step-By-Step

- **Iteration 2, train weak classifier 2**



Threshold
$\theta^* = 0.47$

Dimension
$j^* = 2$

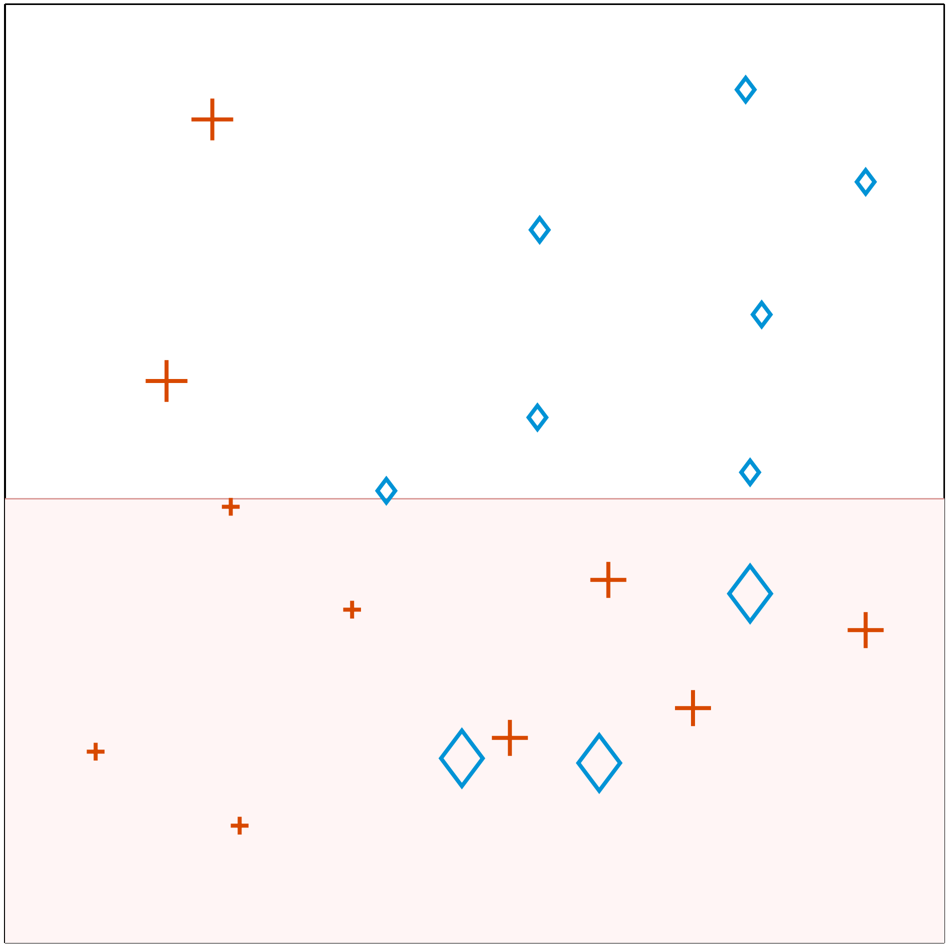
Weighted error
$e_t = 0.16$

Voting weight
$\alpha_t = 1.69$

Total error = 5

AdaBoost: Step-By-Step

- **Iteration 2, recompute weights**



Threshold
$\theta^* = 0.47$

Dimension
$j^* = 2$

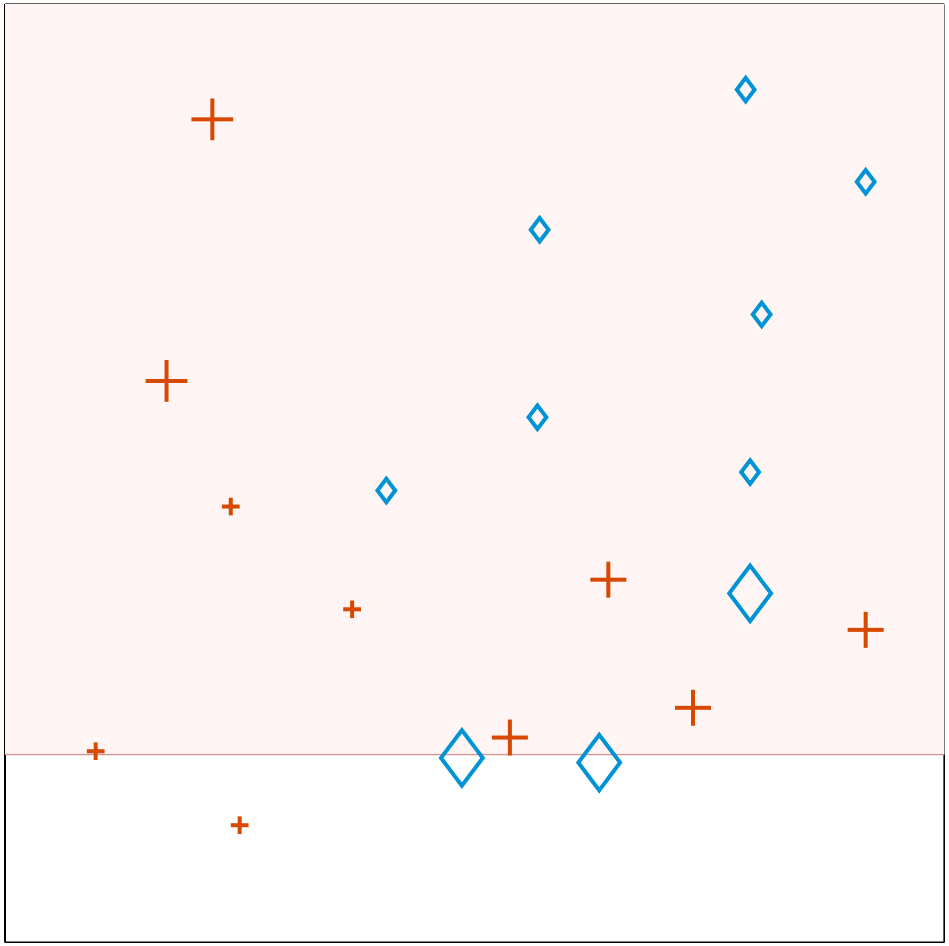
Weighted error
$e_t = 0.16$

Voting weight
$\alpha_t = 1.69$

Total error = 5

AdaBoost: Step-By-Step

- **Iteration 3, train weak classifier 3**



Threshold
$\theta^* = 0.14$

Dimension, sign
$j^* = 2$, neg

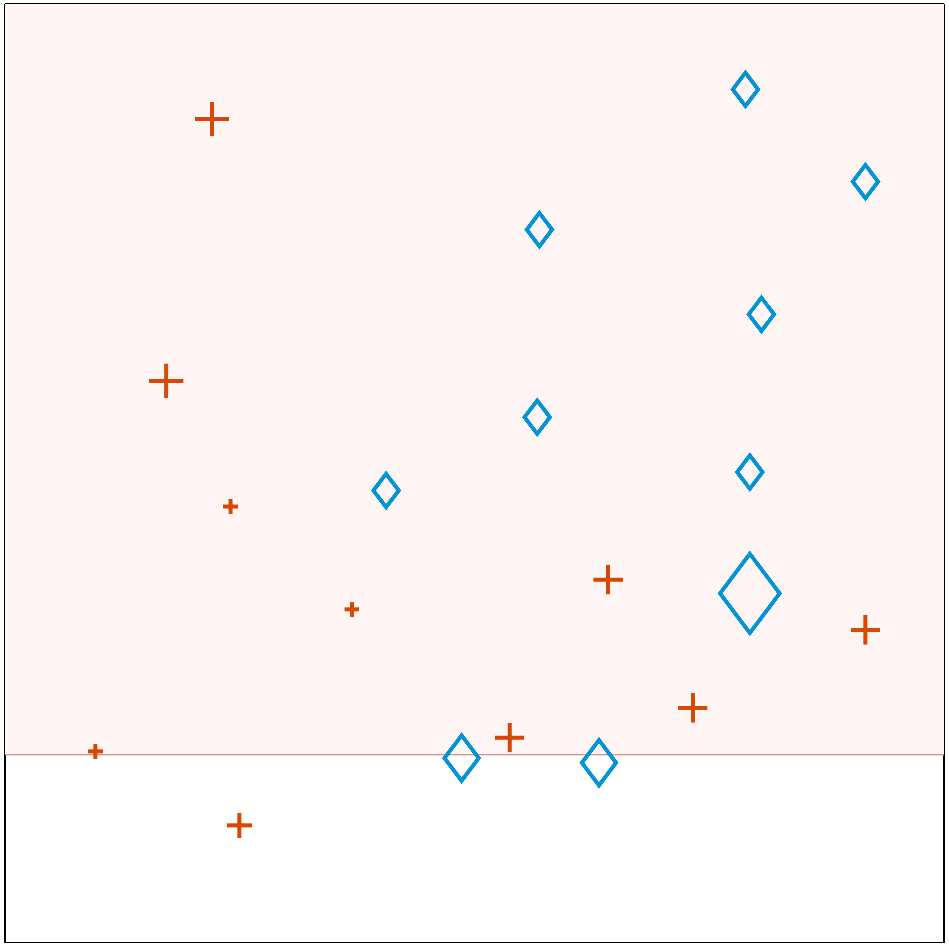
Weighted error
$e_t = 0.25$

Voting weight
$\alpha_t = 1.11$

Total error = 1

AdaBoost: Step-By-Step

- **Iteration 3, recompute weights**



Threshold
$\theta^* = 0.14$

Dimension, sign
$j^* = 2$, neg

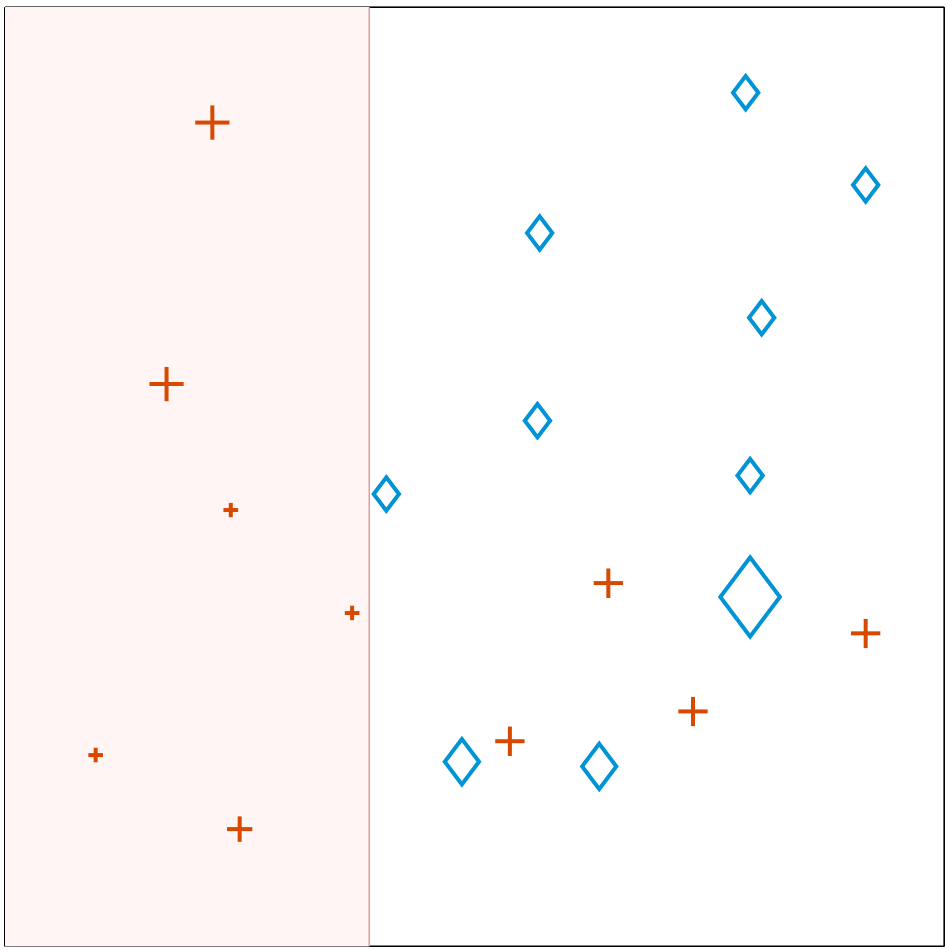
Weighted error
$e_t = 0.25$

Voting weight
$\alpha_t = 1.11$

Total error = 1

AdaBoost: Step-By-Step

- **Iteration 4, train weak classifier 4**



Threshold
$\theta^* = 0.37$

Dimension
$j^* = 1$

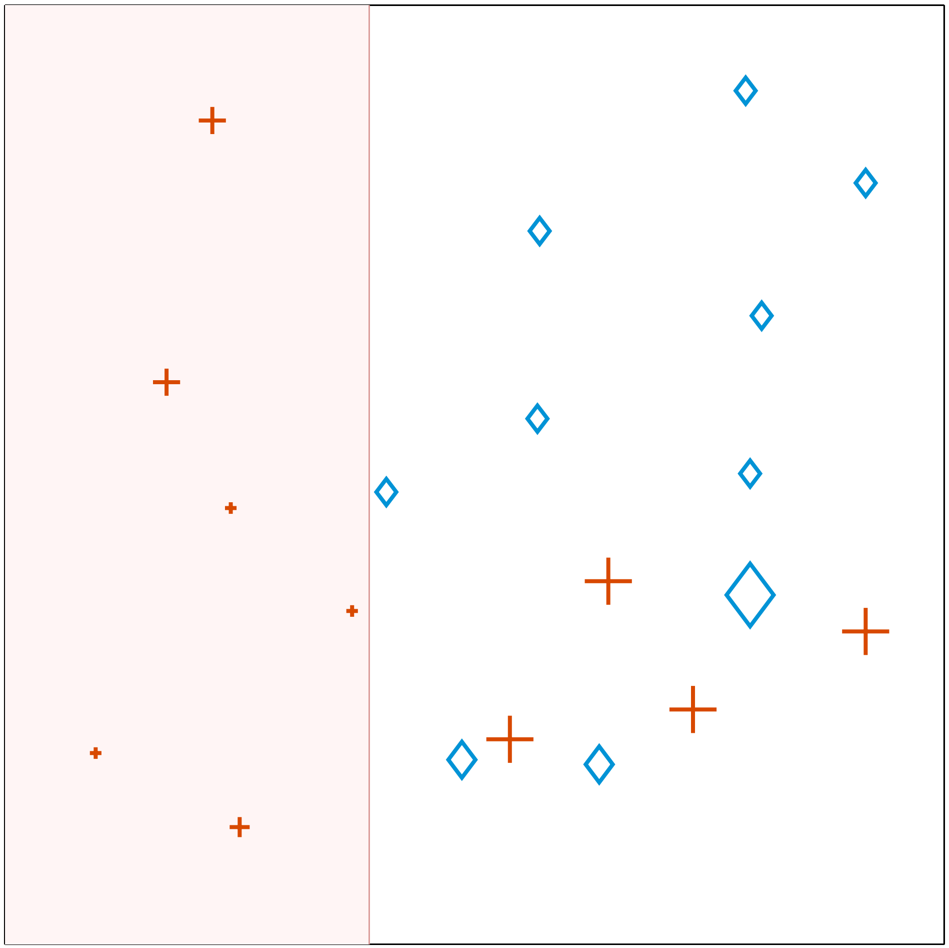
Weighted error
$e_t = 0.20$

Voting weight
$\alpha_t = 1.40$

Total error = 1

AdaBoost: Step-By-Step

- **Iteration 4, recompute weights**



Threshold
$\theta^* = 0.37$

Dimension
$j^* = 1$

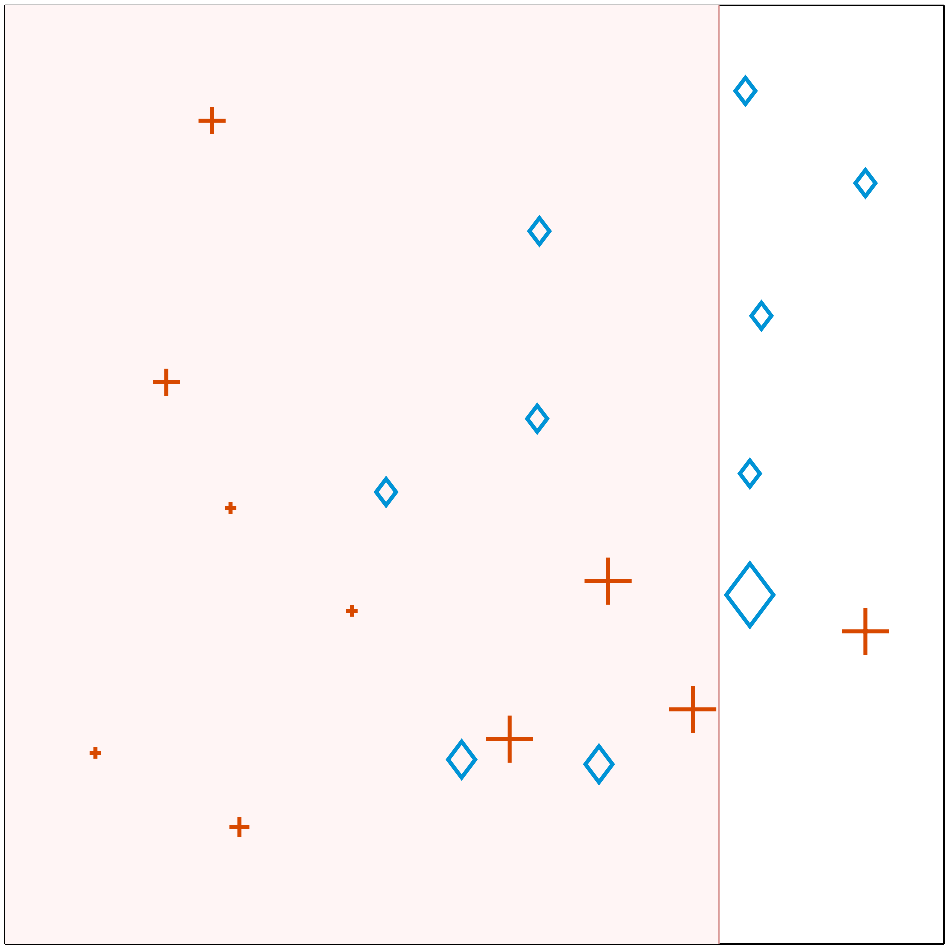
Weighted error
$e_t = 0.20$

Voting weight
$\alpha_t = 1.40$

Total error = 1

AdaBoost: Step-By-Step

- **Iteration 5, train weak classifier 5**



Threshold
$\theta^* = 0.81$

Dimension
$j^* = 1$

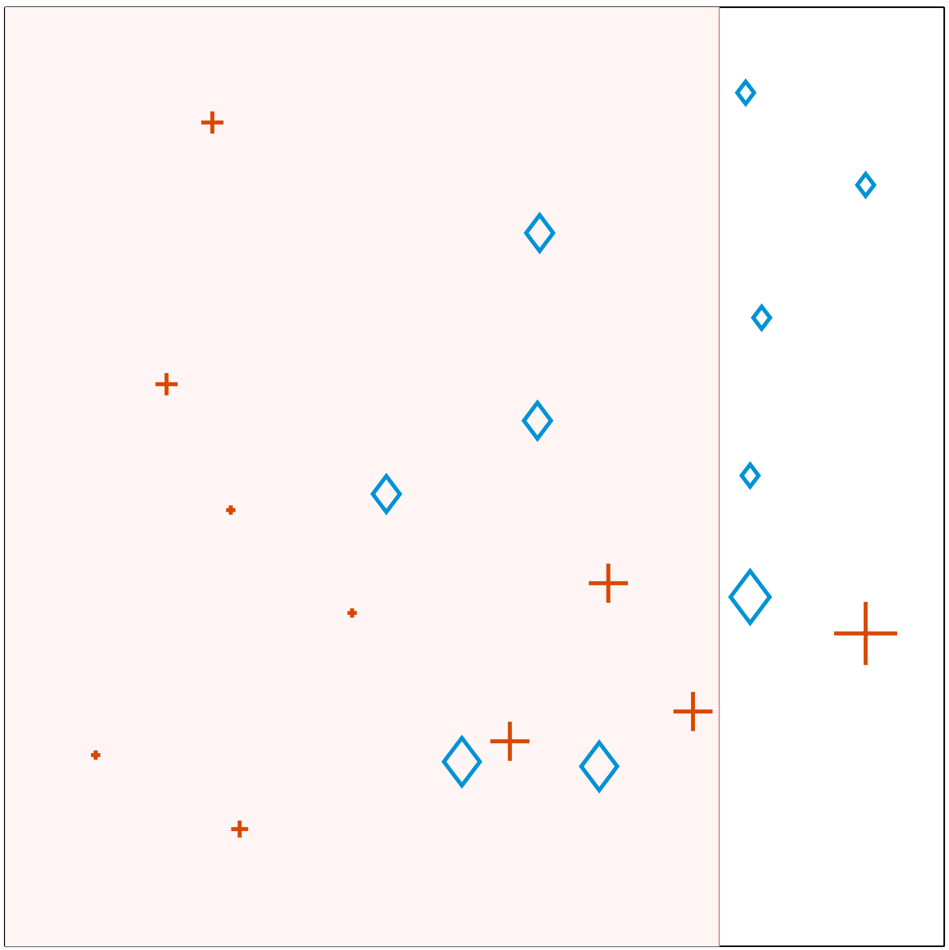
Weighted error
$e_t = 0.28$

Voting weight
$\alpha_t = 0.96$

Total error = 1

AdaBoost: Step-By-Step

- **Iteration 5, recompute weights**



Threshold
