# Mobile robotics
# Probabilistic localization and mapping

Janusz Jakubiak

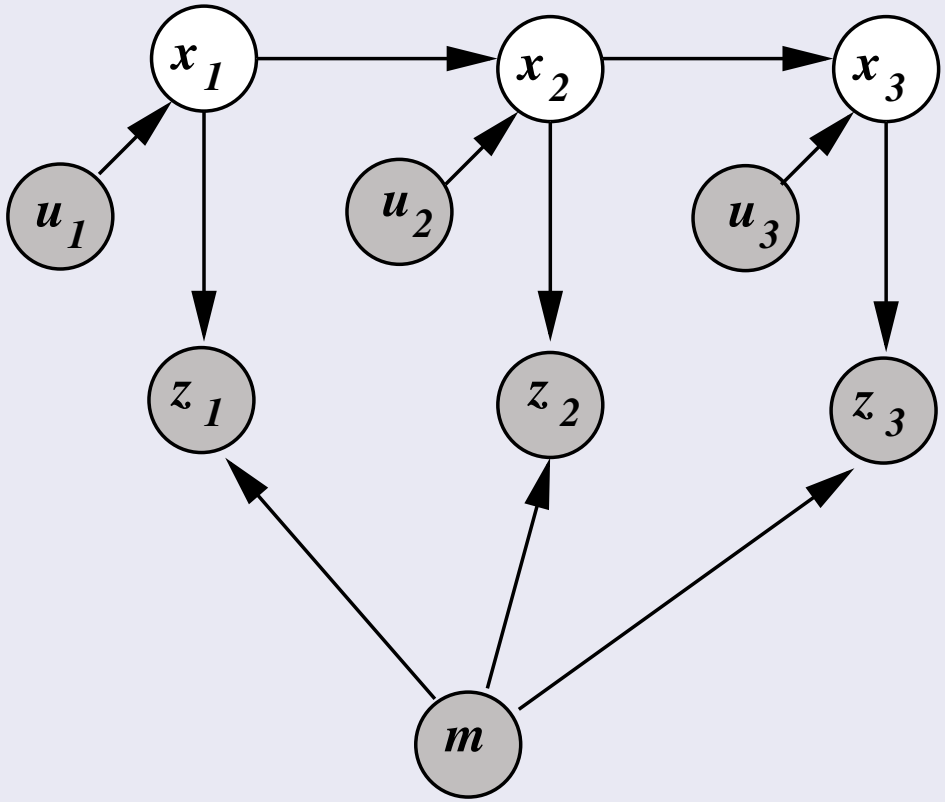
Wrocław, 20.11.2018

# Copyright information

Following slides are a supporting material for the course AREA00100. Most of included notes and illustrations are copyrighted by their authors or publishers. Please respect that copyright by using the notes only for purposes of assigned reading in this class.

These slides contents base on a book Probabilistic Robotics (S. Thurn et al. )

