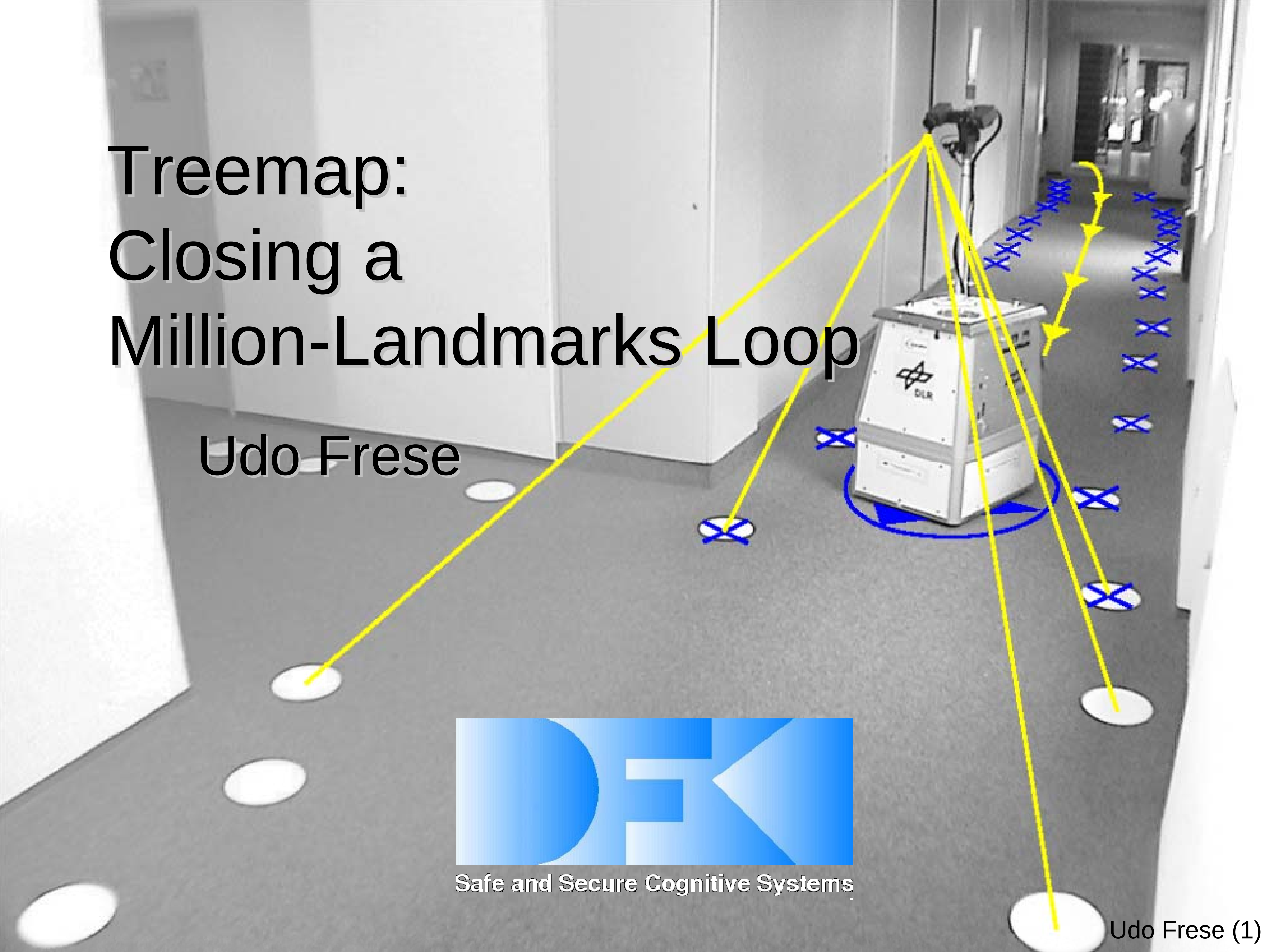
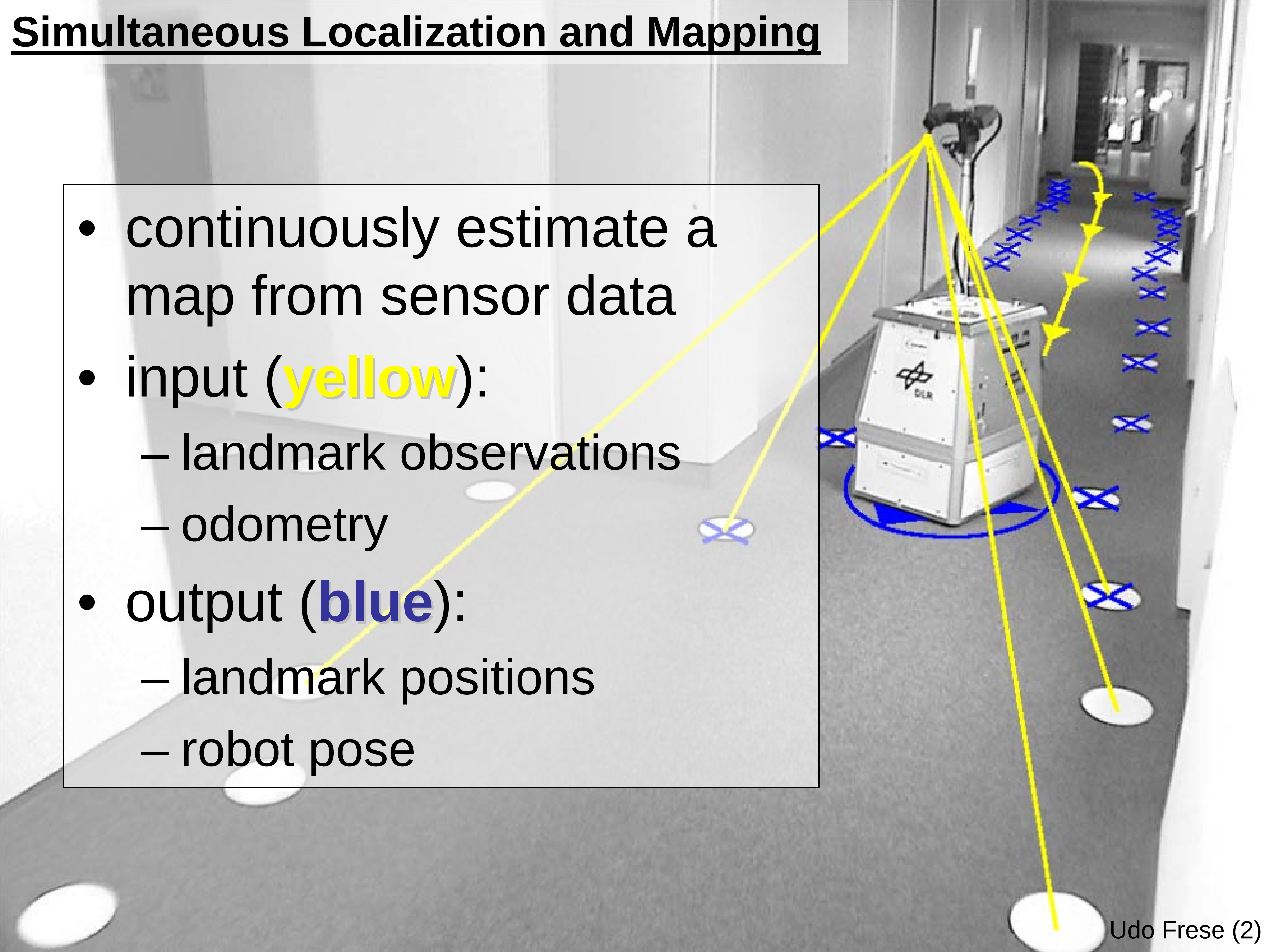