$\theta^* = 0.81$

Dimension
$j^* = 1$

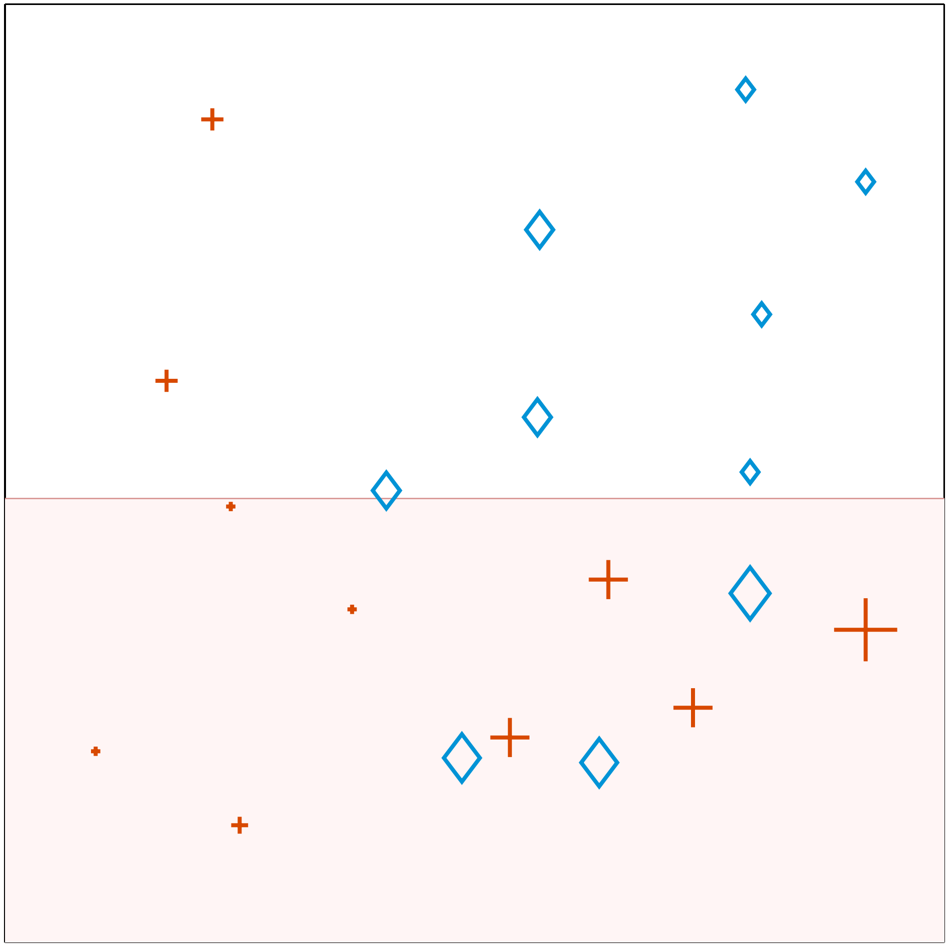
Weighted error
$e_t = 0.28$

Voting weight
$\alpha_t = 0.96$

Total error = 1

AdaBoost: Step-By-Step

- **Iteration 6, train weak classifier 6**



Threshold
$\theta^* = 0.47$

Dimension
$j^* = 2$

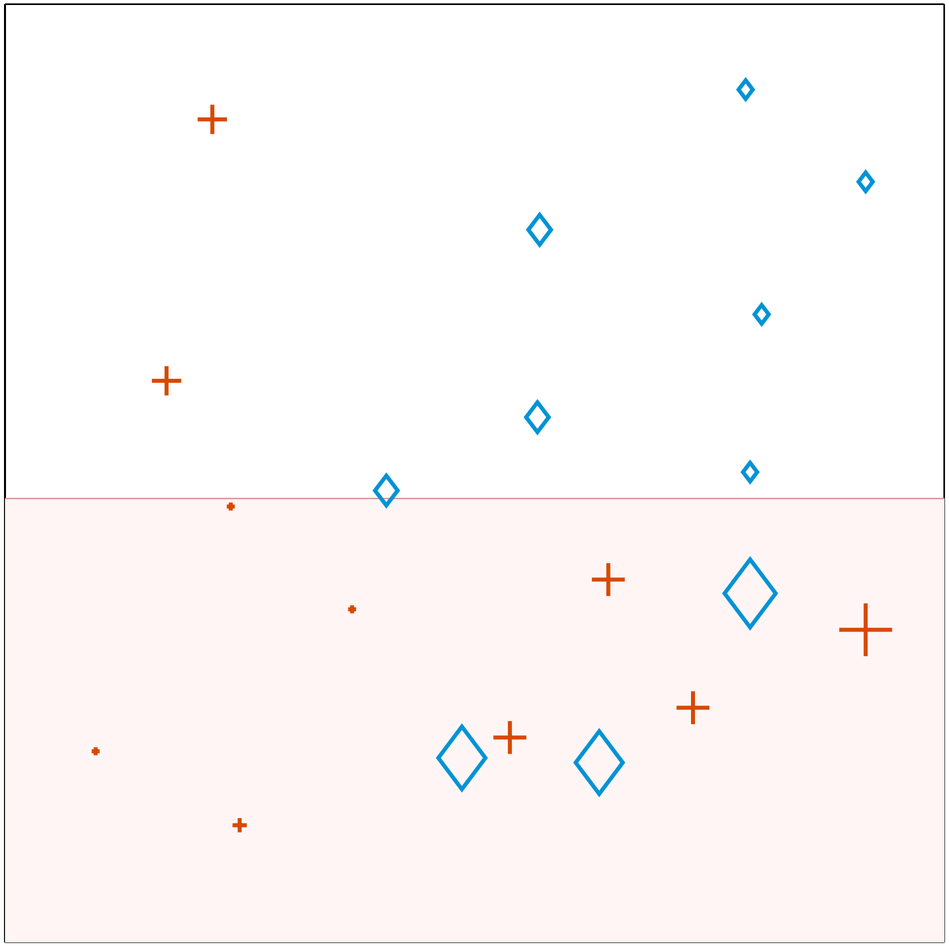
Weighted error
$e_t = 0.29$

Voting weight
$\alpha_t = 0.88$

Total error = 1

AdaBoost: Step-By-Step

- **Iteration 6, recompute weights**



Threshold
$\theta^* = 0.47$

Dimension
$j^* = 2$

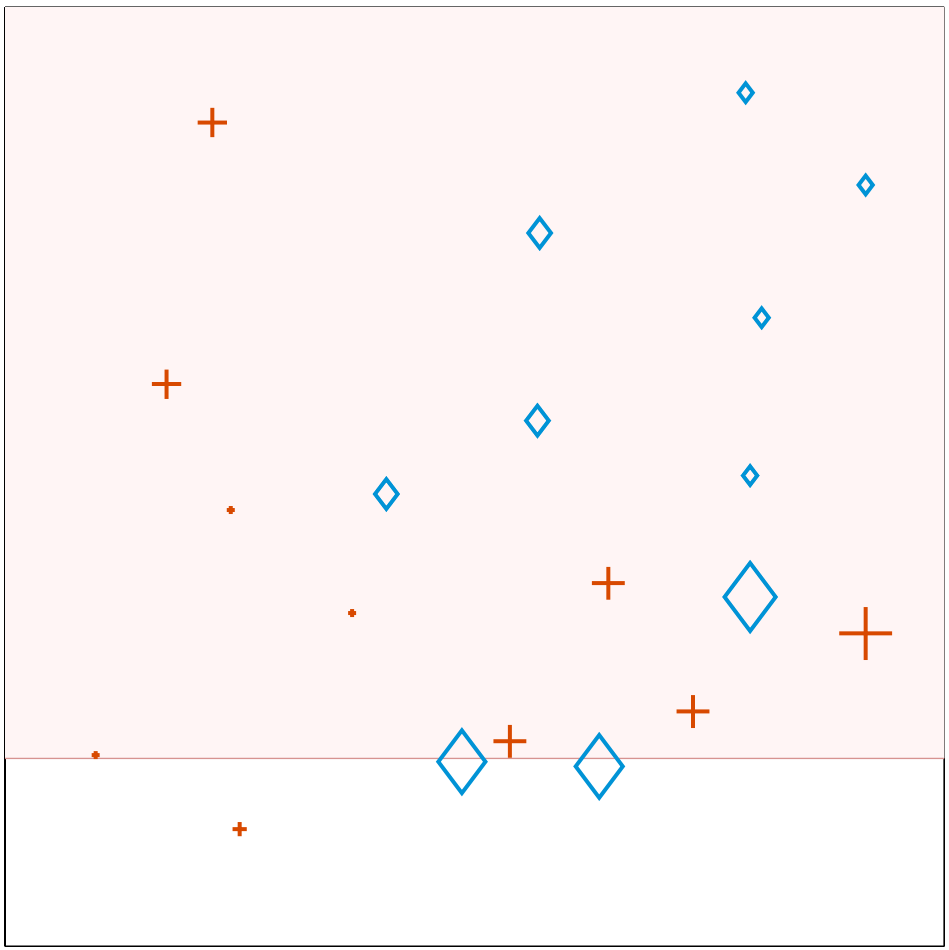
Weighted error
$e_t = 0.29$

Voting weight
$\alpha_t = 0.88$

Total error = 1

AdaBoost: Step-By-Step

- **Iteration 7, train weak classifier 7**



Threshold
$\theta^* = 0.14$

Dimension, sign
$j^* = 2$, neg

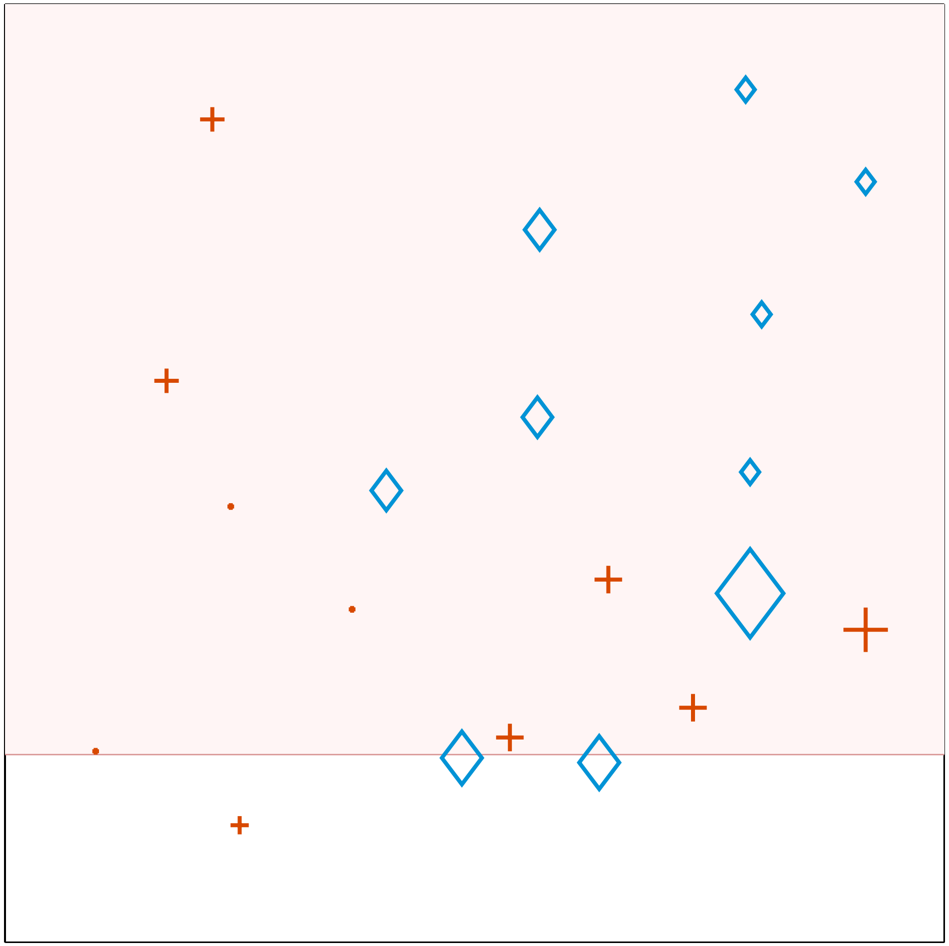
Weighted error
$e_t = 0.29$

Voting weight
$\alpha_t = 0.88$

Total error = 1

AdaBoost: Step-By-Step

- **Iteration 7, recompute weights**



Threshold
$\theta^* = 0.14$

Dimension, sign
$j^* = 2$, neg

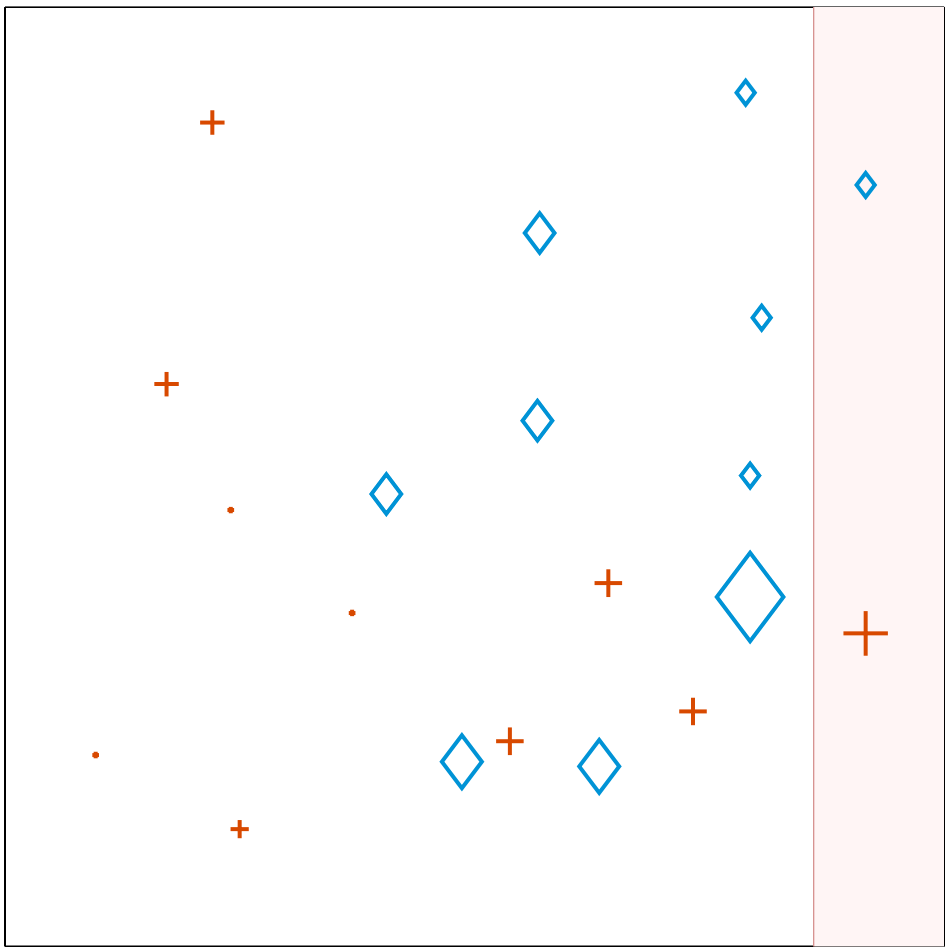
Weighted error
$e_t = 0.29$

Voting weight
$\alpha_t = 0.88$

Total error = 1

AdaBoost: Step-By-Step

- **Iteration 8, train weak classifier 8**



Threshold
$\theta^* = 0.93$

Dimension, sign
$j^* = 1$, neg

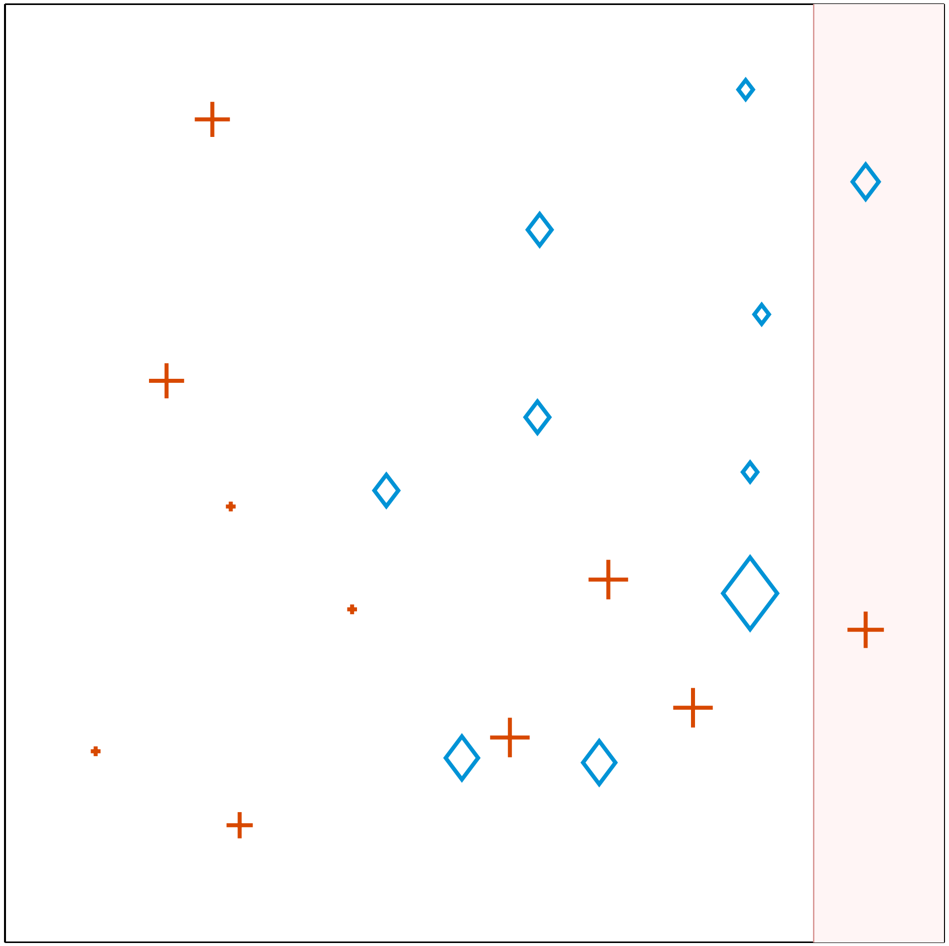
Weighted error
$e_t = 0.25$

Voting weight
$\alpha_t = 1.12$

Total error = 0

AdaBoost: Step-By-Step

- **Iteration 8, recompute weights**



Threshold
$\theta^* = 0.93$

Dimension, sign
$j^* = 1$, neg

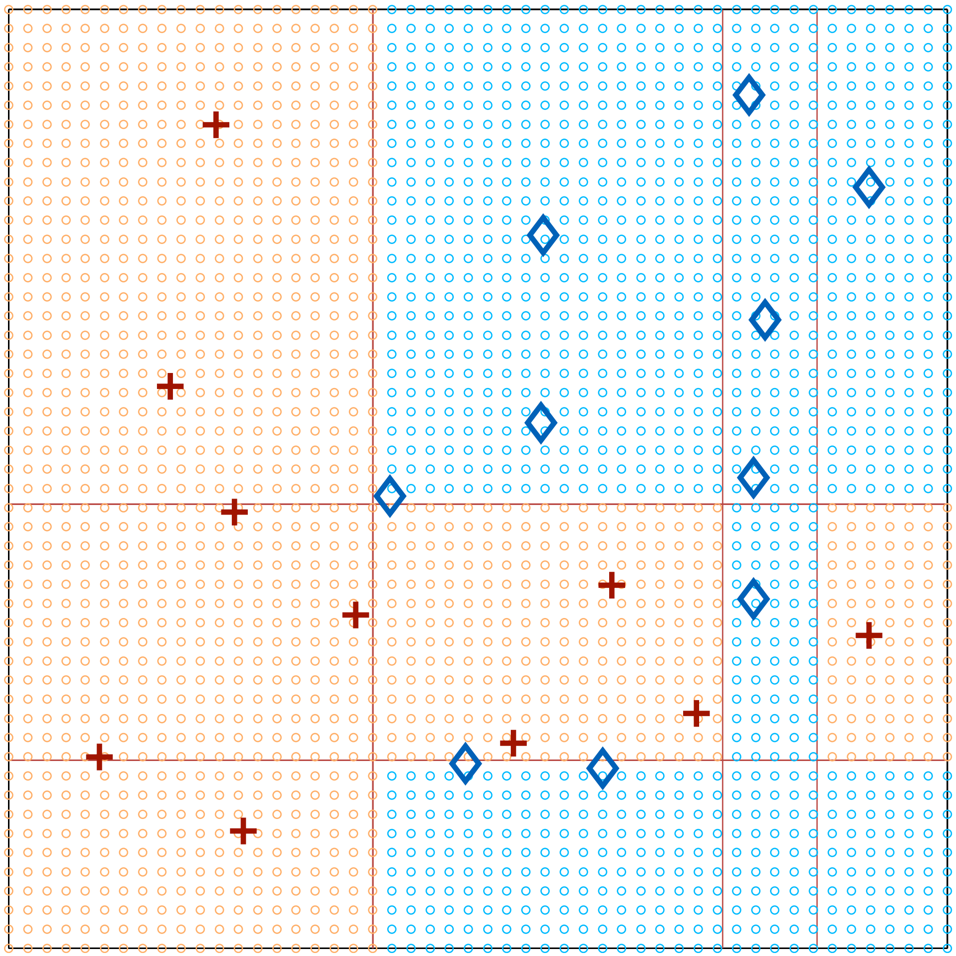
Weighted error
$e_t = 0.25$

Voting weight
$\alpha_t = 1.12$

Total error = 0

AdaBoost: Step-By-Step

- **Final Strong Classifier**



Total training error = 0

(Rare in practice)

AdaBoost: Why Does it Work?