# Robot localization task – definition

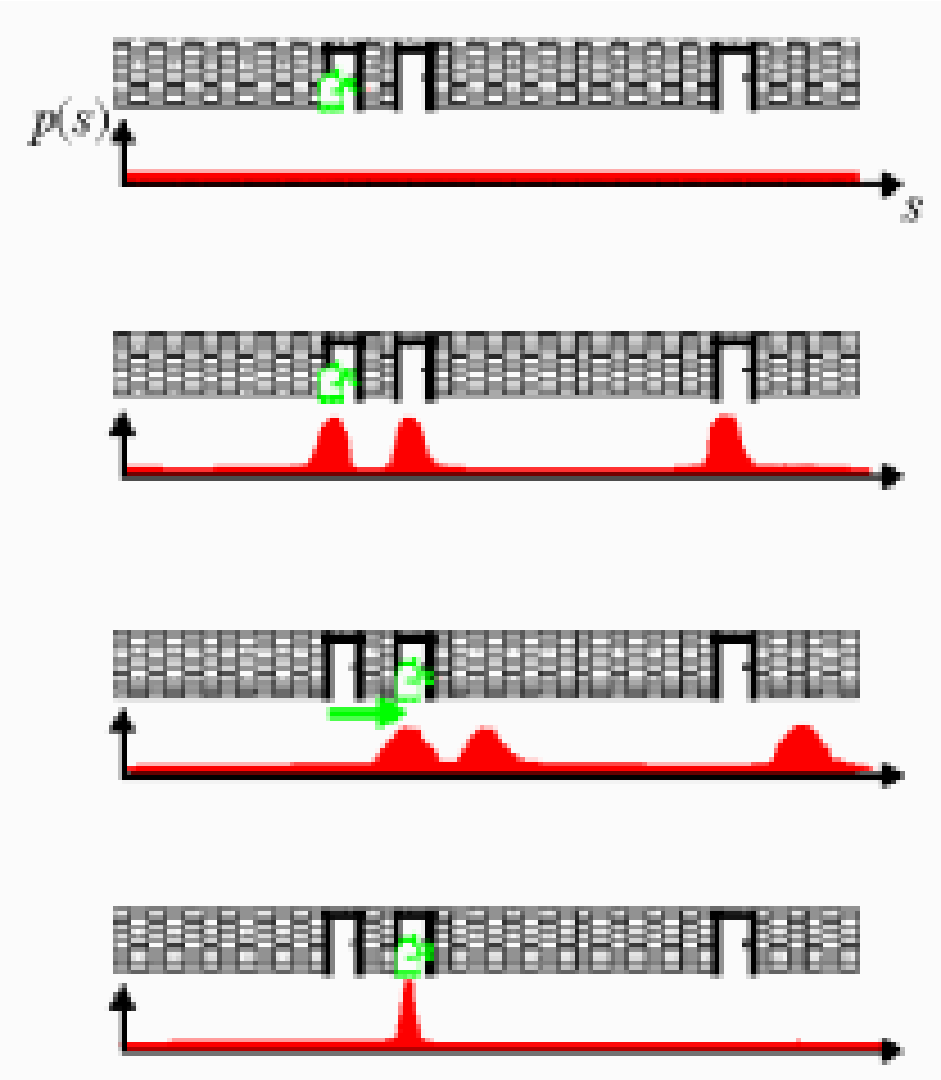



Having known: environment model $m$, controls $\mathcal{U} = \{u_{1:t}\}$ and observations $\mathcal{Z} = \{z_{1:t}\}$ determine $x_t$ (or $\mathcal{X} = \{x_{1:t}\}$).

# Localization task taxonomy

- local (position tracking) / global (incl. kidnapped robot problem)
- static/dynamic environment
- passive/active
- single-/multi-robot

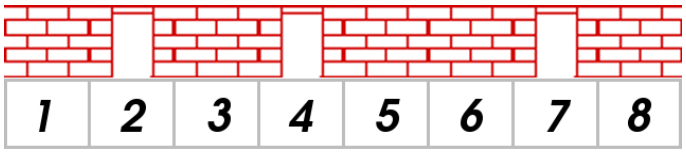
# Probabilistic localization – the idea




[0] http://www.cs.cmu.edu/ thrun/tutorial/sld027.htm

# Example

$$p(z_t = \text{door} \mid x_t = x_i) = \begin{cases} 1 & \text{for } x_i = 2, 4, 7, \\ 0 & \text{for } x_i = 1, 3, 5, 6, 8 \end{cases}$$




$$x_{t+1} = f(x_t, u_t = 1) = x_t + 1$$

$$Z = \{\text{door}, \text{wall}, \text{door}\}$$

$p(x_t = x_i | z_{1:t})$ :

| $i$, $z_i$ \ $x_i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 0 | 1/8 | 1/8 | 1/8 | 1/8 | 1/8 | 1/8 | 1/8 | 1/8 |
| 1 (door) | 0 | 1/3 | 0 | 1/3 | 0 | 0 | 1/3 | 0 |
| 2 (wall) | 0 | 0 | 1/3 | 0 | 1/3 | 0 | 0 | 1/3 |
| 3 (door) | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

# Markov model localization

parameters: $bel(x_{t-1}), u_t, z_t, m$

for all $x_t$ do

prediction phase: expected probability of the new state
$\overline{bel}(x_t) = \int p(x_t|u_t, x_{t-1}, m) bel(x_{t-1}) dx_{t-1}$

correction phase: estimate update based on observation
$bel(x_t) = \eta p(z_t|x_t, m) \overline{bel}(x_t)$

output: $bel(x_t)$

# Example – cont.

## Remark

in most cases: $p(z_t|x_t)$ and $p(x_{t+1}|x_t, u_t)$ are not binary, so the state change and output may suffer from noise.

## Questions

How would the solution change if

1. if $u_i = 0.9$, so the transfer function is given by

$$p(x_{t+1} = x_i + 1|x_t = x_i) = 0.9 \qquad p(x_{t+1} = x_i|x_t = x_i) = 0.1$$

2. the probability that door are open is 0.5 and the robot is unable to distinguish between a closed door and a wall?

(recall Viterbi algorithm)

# EKF localization

EKF localization is a special case of Markov model localization

## Assumptions

1. beliefs (*bel*) are represented by moments: $(\mu_t, \Sigma_t)$
2. map is represented by a collection of features $z_t = \{z_t^1, z_t^2, \ldots\}$
3. $(+)$ the features are identifiable by a correspondence $c_t = \{c_t^1, c_t^2, \ldots\}$

## Properties

- Gaussian is unimodal – suitable for position tracking and cases when the approximate pose may be restricted to a small region
- It does not allow hard constraints (like walls)
- Linearization is usually valid only close to the real pose
- It does not interpret negative information (missing features)

# General EKF algorithm

prediction phase:

$$\bar{\mu}_t = g(u_t, \mu_{t-1})$$
$$\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$$

observation phase:

$$K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1}$$

$$\mu_t = \bar{\mu}_t + K_t(z_t - h_t(\bar{\mu}_t))$$
$$\Sigma_t = (I - K_t H_t)\bar{\Sigma}_t$$

# EKF localization algorithm

Unicycle robot equation in discrete form

$$x_t = x_{t-1} + g(x_{t-1}, u_t) + \mathcal{N}(0, R_t)$$

$$\begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} + \begin{pmatrix} -\frac{v_t}{\omega_t}\sin\theta + \frac{v_t}{\omega_t}\sin(\theta + \omega_t \Delta t) \\ \frac{v_t}{\omega_t}\cos\theta - \frac{v_t}{\omega_t}\cos(\theta + \omega_t \Delta t) \\ \omega_t \Delta t \end{pmatrix}$$

Controls with Gaussian noise

$$\begin{pmatrix} \hat{v} \\ \hat{\omega} \end{pmatrix} = \begin{pmatrix} v \\ \omega \end{pmatrix} + \mathcal{N}(0, M_t) = \begin{pmatrix} v \\ \omega \end{pmatrix} + \begin{pmatrix} \varepsilon(\alpha_1 v_t^2 + \alpha_2 \omega_t^2) \\ \varepsilon(\alpha_3 v_t^2 + \alpha_4 \omega_t^2) \end{pmatrix}$$