AdaBoost minimizes the training error

- Upper bound theorem: the following upper bound holds on the **training error** of H

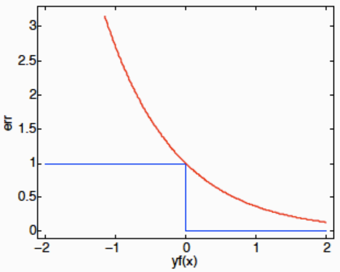
$$\frac{1}{n}\left|\left\{i : H(x_i) \neq y_i\right\}\right| \leq \prod_{t=1}^{T} Z_t$$

- **Proof:** By unravelling the weight update rule

$$\begin{aligned} D_{T+1}(i) &= \frac{D_t(i) exp(-\alpha_t y_i h_t(x_i))}{Z_t} \\ &= \frac{exp(-\sum_t \alpha_t y_i h_t(x_i))}{m \prod_t Z_t} = \frac{exp(-y_i f(x_i))}{m \prod_t Z_t} \end{aligned}$$

If $H(x_i) \neq y_i$ then $y_i f(x_i) \leq 0$ implying that $exp(-y_i f(x_i)) > 1$, thus

$$\begin{aligned} [\![H(x_i) \neq y_i]\!] &\leq exp(-y_i f(x_i)) \\ \frac{1}{m}\sum_i [\![H(x_i) \neq y_i]\!] &\leq \frac{1}{m}\sum_i exp(-y_i f(x_i)) \\ &= \sum_i (\prod_t Z_t) D_{T+1}(i) = \prod_t Z_t \end{aligned}$$



Sochman, Matas

AdaBoost: Why Does it Work?

Ergo...

- Instead of minimizing the **training error** directly, its **upper bound** can be **minimized**
- We have to minimize the normalizer

$$Z_t = \sum_i w_t(i) \exp\left\{ -\alpha_t \, y_i \, h_t(x_i) \right\}$$

in each training round.

This is achieved by

- Finding the **optimal voting weight** α_t
- Finding the **optimal weak classifier** $h_t(x)$

AdaBoost: Why Does it Work?

Optimal voting weight

Theorem:

The minimizer of the bound is

$$\boxed{\alpha_t = \tfrac{1}{2} \log((1-\varepsilon_t)/\varepsilon_t)}$$

Proof:

$$\begin{aligned}
\frac{dZ}{d\alpha} = -\sum_{i=1}^{m} D(i) y_i h(x_i) e^{-y_i \alpha_i h(x_i)} &= 0 \\
-\sum_{i:y_i=h(x_i)} D(i) e^{-\alpha} + \sum_{i:y_i\neq h(x_i)} D(i) e^{\alpha} &= 0 \\
-e^{-\alpha}(1-\epsilon) + e^{\alpha}\epsilon &= 0 \\
\alpha &= \frac{1}{2}\log\frac{1-\epsilon}{\epsilon}
\end{aligned}$$

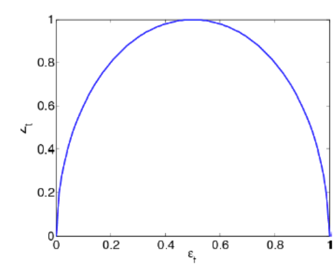
Optimal weak classifier

Theorem:

Z_t is minimized by selecting $h_t(x)$ with minimal weighted error ε_t

Proof:

$$\begin{aligned}
Z_t &= \sum_{i=1}^{m} D_t(i) e^{-y_i \alpha_i h_t(x_i)} \\
&= \sum_{i:y_i=h_t(x_i)} D_t(i) e^{-\alpha_t} + \sum_{i:y_i\neq h_t(x_i)} D_t(i) e^{\alpha_t} \\
&= (1-\epsilon_t) e^{-\alpha_t} + \epsilon_t e^{\alpha_t} \\
&= 2\sqrt{\epsilon_t(1-\epsilon_t)}
\end{aligned}$$



Sochman, Matas

AdaBoost in Action

AdaBoost in Action

Kai O. Arras
Social Robotics Lab, University of Freiburg

Nov 2009 Social Robotics Laboratory

AdaBoost: Summary

- **Misclassified** samples receive **higher weight.** The higher the weight the **"more attention"** a training sample receives
- Algorithm generates weak classifier by training the next learner **on the mistakes** of the previous one
- AdaBoost **minimizes the upper bound** of the training error by properly choosing the optimal weak classifier and voting weight. AdaBoost can further be shown to **maximize the margin** (proof in literature)
- **Large impact** on ML community and beyond

Chapter Contents

- Machine Learning: A Survey
- Classification
- AdaBoost
- **People Detection with Boosted Features**
- Place Recognition with Boosted Features

Motivation: People Detection

- **People detection and tracking** is a key component for many vision systems and for all robots in human environments:
  - Human-Robot-Interaction (HRI)
  - Social Robotics: social learning, learning by imitation and observation
  - Motion planning in populated environments
  - Human activity and intent recognition
  - Abnormal behavior detection
  - Crowd behavior analysis and control

Motivation: People Detection

- **Where are the people?**

Motivation: People Detection

- **Where are the people?**
- **Why is it hard?**

  Range data contain little information on people

  Hard in cluttered environments

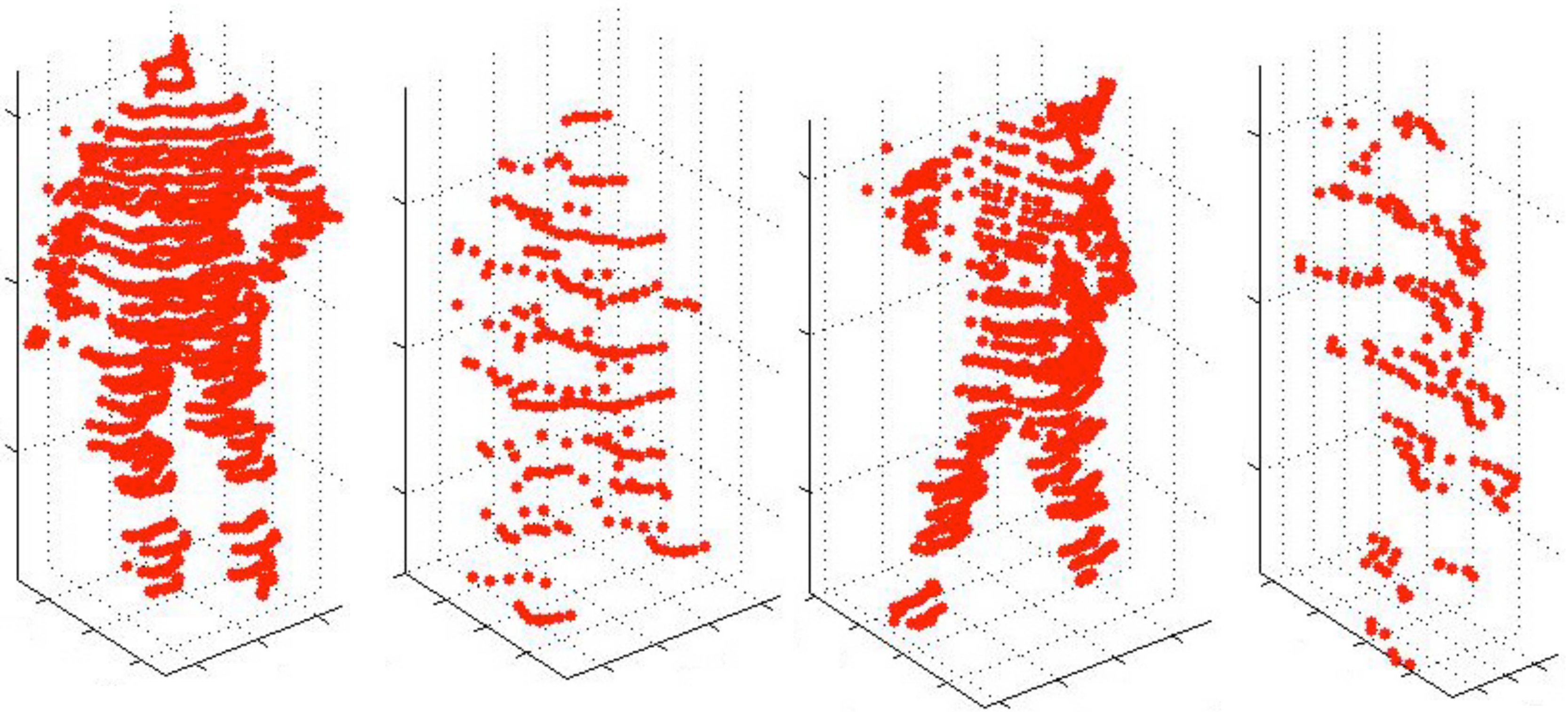
Motivation: People Detection

- **Appearance** of humans in range data **changes drastically** with:
  - Body pose
  - Distance to sensor
  - Occlusion and self-occlusion

- **2D range data** from a SICK laser scanner

Motivation: People Detection

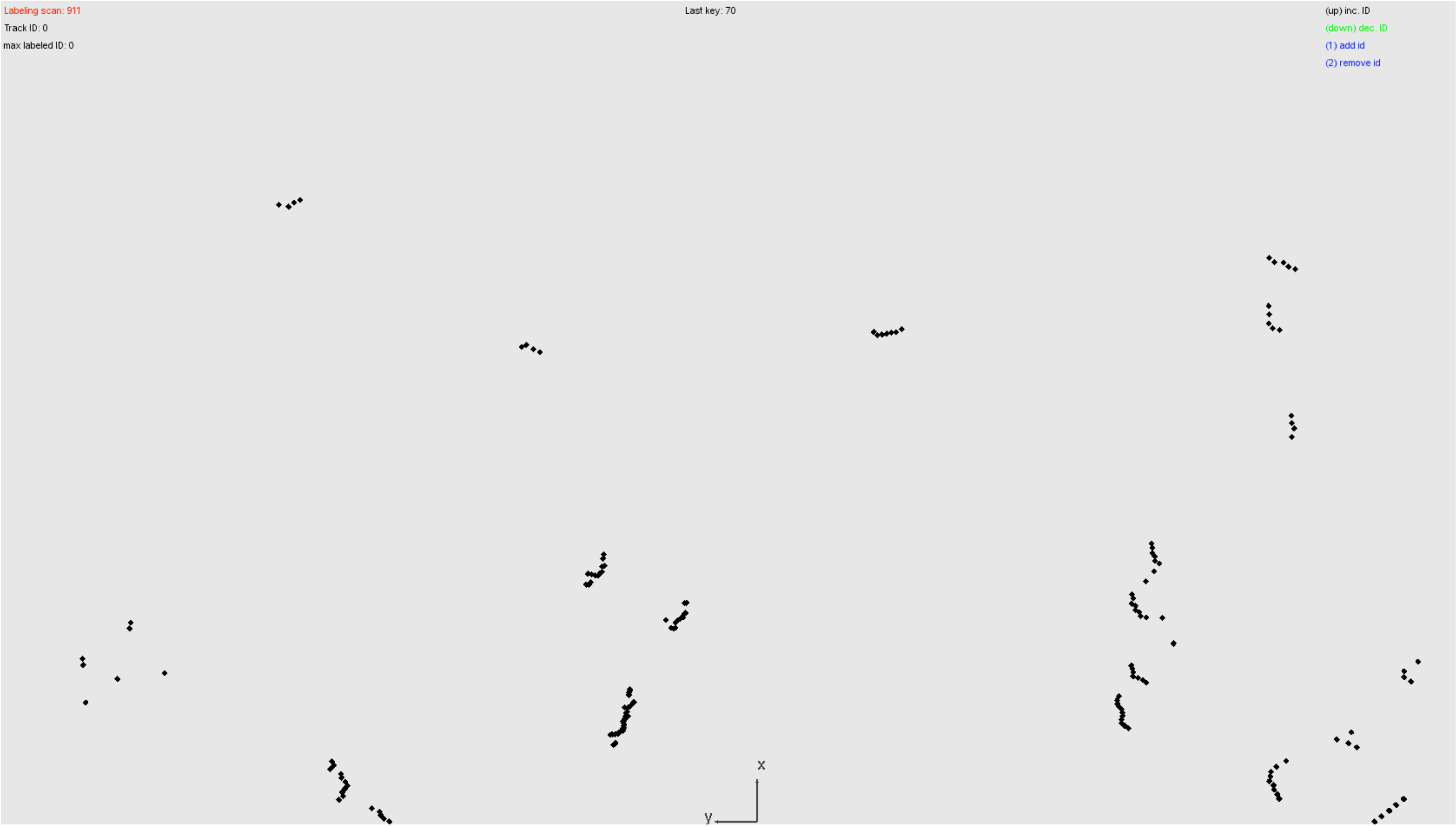
- Appearance of humans in **3D range data** (Velodyne scanner)



Motivation: People Detection

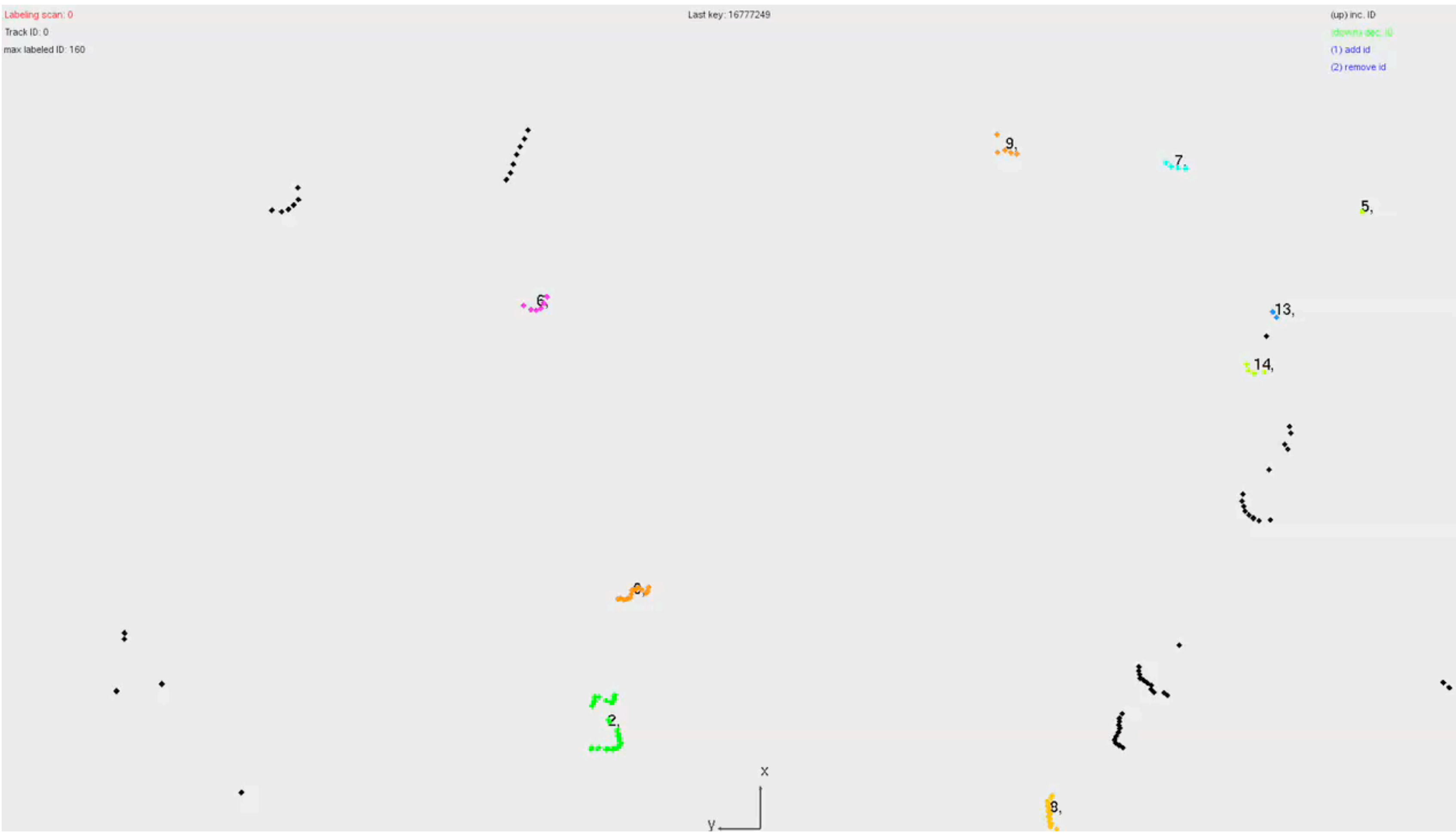
Motivation: People Detection

- Freiburg Main Station data set: **raw data**



Motivation: People Detection

- Freiburg Main Station data set: **annotations**



Approach

- Can we find **robust features** for people, legs and groups of people in 2D range data?
- What are the **best features** for people detection?
- Can we find people that **do not move**?