# EKF localization algorithm – prediction phase

Auxiliary data calculated from the model:

$$\theta = \mu_{t-1,\theta}$$

$$G_t = \frac{\partial g(\mu_{t-1}, u_t)}{\partial \mu_{t-1}} = \begin{pmatrix} 1 & 0 & -\frac{v_t}{\omega_t}\cos\theta + \frac{v_t}{\omega_t}\cos(\theta + \omega_t \Delta t) \\ 0 & 1 & -\frac{v_t}{\omega_t}\sin\theta - \frac{v_t}{\omega_t}\sin(\theta + \omega_t \Delta t) \\ 0 & 0 & 1 \end{pmatrix}$$

$$V_t = \frac{\partial g(\mu_{t-1}, u_t)}{\partial u_t} = \begin{pmatrix} \frac{\partial x'}{\partial v_t} & \frac{\partial x'}{\partial \omega_t} \\ \frac{\partial y'}{\partial v_t} & \frac{\partial y'}{\partial \omega_t} \\ \frac{\partial \theta'}{\partial v_t} & \frac{\partial \theta'}{\partial \omega_t} \end{pmatrix}$$

$$M_t = \begin{pmatrix} \alpha_1 v_t^2 + \alpha_2 \omega_t^2 & 0 \\ 0 & \alpha_3 v_t^2 + \alpha_4 \omega_t^2 \end{pmatrix}$$

Update rule:

$$\bar{\mu}_t = \mu_{t-1} + g(\mu_{t-1}, u_t)$$

$$\bar{\Sigma}_t = G_t \Sigma_t G_t^T + V_t M_t V_t^T$$

# EKF localization algorithm – observation phase

Observation model for an observation $z_t^j = (r_t^j, \phi_t^j, s_t^j)$

$$z_t^j = h(x_t, j, m) + \mathcal{N}(0, Q_t)$$

$$\begin{pmatrix} r_t^j \\ \phi_t^j \\ s_t^j \end{pmatrix} = \begin{pmatrix} \sqrt{(m_{j,x} - x_{t,x})^2 + (m_{j,y} - x_{t,y})^2} \\ \operatorname{atan2}(m_{j,y} - x_{t,x}, m_{j,x} - x_{t,x}) - x_{t,\theta} \\ m_{j,s} \end{pmatrix} + \mathcal{N}(0, Q_t)$$

with $Q_t = \operatorname{diag}\{\sigma_r, \sigma_\phi, \sigma_s\}$ and

$$H_t^j = \frac{\partial h(\hat{\mu}_t, j, m)}{\partial x_t} = \begin{pmatrix} -\frac{m_{j,x} - \hat{\mu}_{t,x}}{h_3} & -\frac{m_{j,y} - \hat{\mu}_{t,y}}{h_3} & 0 \\ \frac{m_{j,y} - \hat{\mu}_{t,y}}{h_3^2} & -\frac{m_{j,x} - \hat{\mu}_{t,x}}{h_3^2} & -1 \\ 0 & 0 & 0 \end{pmatrix}, \quad \text{where} \quad h_3 = h(\hat{\mu}_t, j, m)_r$$

# EKF localization algorithm – observation phase

For each observation $z_t^j = (r_t^j, \phi_t^j, s_t^j)$

$$S_t^j = H_t^j \bar{\Sigma}_t (H_t^j)^T + Q_t$$

$$K_t^j = \bar{\Sigma}_t (H_t^j)^T (\bar{\Sigma}_t)^{-1}$$

$$\bar{\mu}_t = \bar{\mu}_t + K_t^j (z_t^j - \hat{z}_t^j)$$

$$\bar{\Sigma}_t = (I - K_t^j H_t^j) \bar{\Sigma}_t$$

Finally:

$$\mu_t = \bar{\mu}_t$$

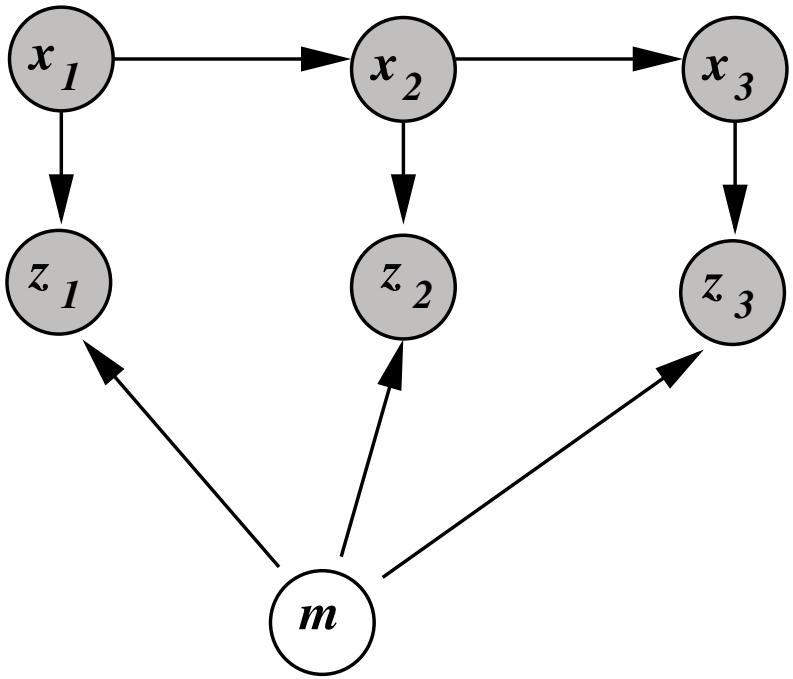
$$\bar{\Sigma}_t = \Sigma_t$$

$$p_{z_t} = \prod_j \frac{1}{\sqrt{\det 2\pi S_t^j}} \exp\left\{ -\frac{1}{2} (z_t^j - \hat{z}_t^j)^T (S_t^j)^{-1} (z_t^j - \hat{z}_t^j) \right\}$$