Approach:

- Classifying **groups of adjacent** beams (segments)
- Computing a set of scalar features on these groups
- **Boosting the features**

Related Work

- **People Tracking**

  [Fod et al. 2002]
  [Kleinhagenbrock et al. 2002]
  [Schulz et al. 2003]
  [Scheutz et al. 2004]
  [Topp et al. 2005]
  [Cui et al. 2005]
  [Schulz 2006]
  [Mucientes et al. 2006]

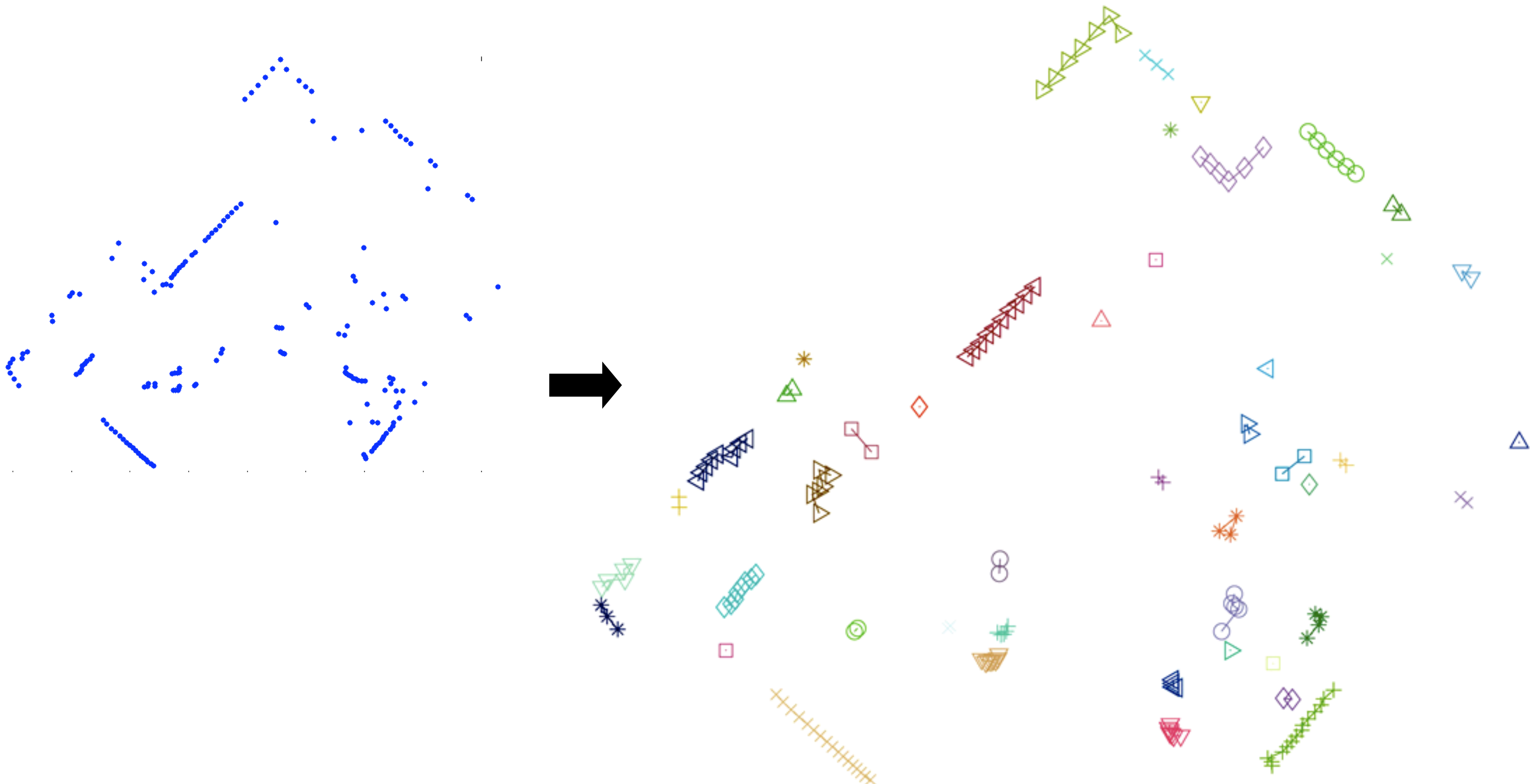
SLAM in dynamic env.

[Montemerlo et al. 2002]
[Hähnel et al. 2003]
[Wang et al. 2003]
...

- People detection done with very simple classifiers: **manual** feature selection, **heuristic** thresholds
- **Typically:** narrow local-minima blobs that move

Segmentation

- Divide the scan into segments

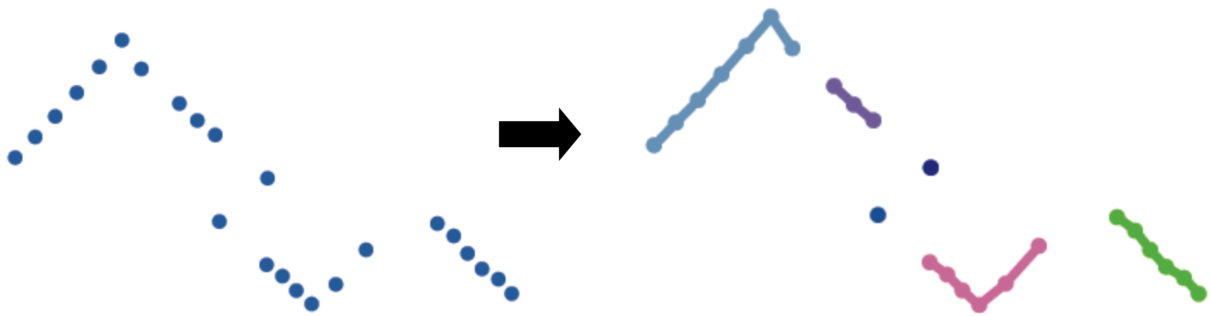


Range image segmentation

Segmentation

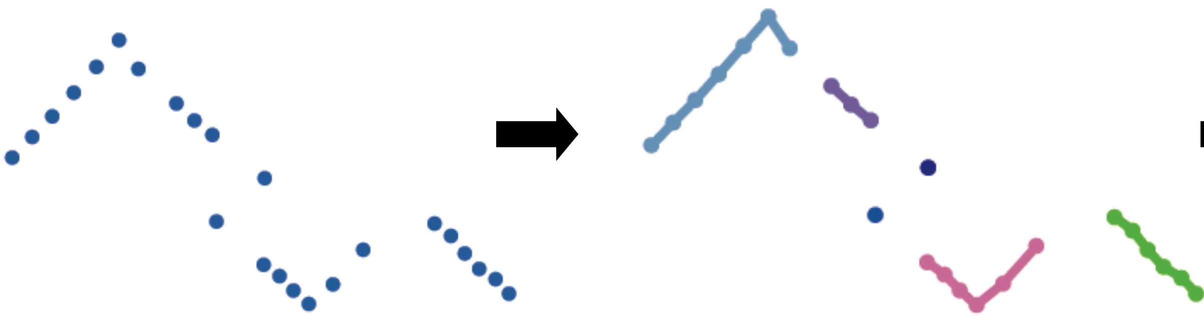
- **Method:** Jump distance condition
- **Size filter:**
  rejection of too small segments

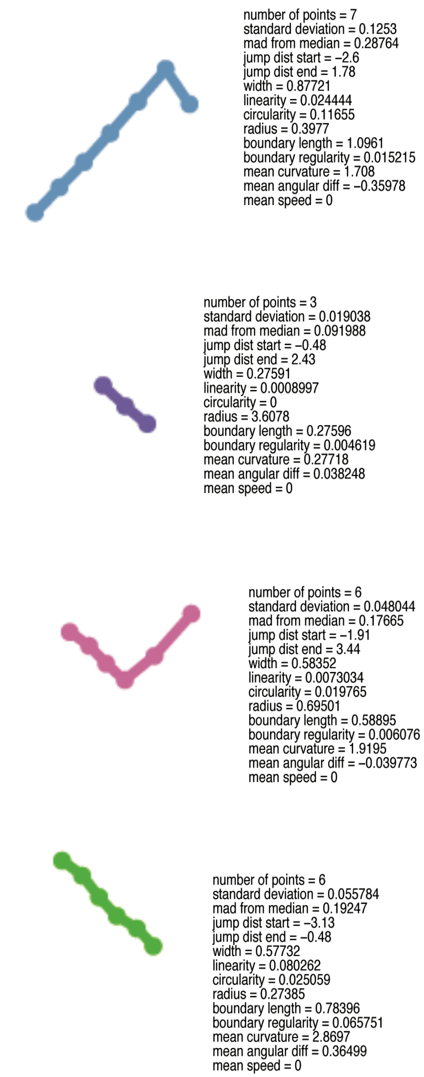
Segmentation



- **Method:** Jump distance condition
- **Size filter:**
  rejection of too small segments

Segmentation

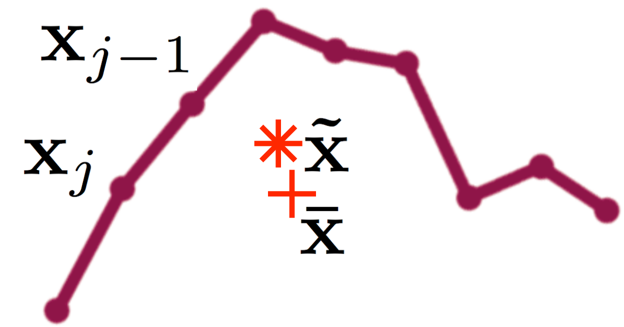




- **Method:** Jump distance condition
- **Size filter:**
  rejection of too small segments

Features

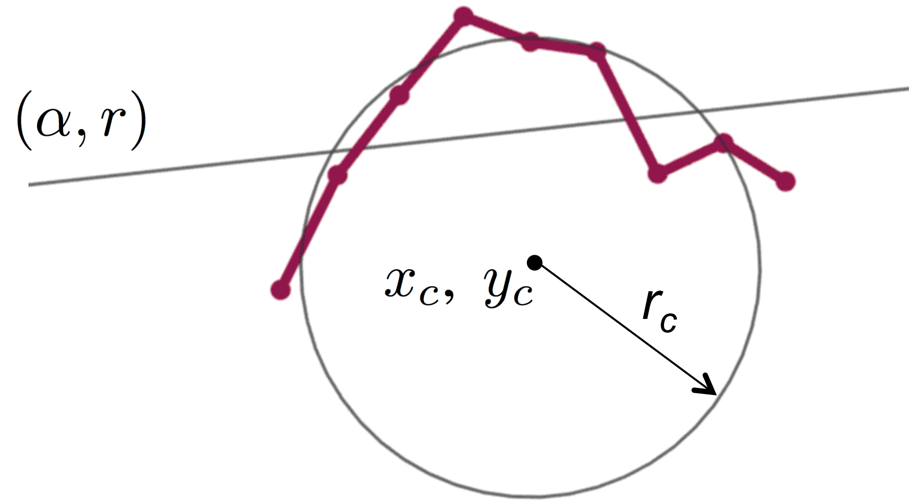
Segment S_i



1. Number of points $n = |S_i|$
2. Standard Deviation $\sigma = \sqrt{\frac{1}{n-1}\sum ||\mathbf{x}_j - \bar{\mathbf{x}}||^2}$
3. Mean avg. deviation from median $\varsigma = \frac{1}{n}\sum ||\mathbf{x}_j - \tilde{\mathbf{x}}||$
4. Jump dist. to preceding segment $\delta_{j-1,j}$
5. Jump dist. to succeeding segment $\delta_{j,j+1}$
6. Width $w_i = ||\mathbf{x}_1 - \mathbf{x}_n||$

Features

Segment S_i



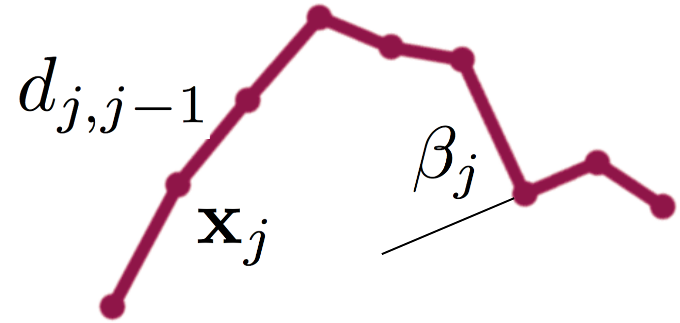
7. Linearity $s_l = \sum (x_j cos(\alpha) + y_j sin(\alpha) - r)^2$

8. Circularity $s_c = \sum (r_c - \sqrt{(x_c - x_i)^2 + (y_c - y_i)^2})^2$

9. Radius r_c

Features

Segment S_i



10. Boundary Length $l = \sum_j d_{j,j-1}$

11. Boundary Regularity $\sigma_d = \sqrt{\frac{1}{n-1} \sum (d_{j,j-1} - \bar{d})^2}$

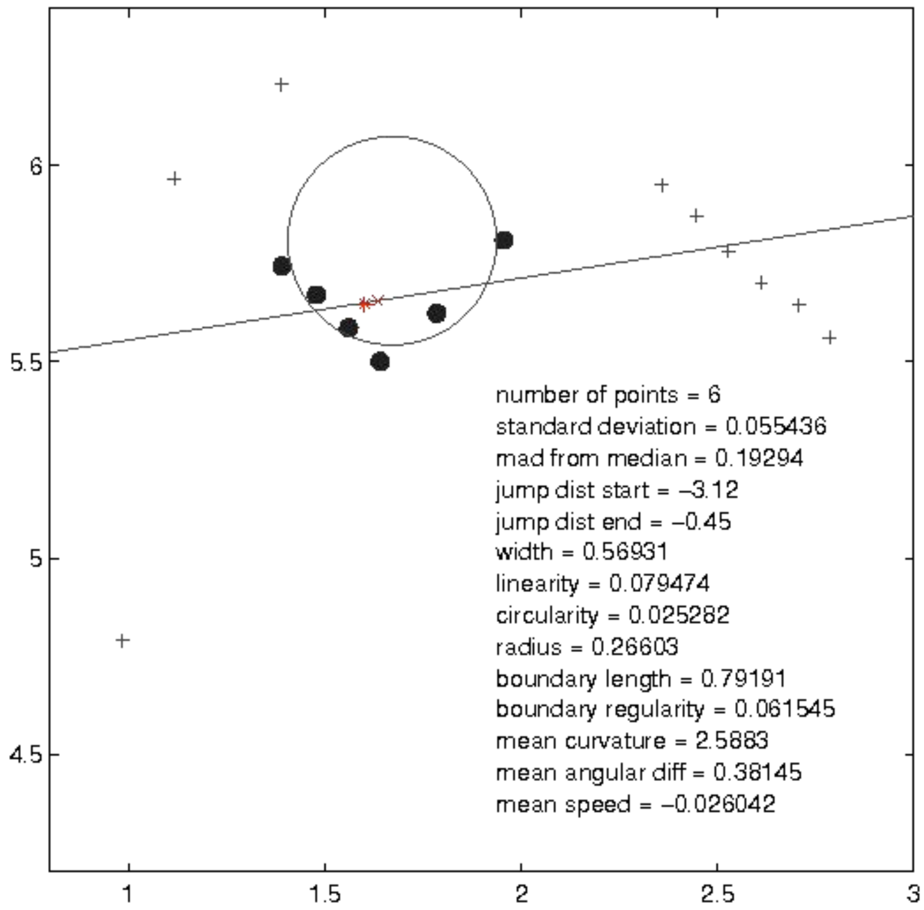
12. Mean curvature $\bar{k} = \frac{1}{n} \sum \hat{k}_j$

13. Mean angular difference $\bar{\beta} = \frac{1}{n} \sum \beta_j$

14. Mean speed $\bar{v} = \frac{1}{n} \sum \frac{\rho_j^{k+1} - \rho_j^k}{\Delta T}$

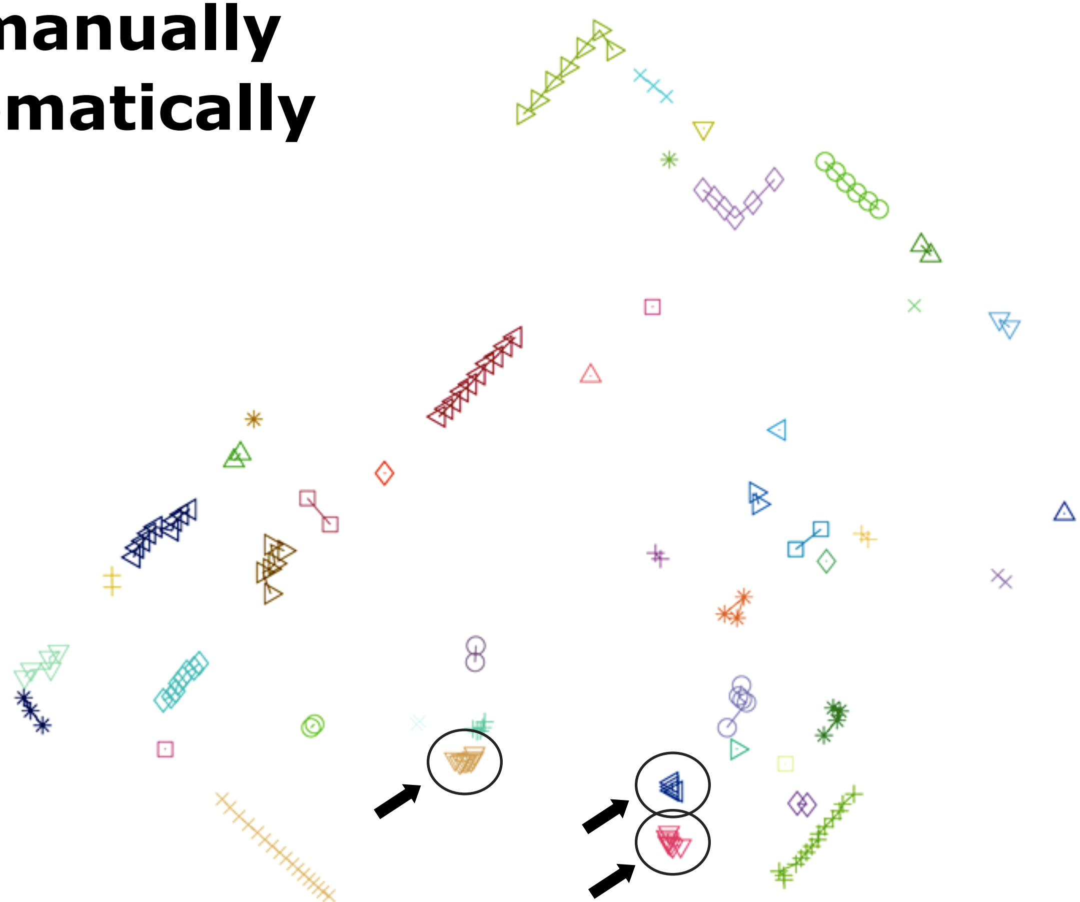
Features

- Resulting **feature signature** for each segment



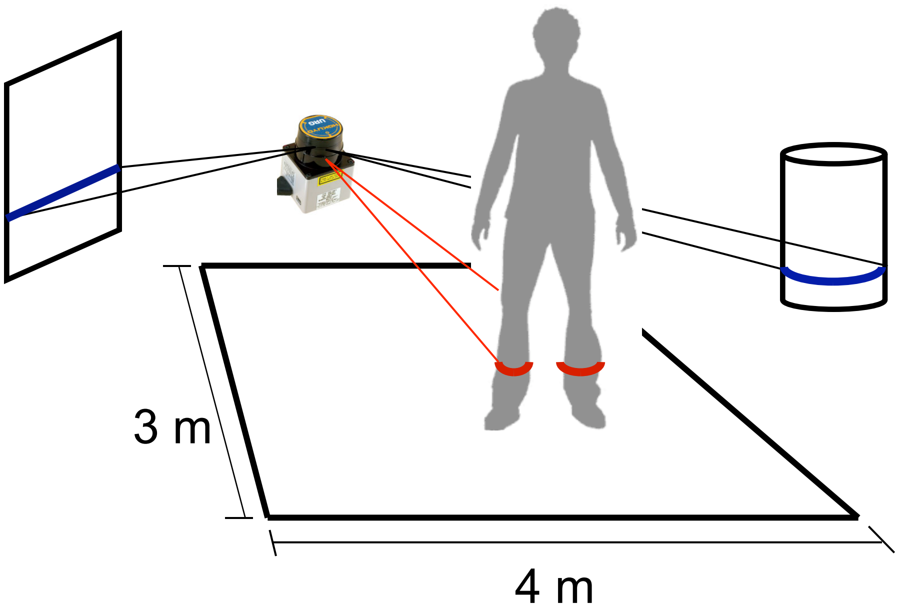
Training: Data Labeling

- **Mark segments** that correspond to people
- Either **manually** or **automatically**



Training: Data Labeling

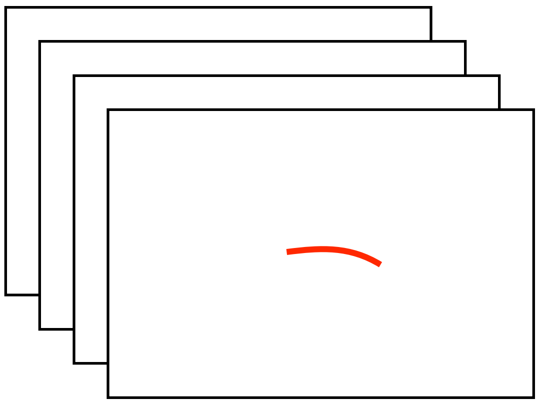
- **Automatic labeling**: obvious approach, define area of interest