# Mapping

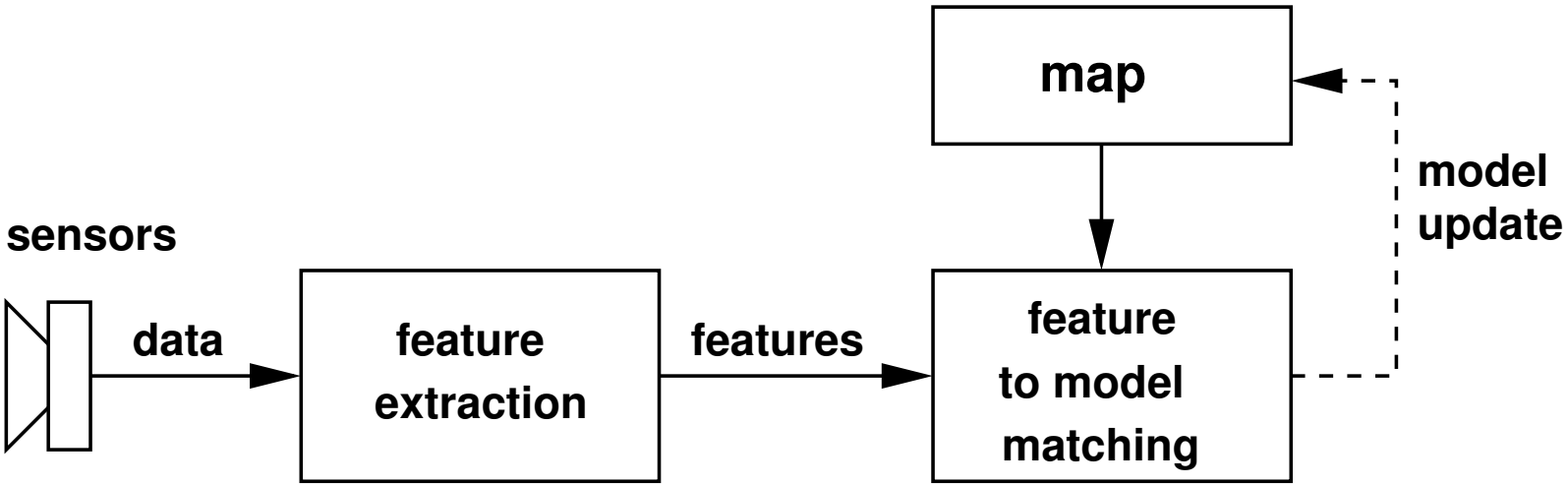
## Task definition

Based on **known** series of robot poses (*trajectory*) and sensor readings (*observations*) in those poses – determine obstacles' coordinates and free regions in the robot surrounding (a map *in chosen representation*).




- $m$ – map
- $x_i$ – robot pose in time step $i$
- $z_i$ – observation in time step $i$

# Mapping – scheme




Mapping – phases [Crowley 1989]

1. Creating high level description based on current sensor readings and sensor models
2. Feature (detected by sensors) matching to current environment description
3. Map update (modification of object detection probabilities)

# Mapping – features and sensors

## Typical features

- lines (and line segments)
- corners
- markers

Detected features, even in static environment, must be tracked to verify and correct map if necessary.

## Sensors

In map building one usually uses

- data from robot self-localization (odometry and others)
- laser scanners
- sonars
- mono- and stereo-vision
- depth sensors (TOF, structural light)

# Map taxonomy

## Steadiness

- static
- semi-static
- dynamic

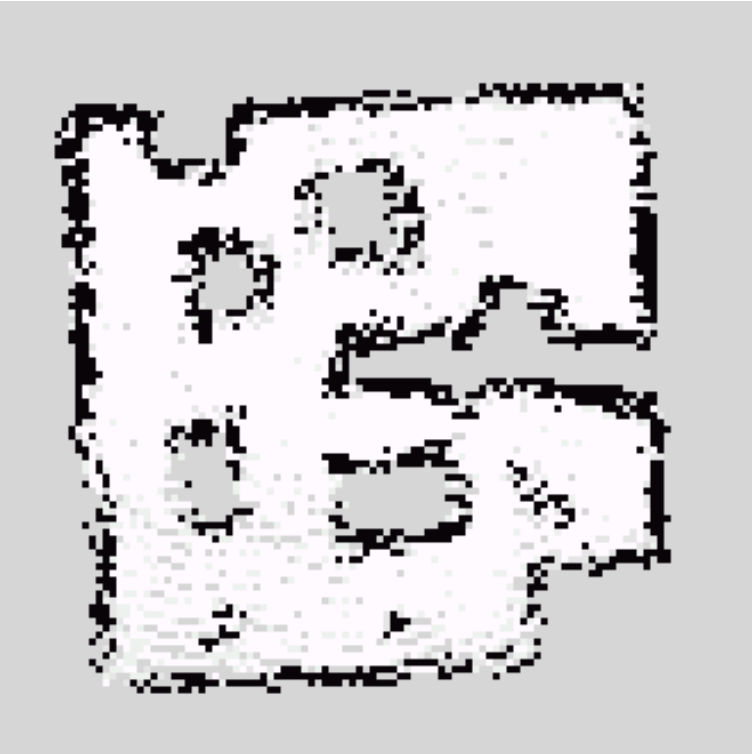
## Structure

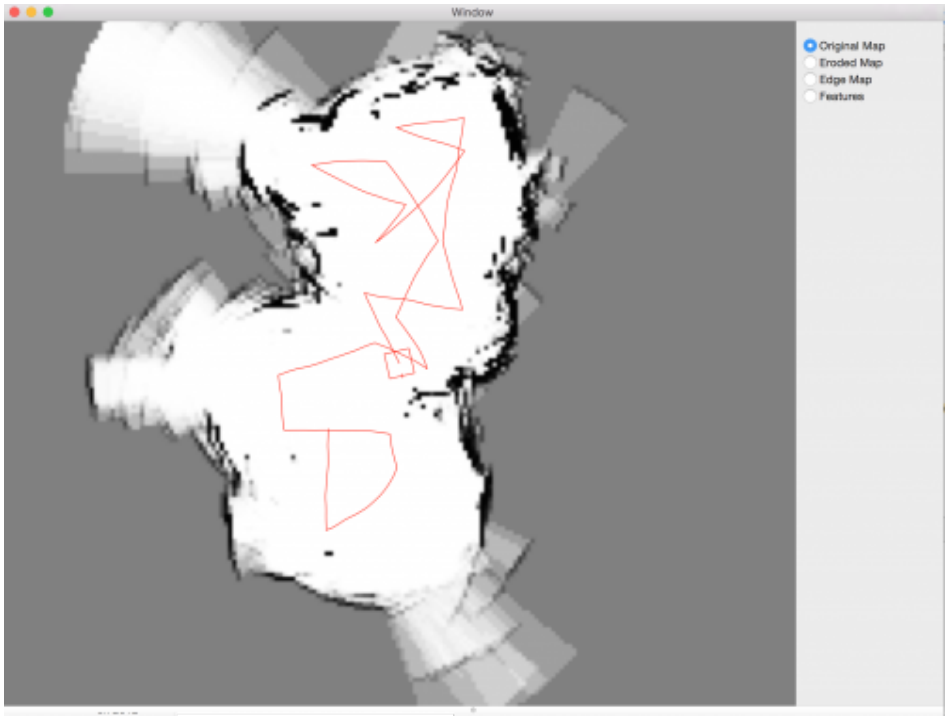
- metric
- topological
- hybrid

## Representation (of distances)

- grid
- vector

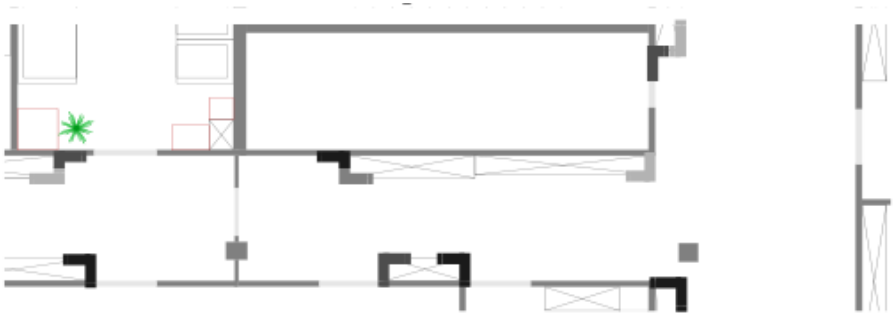
# Grid metric map

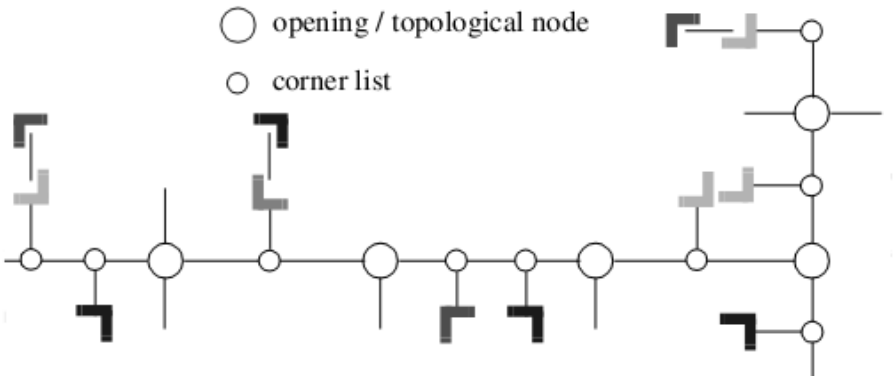


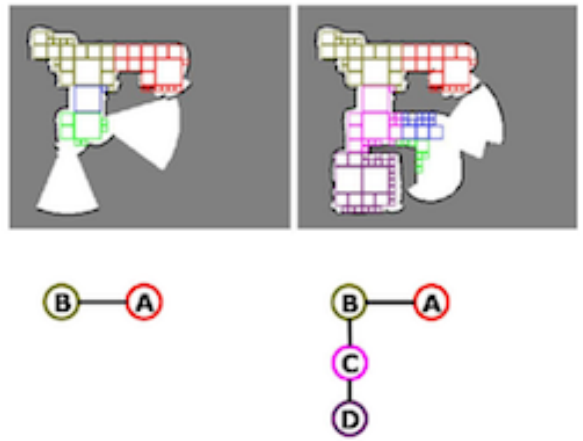


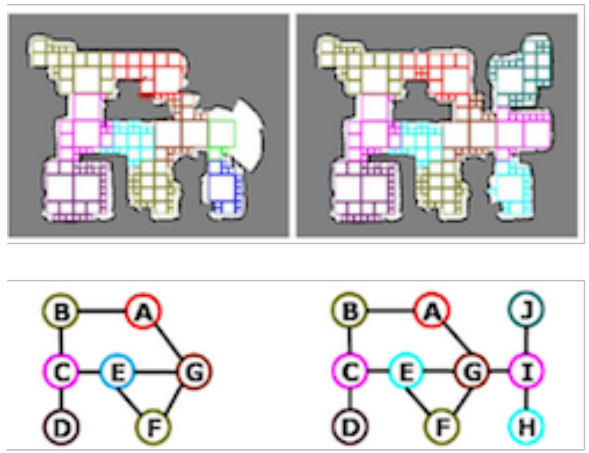

[0] http://emergent.brynmawr.edu/eprg/?page=PyroModuleMapping