- Here: discrimination from background is relevant information, includes spatial relation between fore- and background. Thus: labeling is done **by hand**

Training

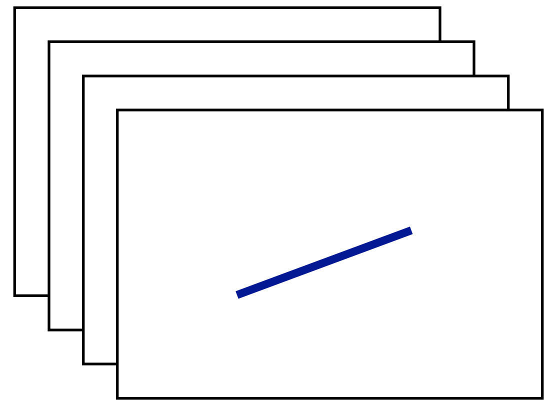
- Resulting **Training Set**



+1

Segments corresponding to people

(foreground segments)

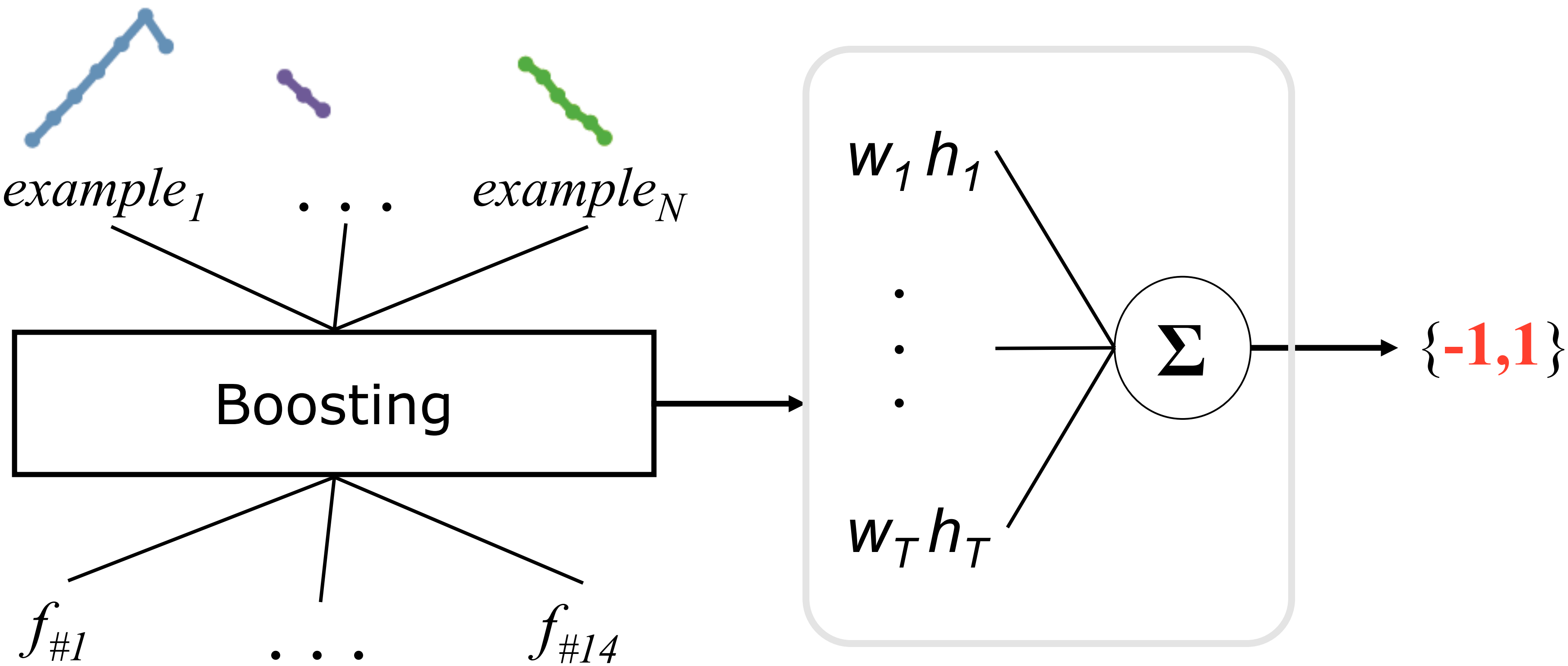


-1

Segments corresponding to other objects

(background segments)

AdaBoost: Final Strong Classifier



Vocabulary of features

Weighted majority vote classifier

$$h_s(\mathbf{x}) = \sum_{t=1}^{T} w_t h_t(\mathbf{x})$$

Experiments

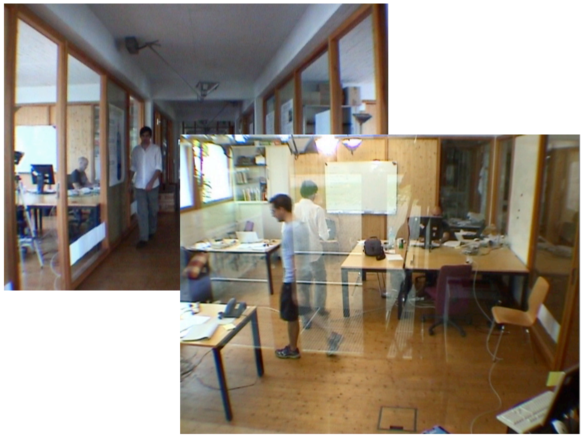
Env. 1: Corridor, no clutter

	Detected Label		
True Label	Person	No Person	Total
Person	**239** (99.58%)	**1** (0.42%)	**240**
No Person	**27** (1.03%)	**2589** (98.97%)	**2616**

Env. 2: Office, very cluttered

	Detected Label		
True Label	Person	No Person	Total
Person	**497** (97.45%)	**13** (2.55%)	**510**
No Person	**171** (2.73%)	**6073** (96.26%)	**6244**

Experiments



Env. 1 & 2: Corridor and Office

	Detected Label		
True Label	Person	No Person	Total
Person	**722** (96.27%)	**28** (3.73%)	**750**
No Person	**225** (2.54%)	**8649** (99.88%)	**8860**

Env. 1→2: Cross-evaluation
Trained in corridor, applied in office

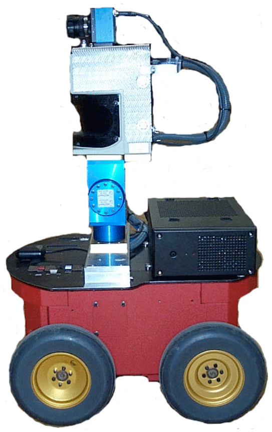
	Detected Label		
True Label	Person	No Person	Total
Person	**217** (90.42%)	**23** (9.58%)	**240**
No Person	**112** (4.28%)	**2504** (95.72%)	**2616**

Experiments

Adding motion feature (mean speed, f#14)

	Without Motion Feature	With Motion Feature
False Negatives (%)	3.73	3.47
False Positives (%)	2.54	3.13
Total Error (%)	2.63	3.15

→ **Motion feature has no contribution**



Experimental setup:

- Robot Herbert
- SICK 2D laser range finder, **1 degree** resolution

Experiments

- Comparison with **hand-tuned classifier**

  - **Jump distance** θ_δ = 30 cm
  - **Width** $\theta_{w,m}$ = 5 cm, $\theta_{w,M}$ = 50 cm
  - **Number of points** θ_n = 4
  - **Standard deviation** θ_σ = 50 cm
  - **Motion of points** θ_v = 2 cm

	Heuristic Approach	AdaBoost
False Negatives (%)	34.67	3.73
False Positives (%)	9.06	2.54
Overall Error (%)	11.06	2.63

People are often not detected

Experiments

Five **best features:**

1: Radius r_c
of LSQ-fitted circle, robust size measure (#9)

2: Mean angular difference
Convexity measure (#13)

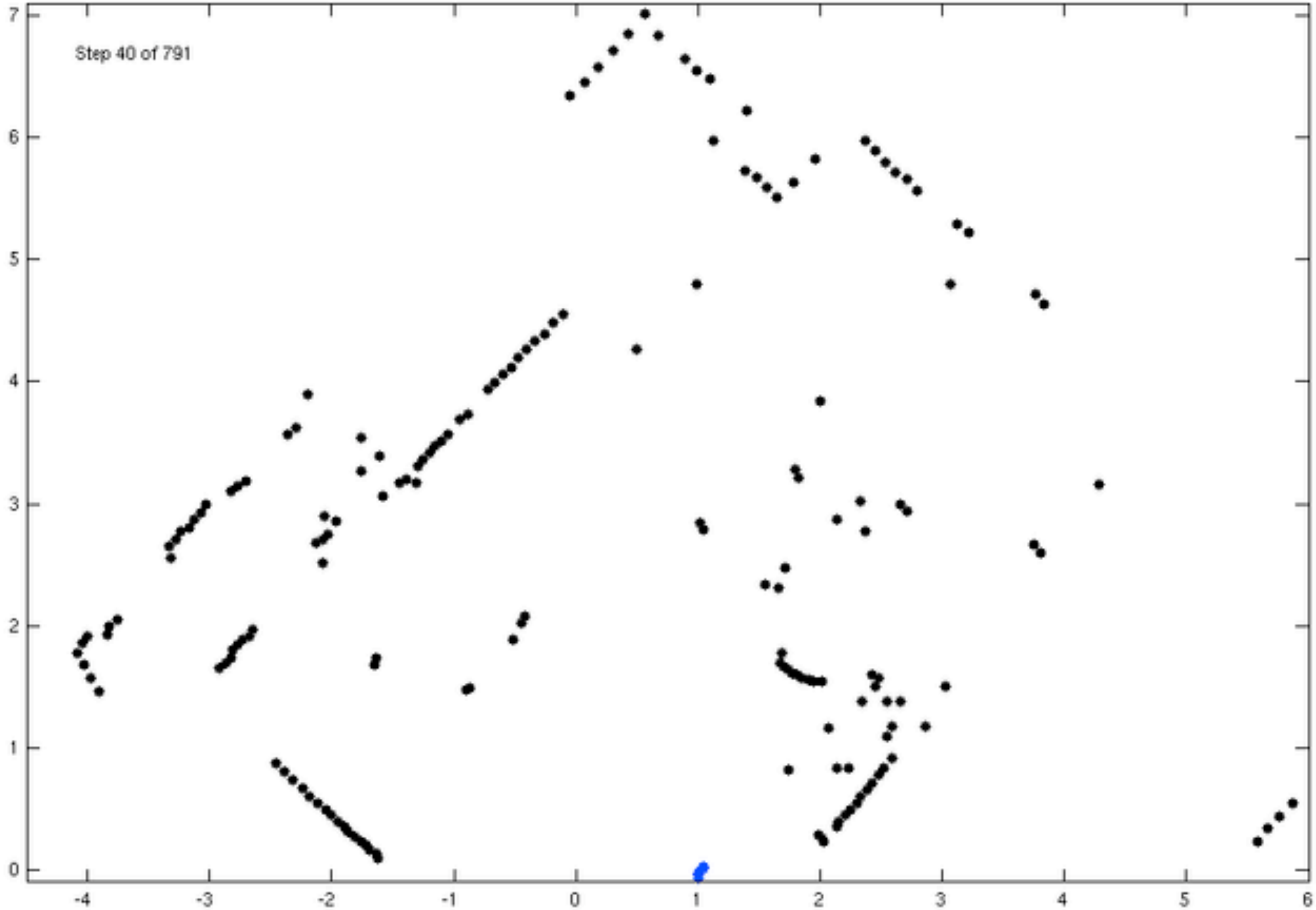
3/4: Jump distances
Local minima measure (#4 and #5)

5: Mad from median
Robust compactness measure (#3)

Environment	Five Best Features
Corridor	9, 4, 5, 2, 4
Office	9, 13, 3, 4, 5
Both	9, 13, 4, 3, 5

Result: Classification

	T	F
T	● (blue)	● (red)
F	● (red)	● (black)



Chapter Contents

- Machine Learning: A Survey
- Classification
- AdaBoost
- People Detection with Boosted Features
- **Place Recognition with Boosted Features**

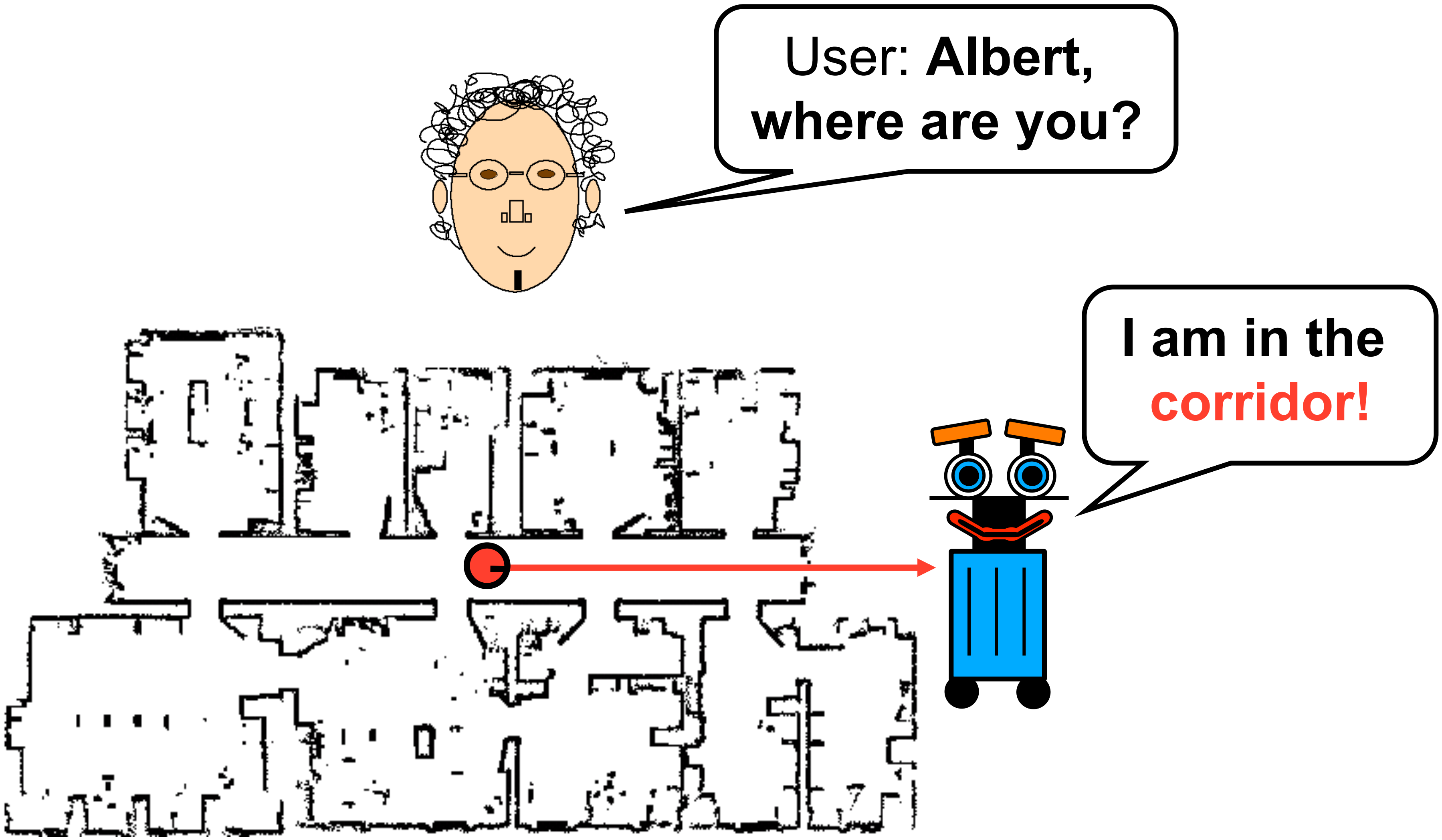
Place Labeling: Motivation

- A map is a **metric** and **topological** model of the environment

Place Labeling: Motivation

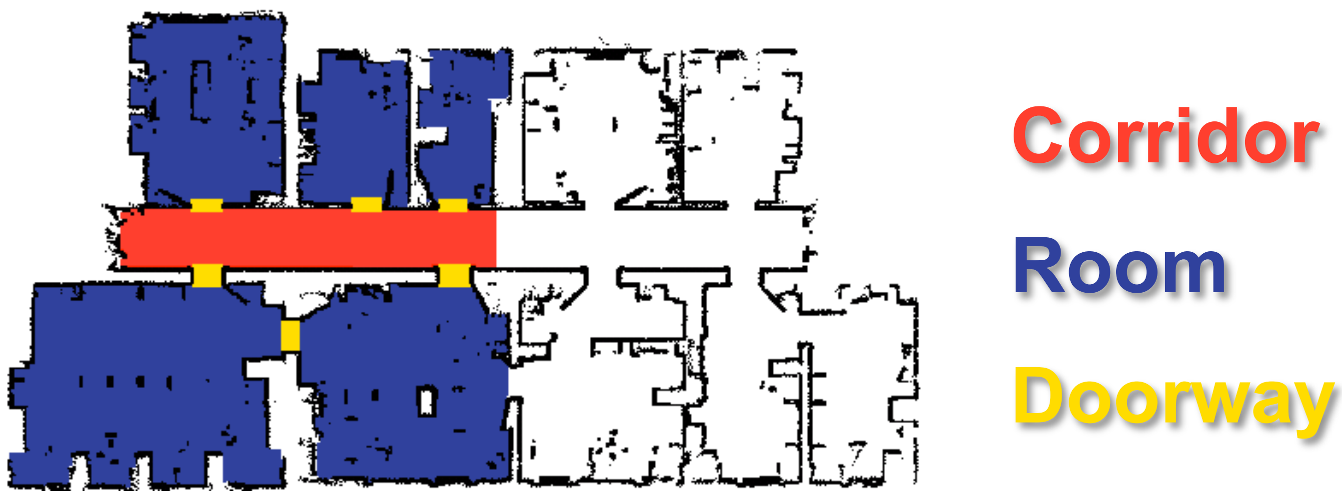
- Wanted: **semantic** information about places

Scenario Example



Scenario Example 2

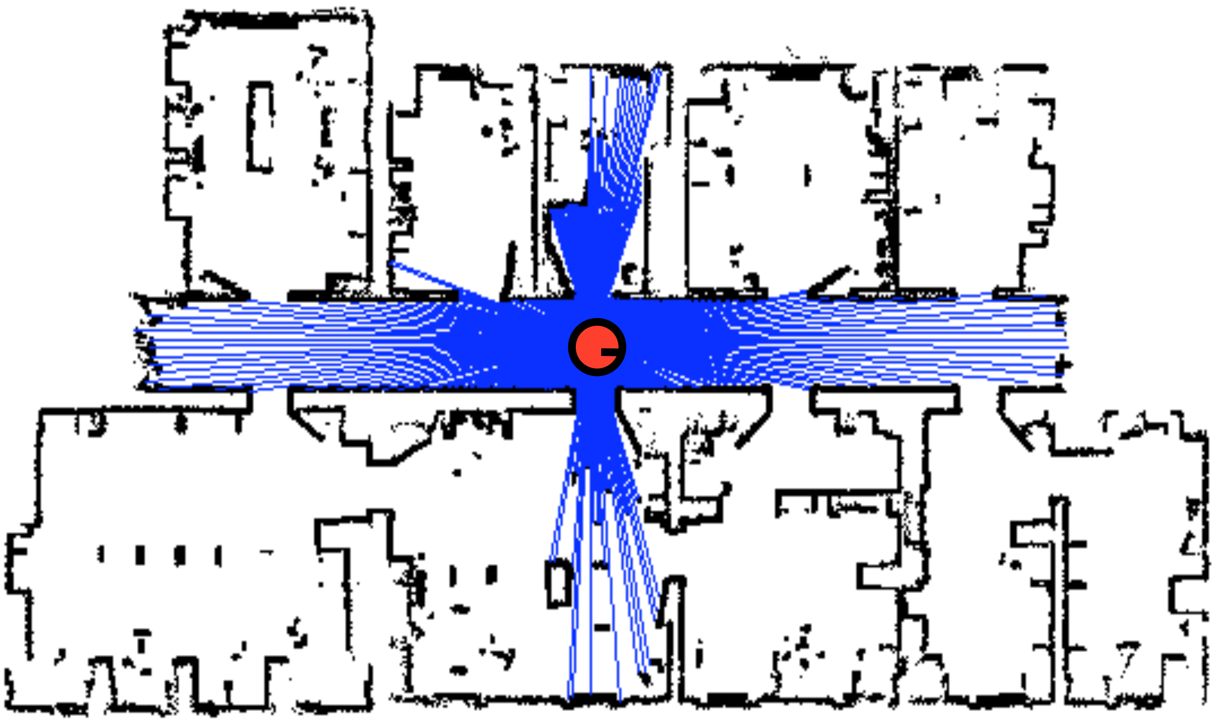
- **Semantic mapping**



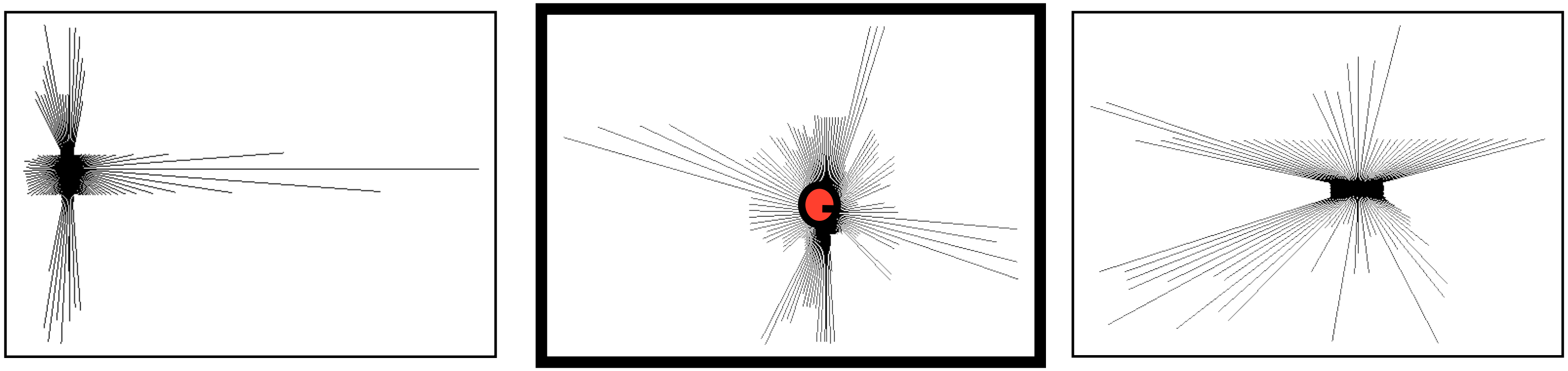
- Human-Robot Interaction of type:
"Robot, get out of my **room**, go into the **corridor**!"

Problem Statement

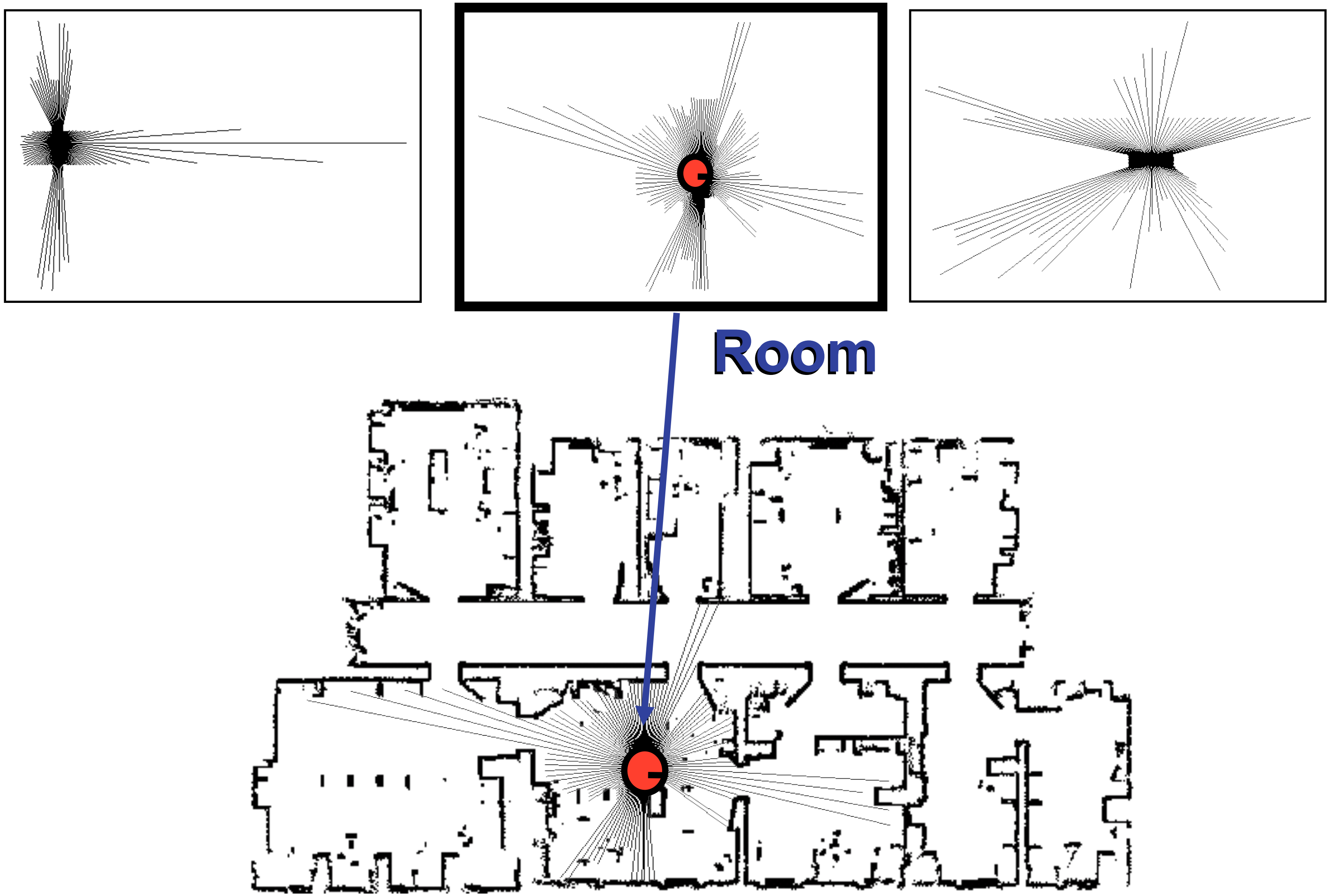
- **Classification of the position** of the robot using a single observation: **a 360° laser range scan**