http://letsmakerobots.com/robot/project/
mapping-rover-the-classic-rover-5-with-improved-3d-printed-axis-adaptors

# Topological map






[0]Tomatis et al. Combining topological and metric: a natural integration for SLAM, `http://www.autonomousrobotsblog.com/tag/mapping/`

# Comparison of topological and metric maps

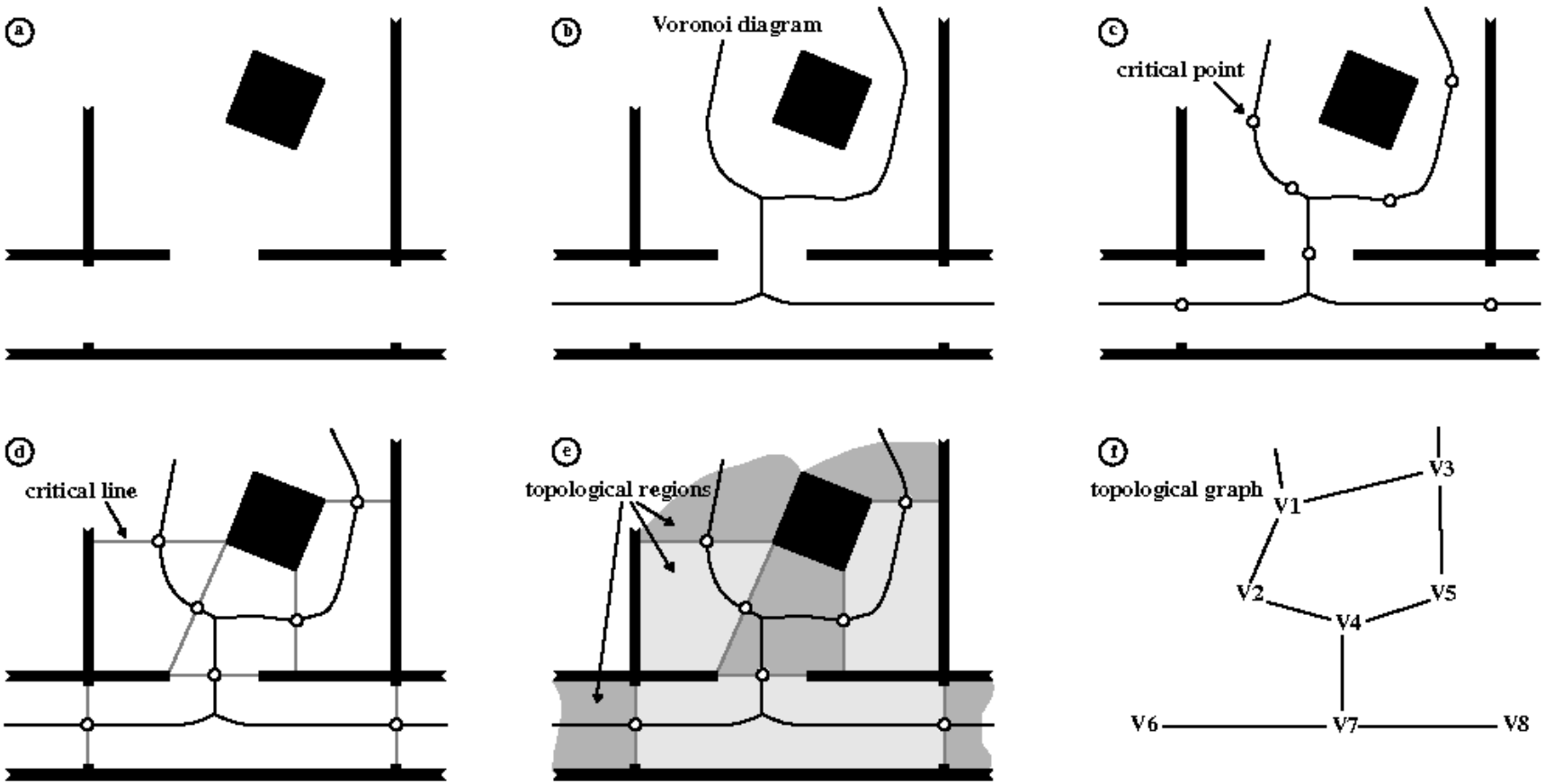
## Metric maps advantages

- ease of creation, representation, updates
- recognition independent of robot pose
- optimization of path lengths possible

## Topological maps advantages

- complexity dependant on region, not its size
- high precision of robot localization is not necessary
- simpler high level processing

# Conversion of metric map to topological




[0]Thrun. Learning maps for indoor mobile robot navigation

# Mapping – complexity

## Main factors influencing complexity:

- size – environment area with respect to sensors range
- noise in measurements and localization
- ambiguity of perception (multiple hypotheses matching)
- closed cycles (with localization errors)

# Practical limitations

- available measurements types
- sensor ranges
- measurement errors
- ambiguity of interpretation
- localization errors
- environment dynamics
- real time processing requirement

# Occupancy grid map

Searching for a (static) map to maximize probability

$$p(m|z_{1:t}, x_{1:t}).$$

Divide a map into cells $m = \{m_i\}$, each one marked with occupancy probability (1 – occupied, 0 – empty).

## Scale problem:

Map with 100x100 cells means $10^5$ cells, then number of possible maps: $2^{10000}$.

## Standard solution:

Assume independence of map cells and calculate each cell separately

$$p(m|z_{1:t}, x_{1:t}) \simeq \Pi_i \; p(m_i|z_{1:t}, x_{1:t})$$

Map is updated with use of binary Bayes filter and some representation of probability

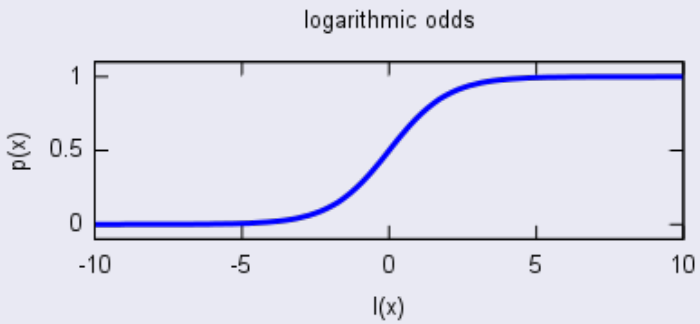
# Grid maps – representation of cell value