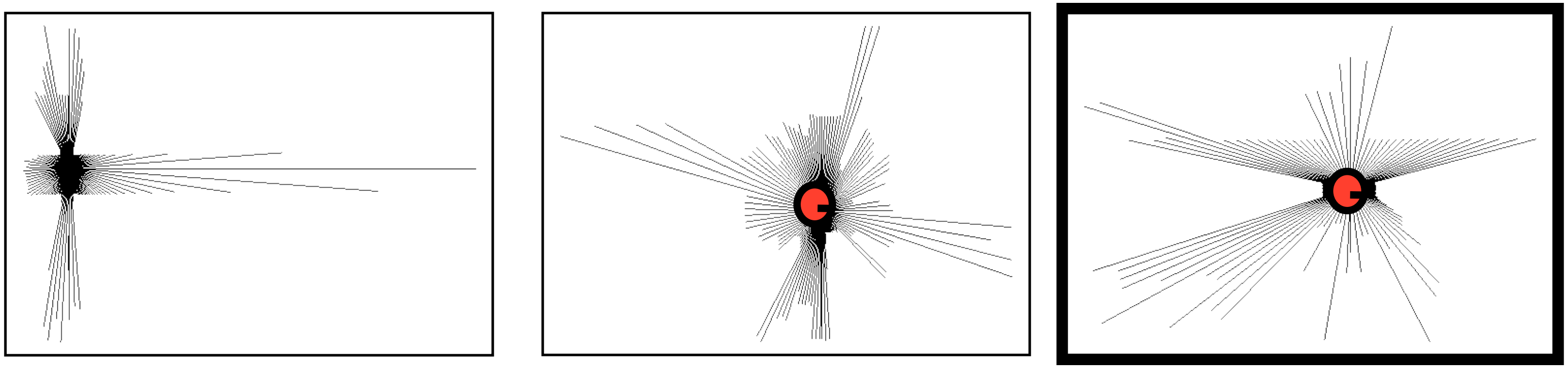
Observations



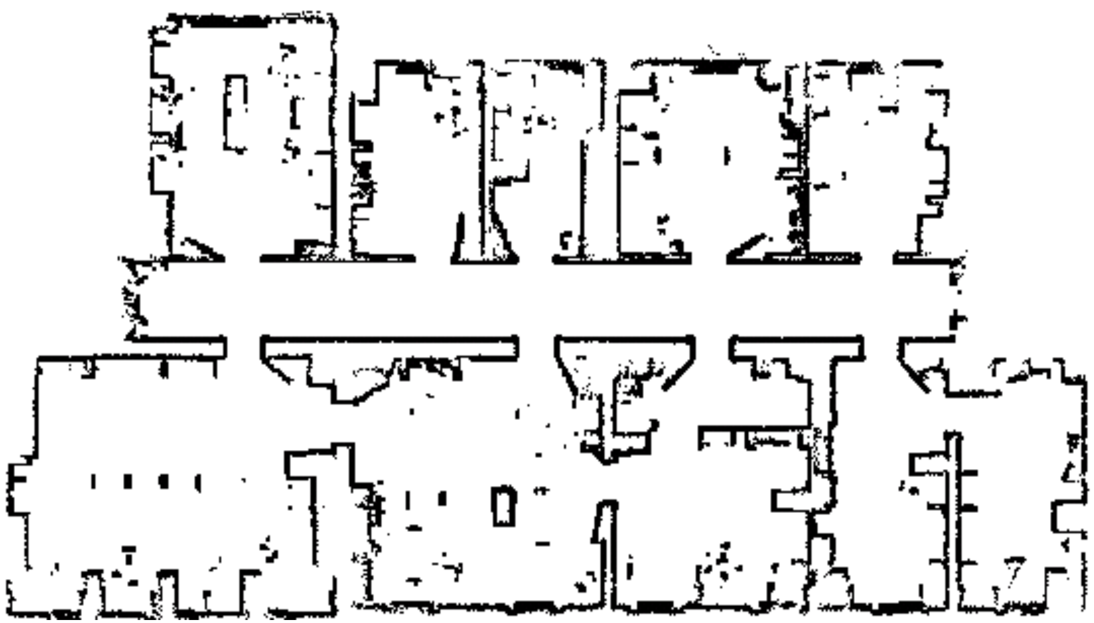
Observations



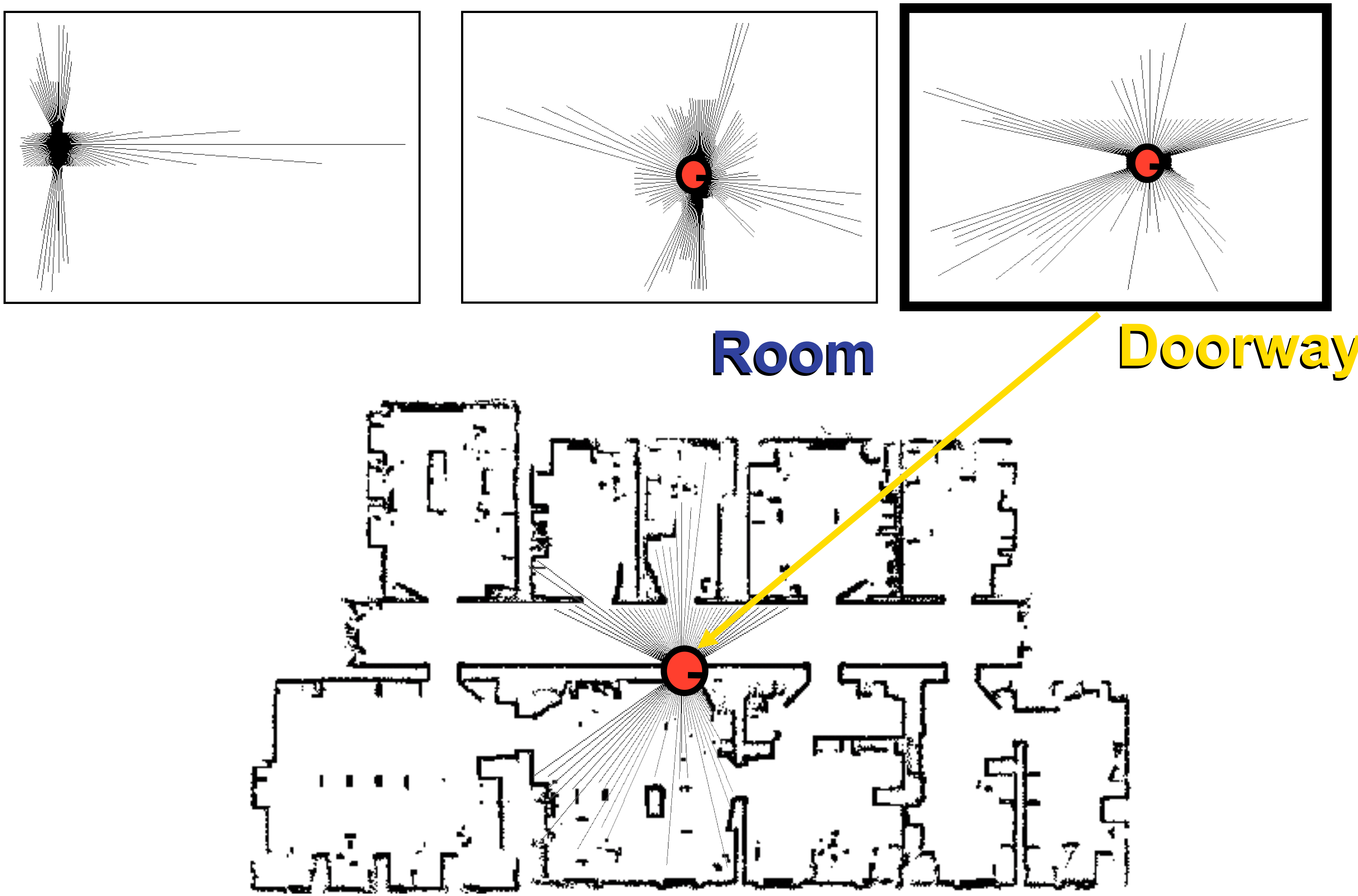
Observations



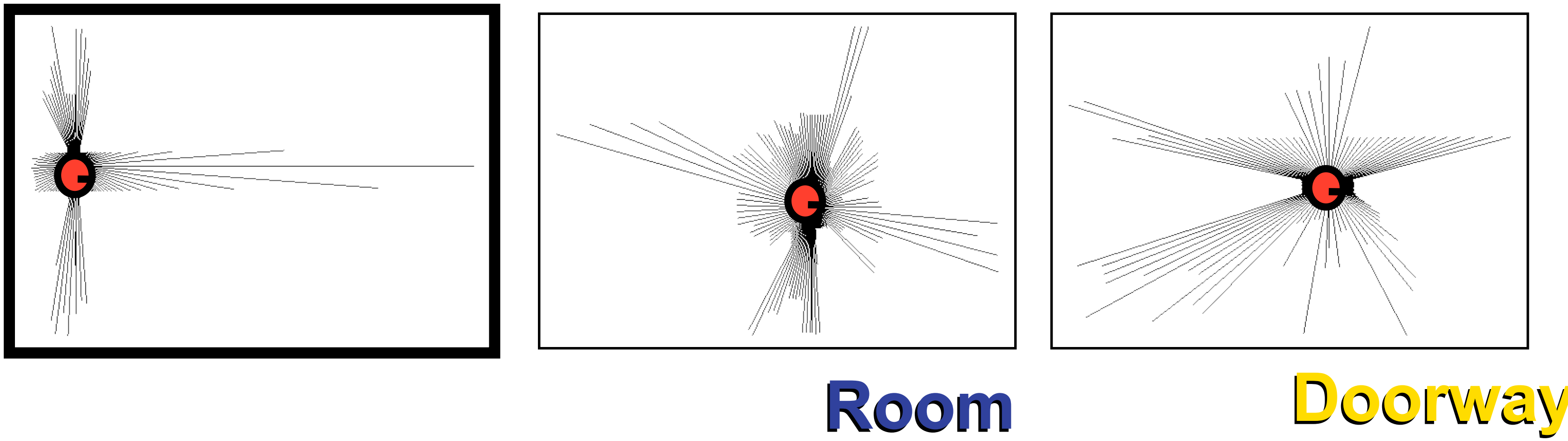
Room

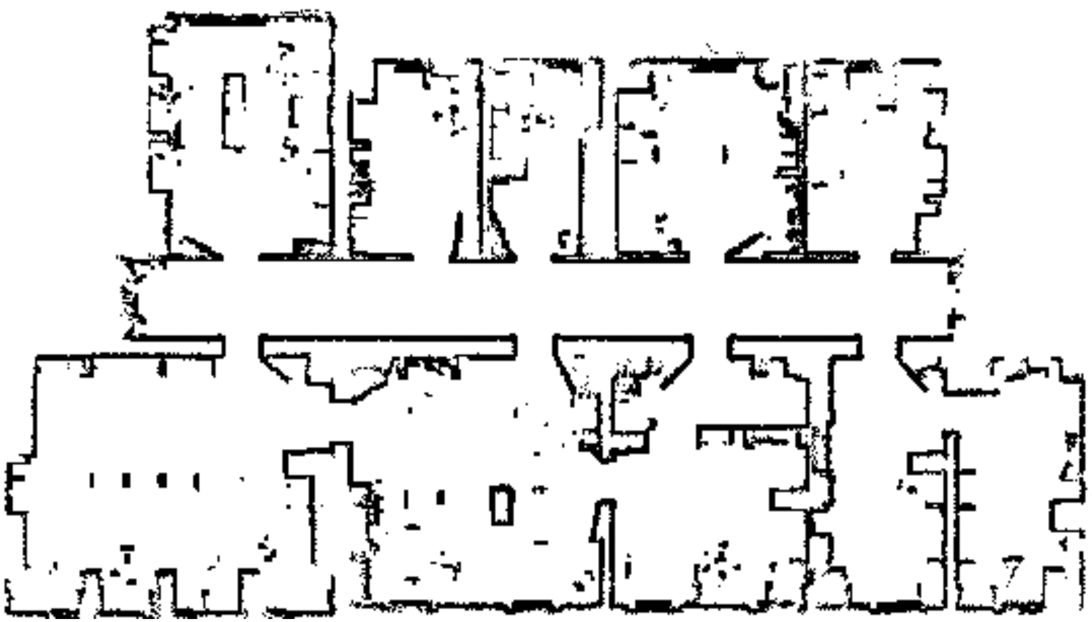


Observations

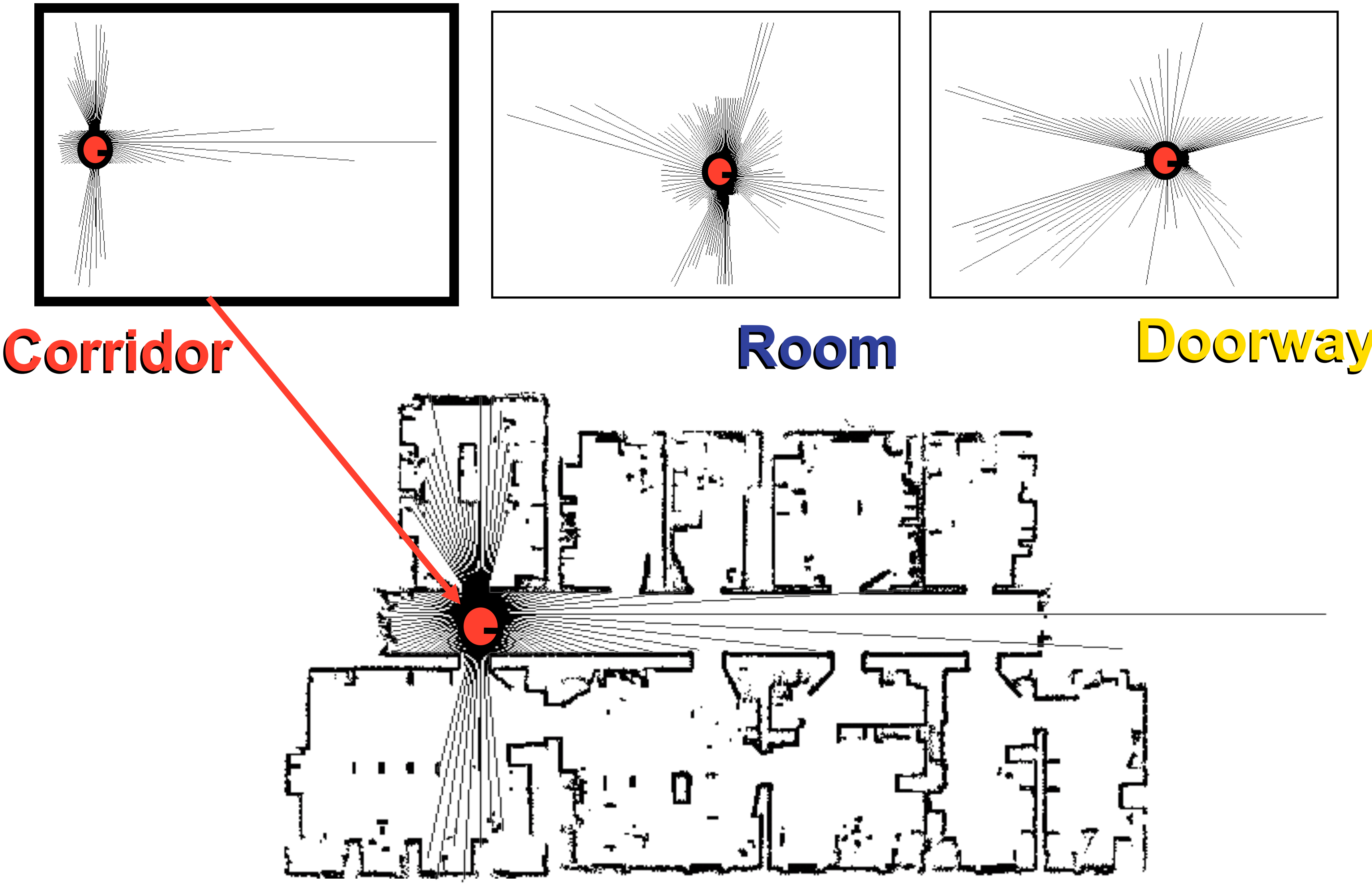


Observations

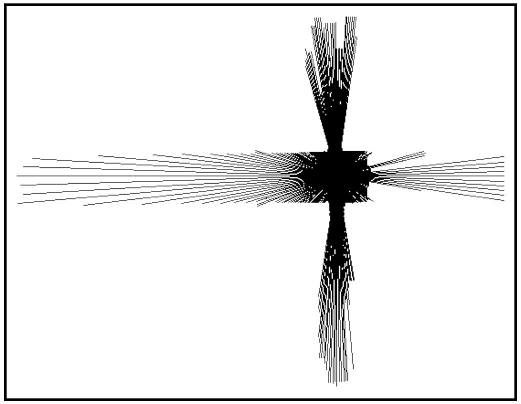


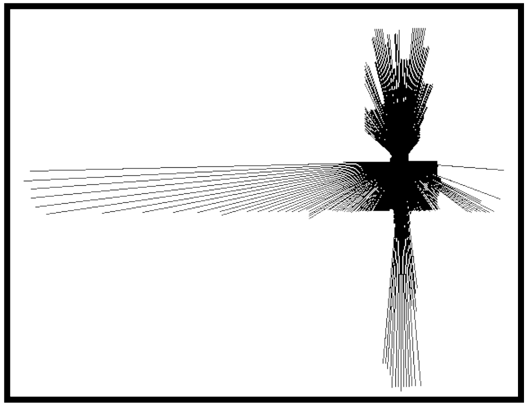


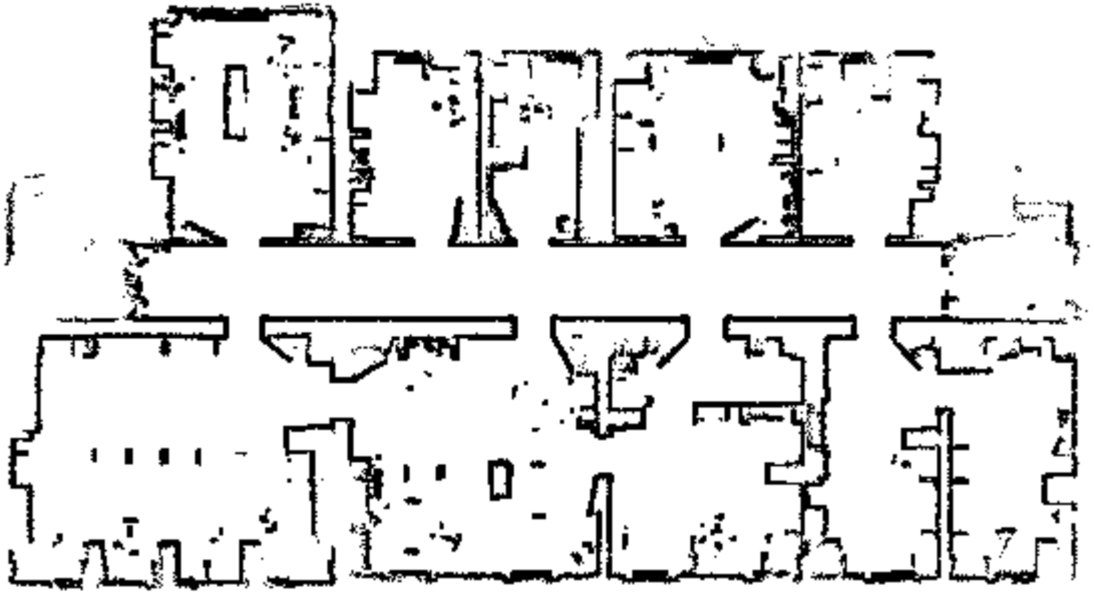
Observations



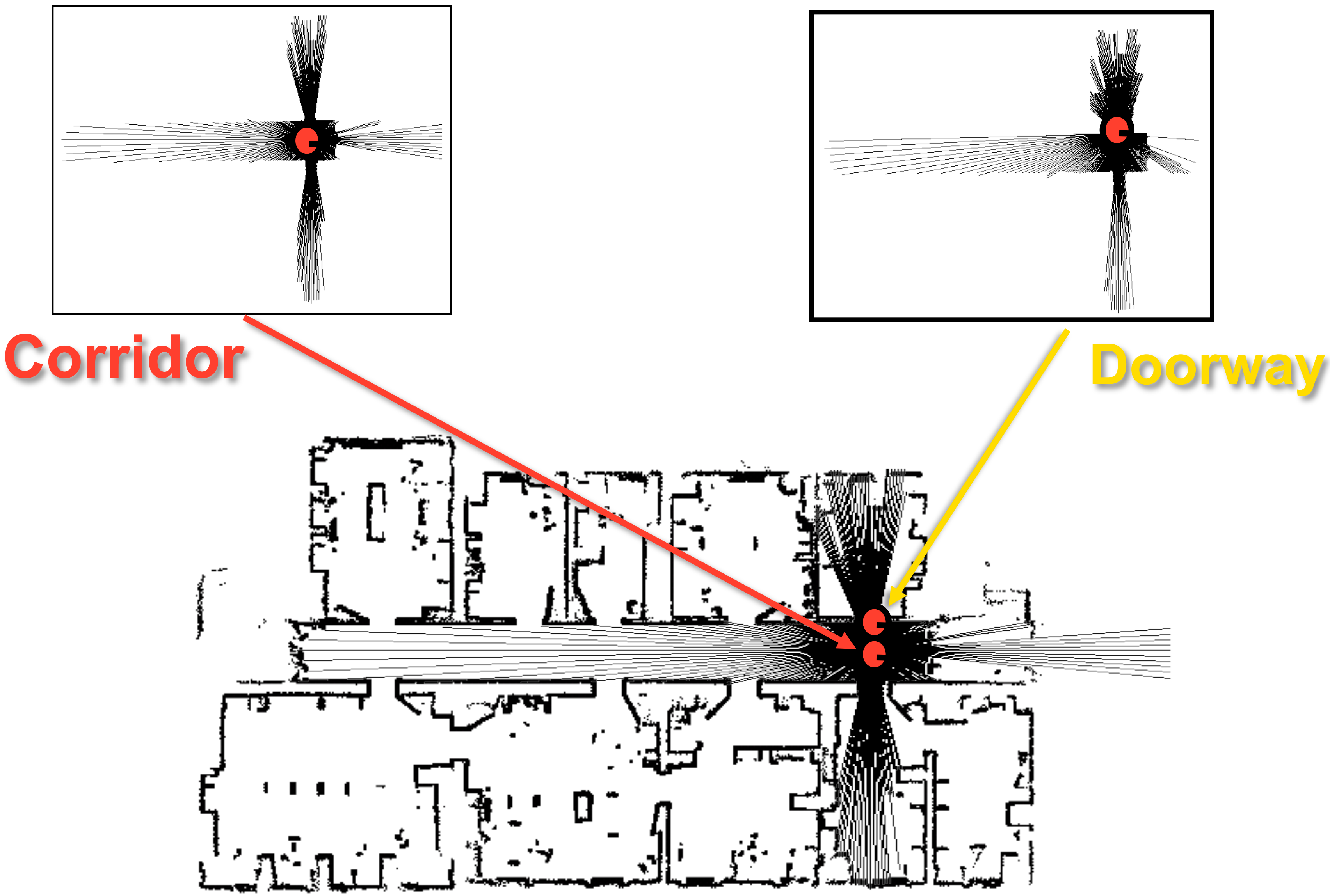
Similar Observations



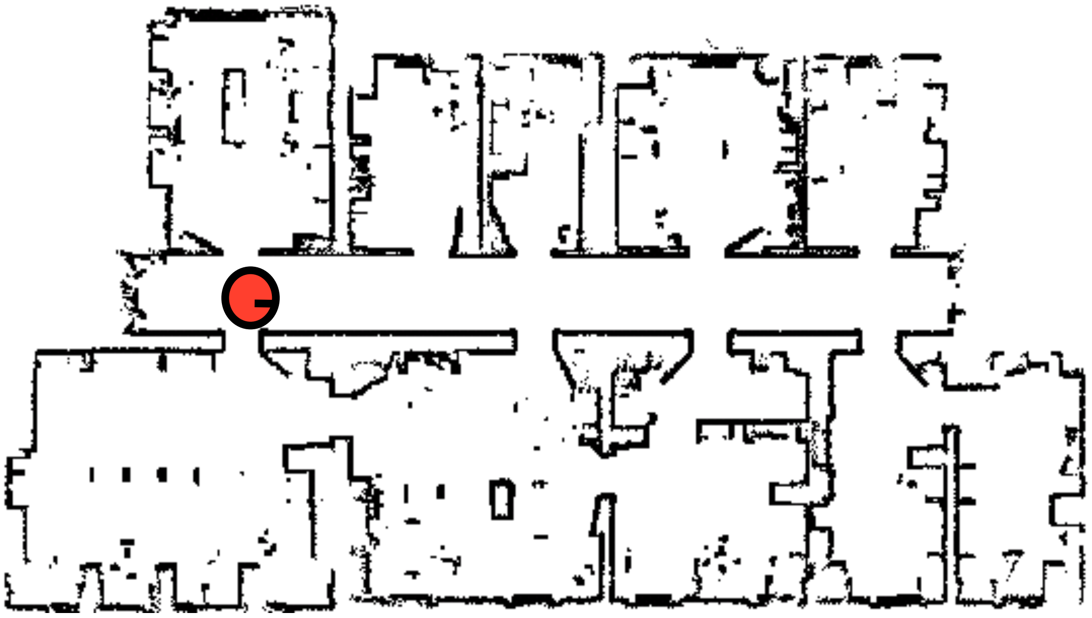




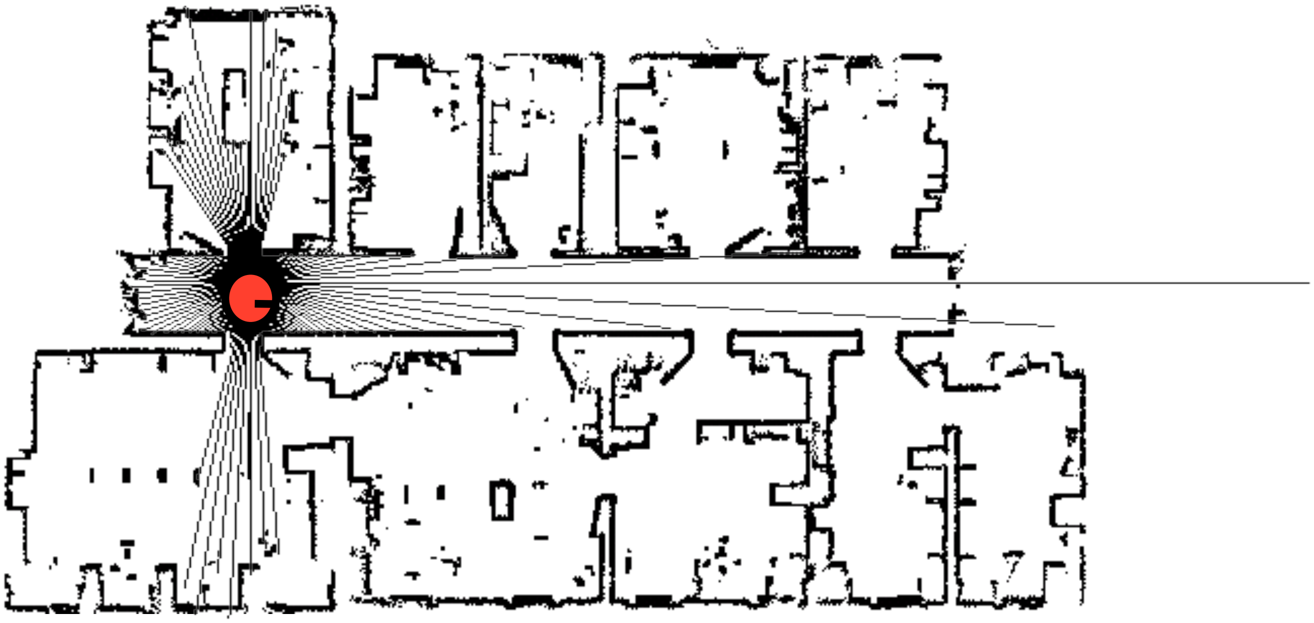
Similar Observations



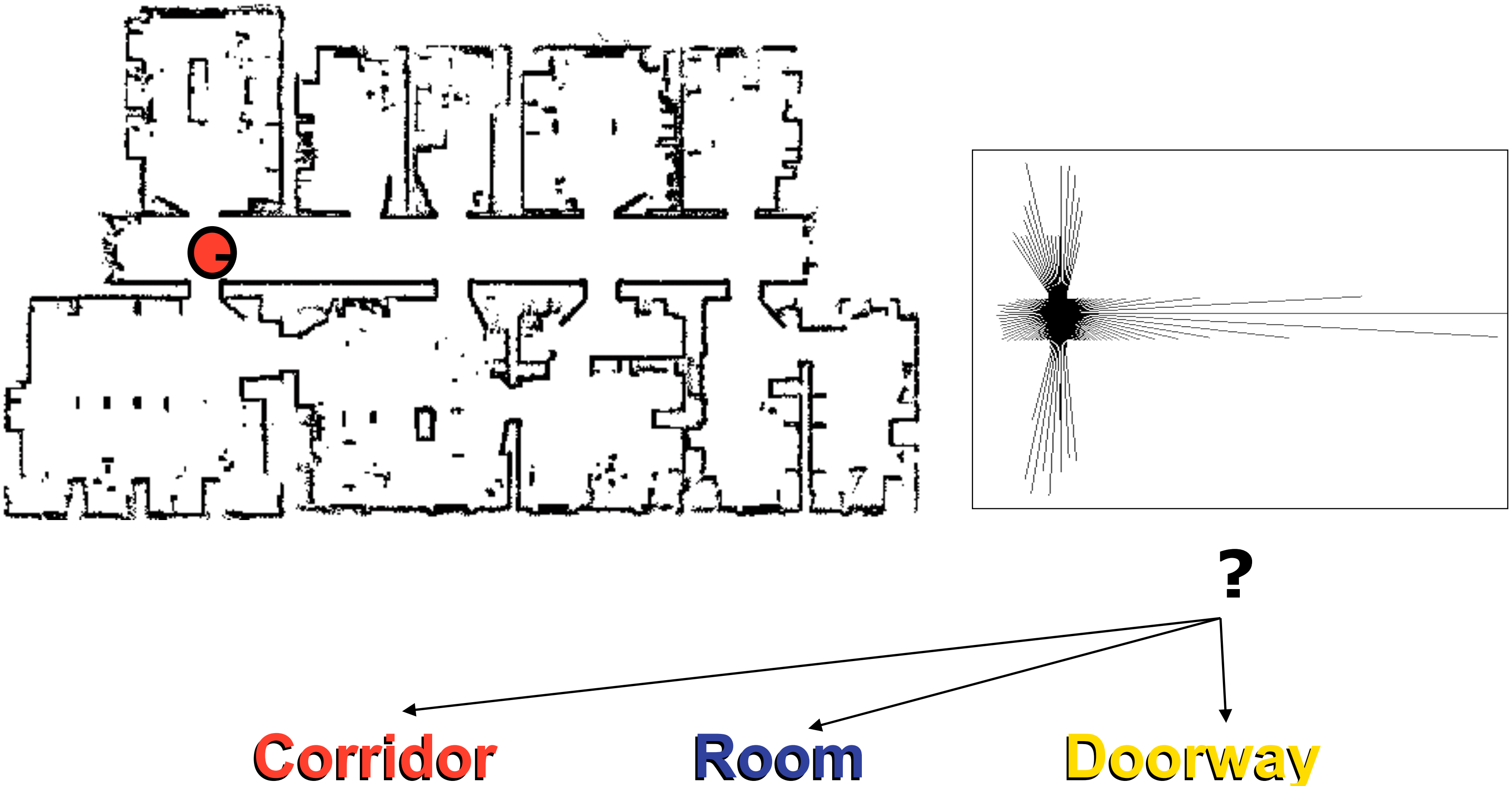
Classification Problem



Classification Problem



Classification Problem

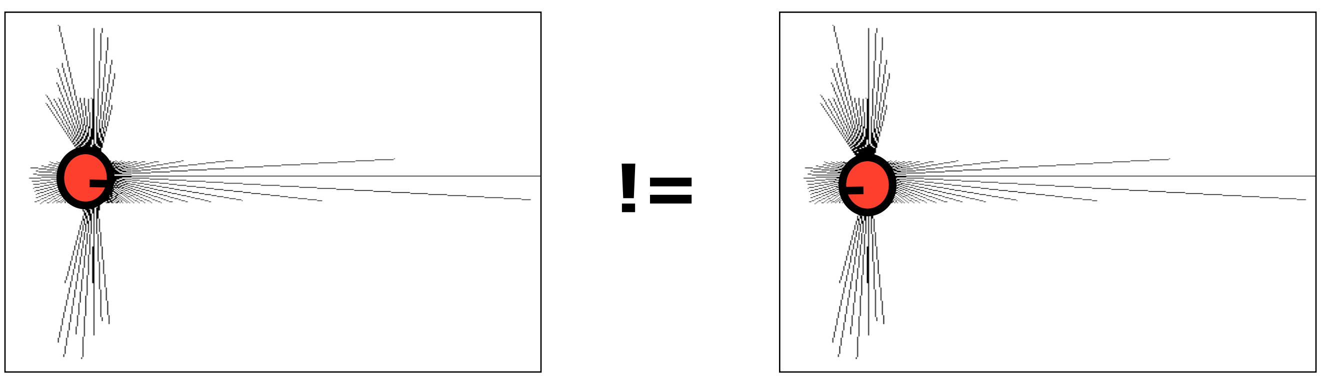


Representing the Observations

- How we represent the 360 laser beams for our classification task?

- As a list of beams $z = \{b_1, b_2, \ldots.., b_M\}$
  Problem: which beam is the first beam?

  Not **invariant to rotation**!



Representing the Observations

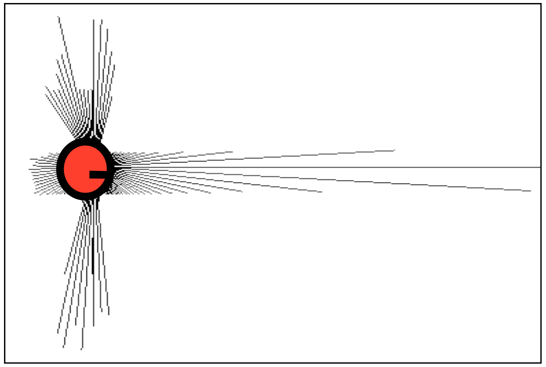
- A list of **scalar geometrical features** of the scan

$$x = \{b_1, b_2, ..., b_M\}$$

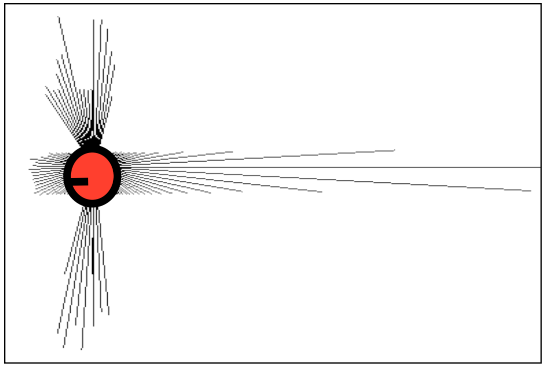
$$\Downarrow$$

$$z = \{f_1, f_2, ..., f_N\} \qquad f_i : f(x) \rightarrow \Re$$

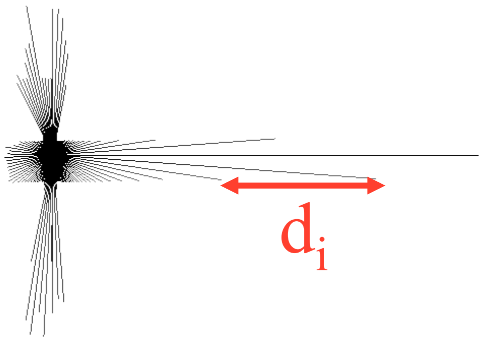
The features are all **invariant to rotation**



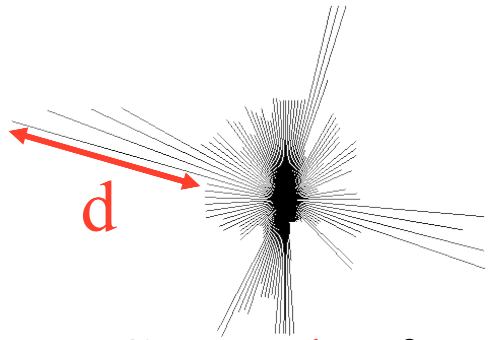
=



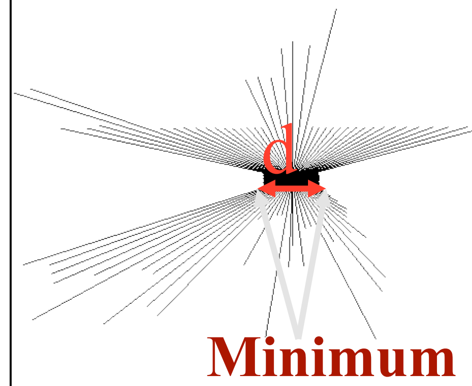
Simple Features



- $f = \frac{1}{N} \sum d_i$

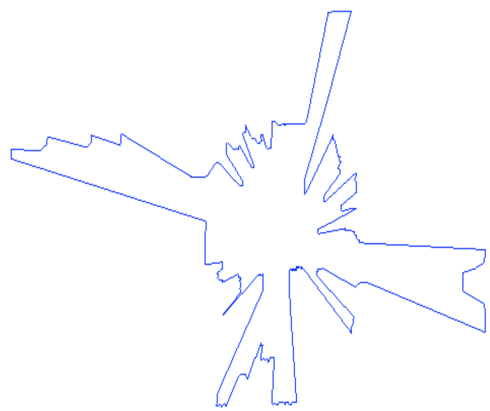


- Gap = $d > \theta$
- f = # Gaps

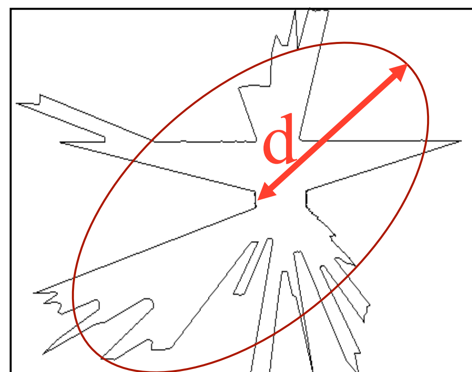


- $f = d$

- f =Area



- f =Perimeter



- $f = d$

Simple Features

- Features of the **raw beams**

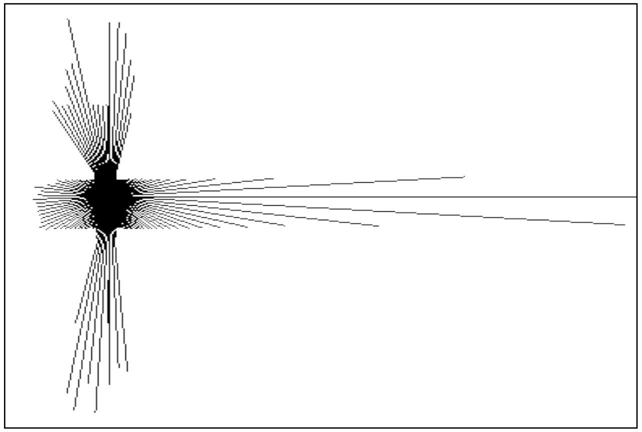
1) The average difference between the length of consecutive beams.
2) The standard deviation of the difference between the length of consecutive beams.
3) Same as 1), but considering different max-range values.
4) The average beam length.
5) The standard deviation of the length of the beams.
6) Number of gaps in the scan. Two consecutive beams build a gap if their difference is greater than a given threshold. Different features are used for different threshold values.
7) Number of beams lying on lines that are extracted from the range scan [16].
8) Euclidean distance between the two points corresponding to the two smallest local minima.
9) The angular distance between the beams corresponding to the local minima in feature 8).

Simple Features

- Features of the **closed polynom** *P(z)* made up by the beams

1) Area of $\mathbf{P}(z)$.
2) Perimeter of $\mathbf{P}(z)$.
3) Area of $\mathbf{P}(z)$ divided by Perimeter of $\mathbf{P}(z)$.
4) Mean distance between the centroid to the shape boundary.
5) Standard deviation of the distances between the centroid to the shape boundary.
6) 200 similarity invariant descriptors based in the Fourier transformation.
7) Major axis Ma of the ellipse that approximates $\mathbf{P}(z)$ using the first two Fourier coefficients.
8) Minor axis Mi of the ellipse that approximate $\mathbf{P}(z)$ using the first two Fourier coefficients.
9) Ma/Mi.
10) Seven invariants calculated from the central moments of $\mathbf{P}(z)$.
11) Normalized feature of compactness of $\mathbf{P}(z)$.
12) Normalized feature of eccentricity of $\mathbf{P}(z)$.
13) Form factor of $\mathbf{P}(z)$.

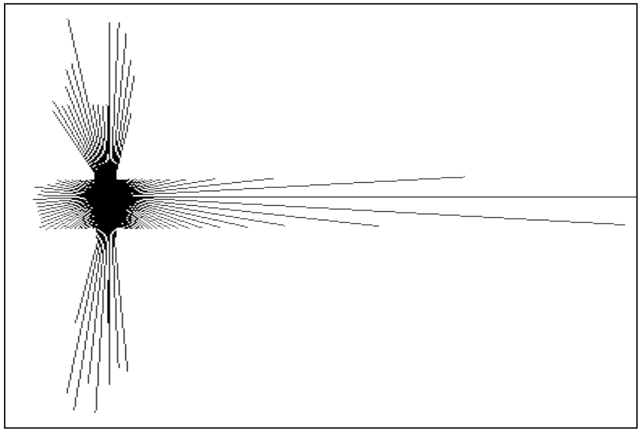
Multiple Classes



$$z = \{f_1, f_2, \ldots.., f_N\}$$

Corridor	Room	Doorway
1	2	3

Multiple Classes



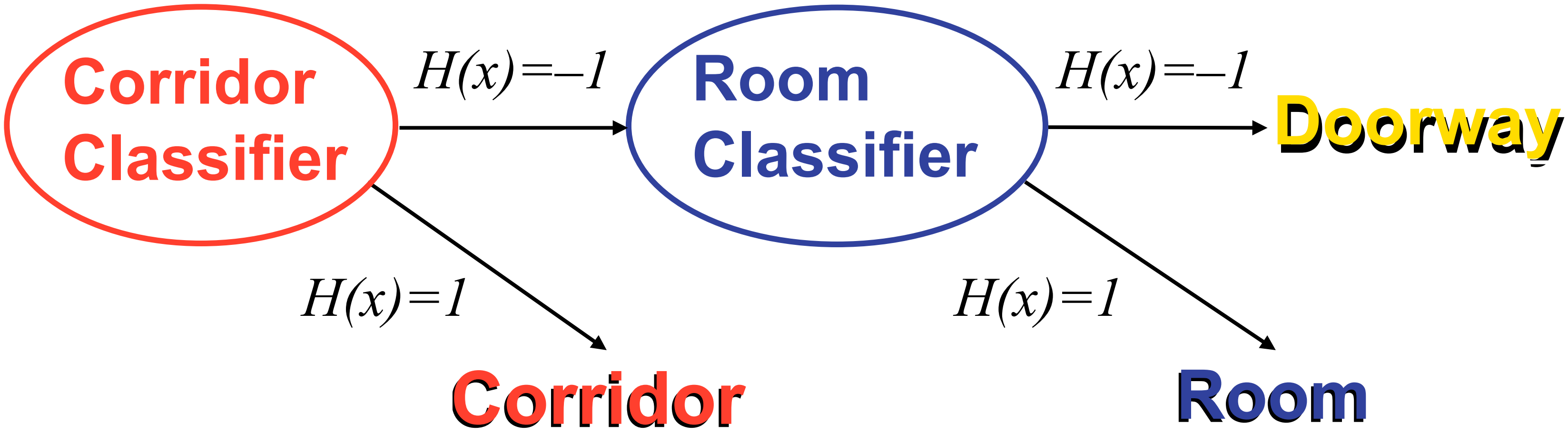
$$z = \{f_1, f_2, ..., f_N\}$$

$$Classifier : A(z) \rightarrow \{1, 2, 3\}$$

Corridor	Room	Doorway
1	2	3

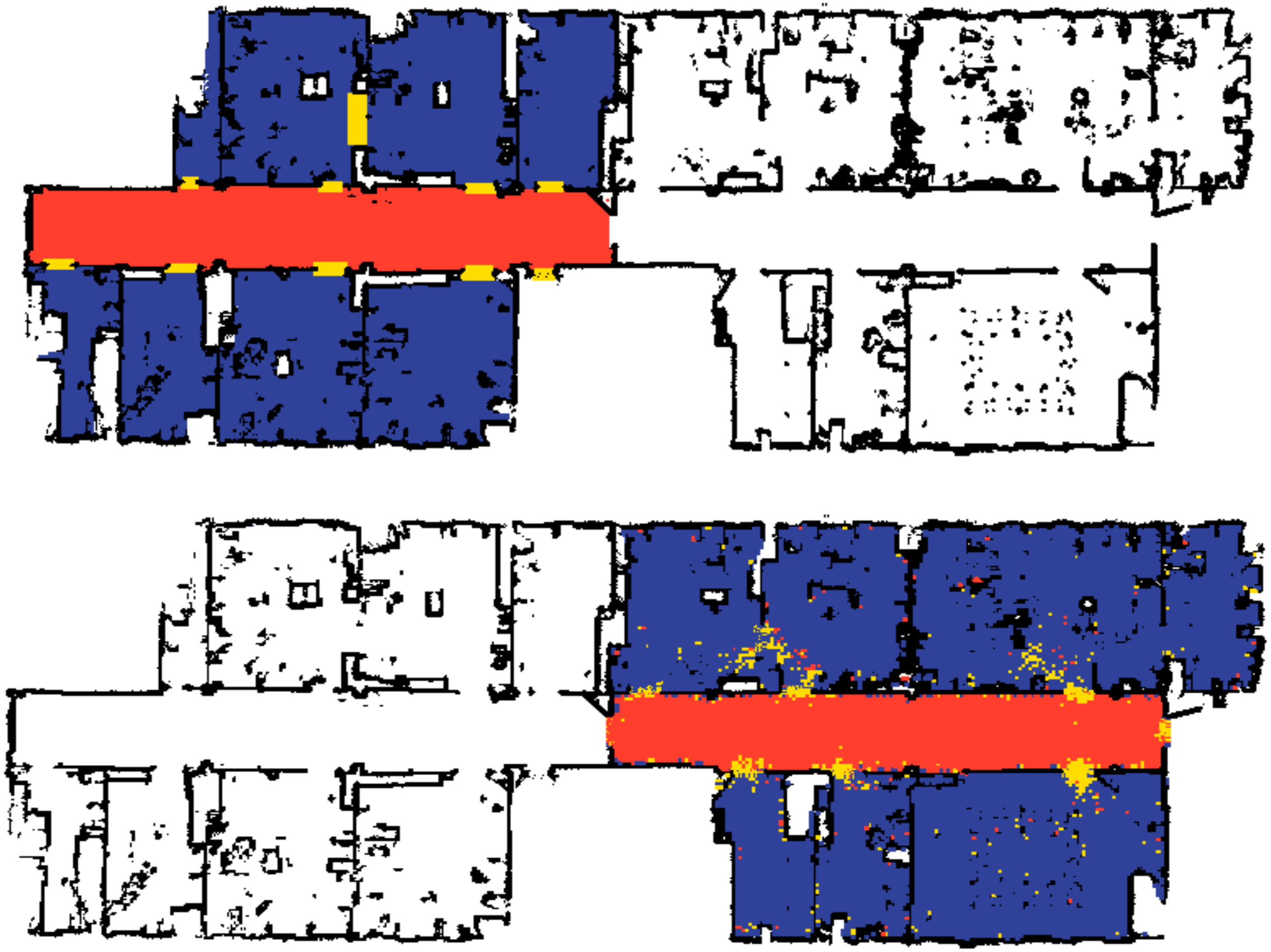
Multiple Classes

- **Sequence of binary classifiers** in a **decision list**



- Alternative to **AdaBoost.M2**, the multi-class variant of AdaBoost
- Order matters, chosen to be according to **error rate**
- One-vs-all learning

Experiments



Training (top)
examples:
16045

Test (bottom)
examples:
18726
classification:
93.94%

Building 079
Uni. Freiburg

Corridor Room Doorway

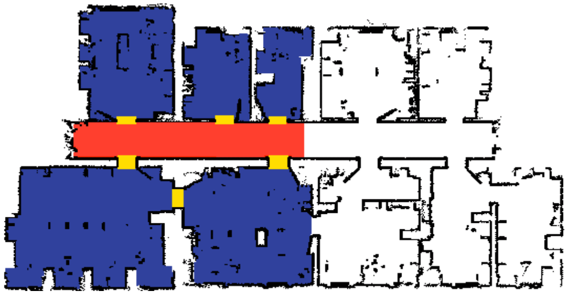
Experiments

Training (left)
examples:
13906

Test (right)
examples:
10445
classification:
89.52%

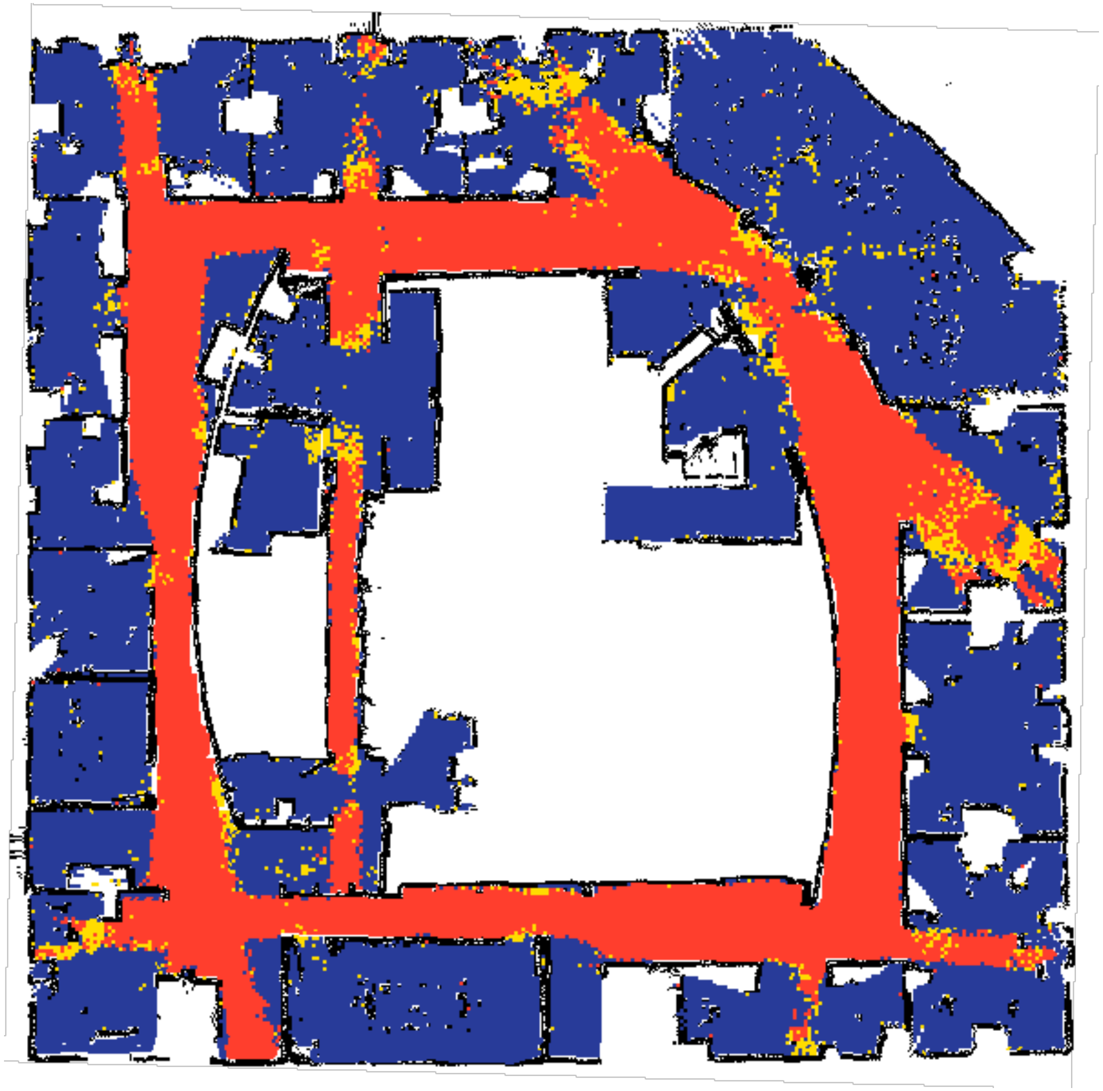
Building 101
Uni. Freiburg

Application to New Environment

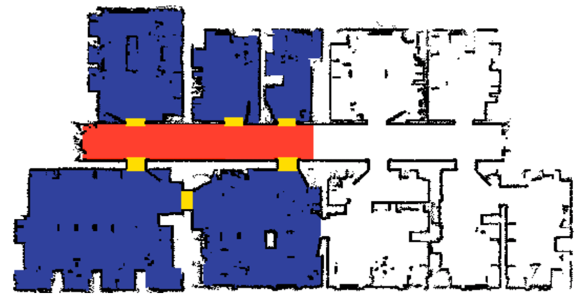

Training map

Intel Research Lab
in Seattle

Application to New Environment



Corridor Room Doorway



Training map

Intel Research Lab
in Seattle

Summary

- **People detection** and **place recognition** phrased as a classification problem using (geometrical and statistical) features that characterize range data (entire scans, groups of neighboring beams)
- **AdaBoost** allows for a **systematic approach** to perform this task
- Both, **single-frame people detection** and **place recognition** with around **90%** accuracy
- Learned classifier clearly **superior** to hand-tuned classifier