1. probability $p(x) \in [0, 1]$
2. logarithmic $l(x) \in (-\infty, +\infty)$

$$l(x) = \log \frac{p(x)}{p(\neg x)} = \log \frac{p(x)}{1 - p(x)}$$

with probabilities recovered by

$$p(x) = 1 - \frac{1}{1 + \exp\{l(x)\}}$$




3. hits/misses $(h(x), f(x))$

# Occupancy grid map – algorithm

$l_0 = \log \frac{p(m_i=1)}{p(m_i=0)} = \log \frac{p(m_i)}{1-p(m_i)}$

foreach $m_i$ do

  if $m_i$ in perception_field$(z_t)$

    $l_{t,i} = l_{t-1,i} + inverse\_sensor\_model(m_i, x_t, z_t) - l_0$

  *else*

    $l_{t,i} = l_{t-1,i}$

  endif

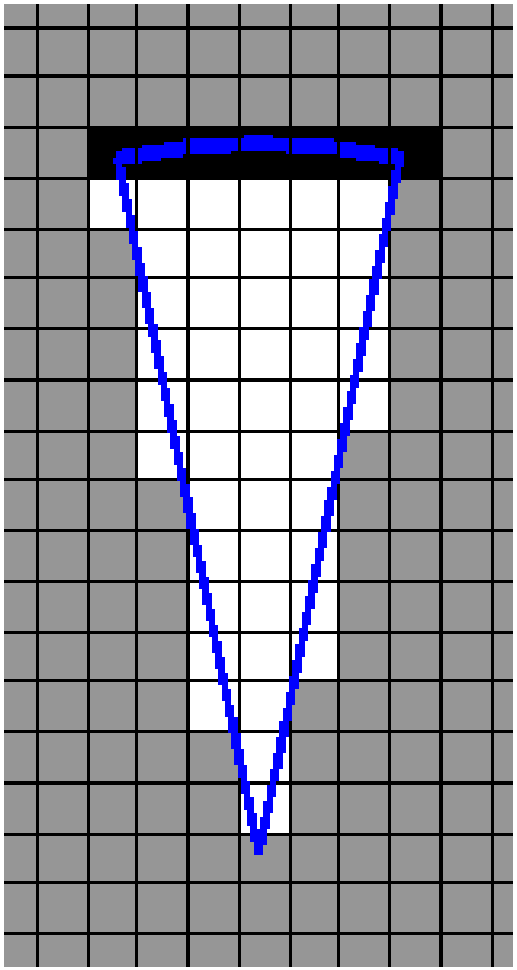
endfor

$p(m_i|z_{1:t}, x_{1:t}) = 1 - \frac{1}{1+\exp\{l_{t,i}\}}$

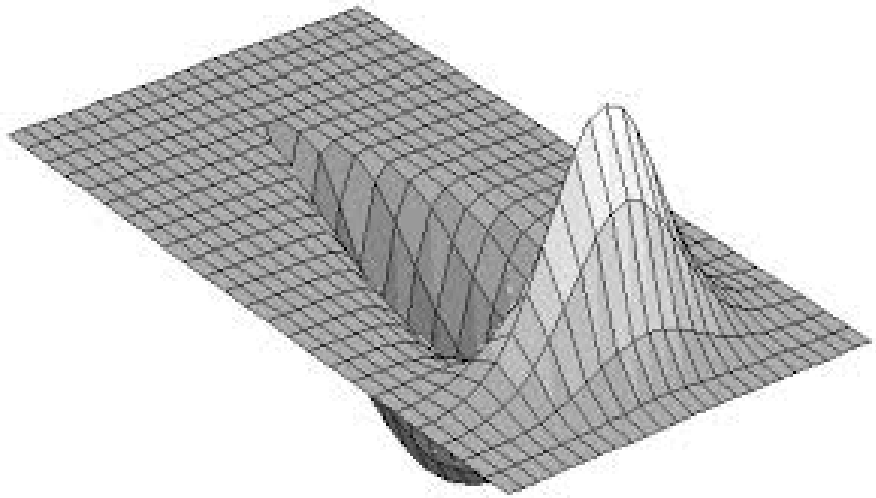
$inverse\_sensor\_model(m_i, x_t, z_t) = \log \frac{p(m_i|z_t,x_t)}{1-p(m_i|z_t,x_t)}$

# Sensor model

Simple version



All details:



# Occupancy grid and sensor fusion

## Question

How to create a map using simultanously various sensors?

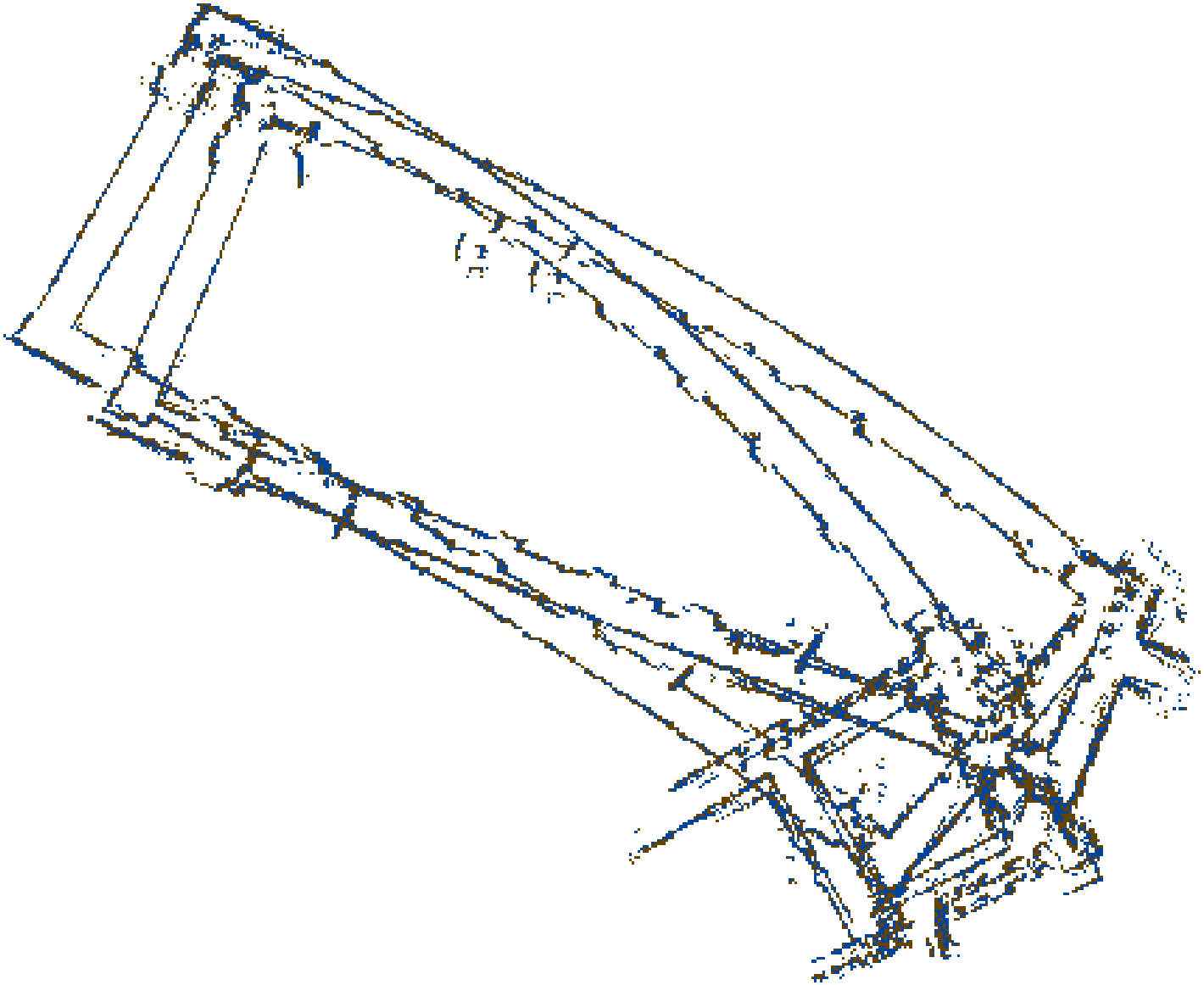
1. (simple) Run the algorithm consecutively for each sensor and the same map.
   Problem: If sensors detect various obstacles, map may be unstable
2. (typical) Create and update separate maps for each sensor type, when a map is needed – combine the sensor maps with chosen method, for example, if measurements are independent

$$p(m_i) = 1 - \Pi_k(1 - p(m_i^k)),$$

or using the most pessimistic hypothesis

$$p(m_i) = \max_k p(m_i^k).$$

# Consequences of odometric error in cycles



[0]Thrun. Integrating topological and metric maps...