Treemap: Closing a Million-Landmarks Loop

Udo Frese

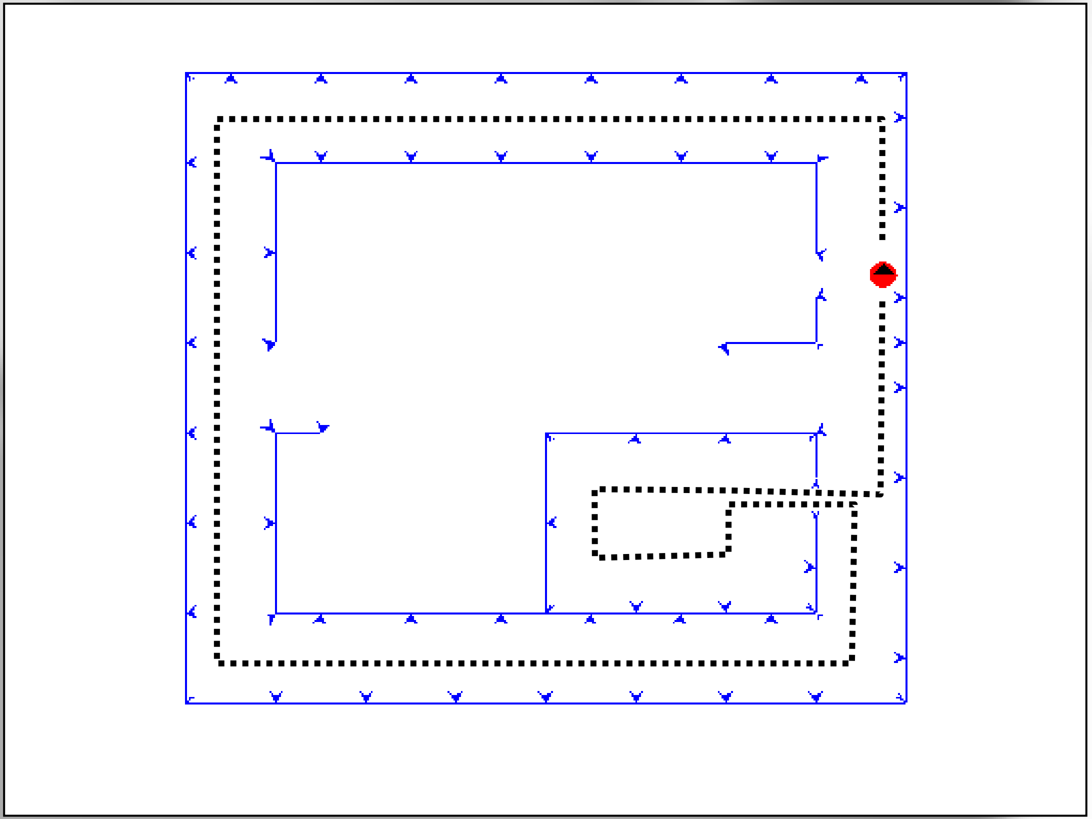
Safe and Secure Cognitive Systems

Simultaneous Localization and Mapping



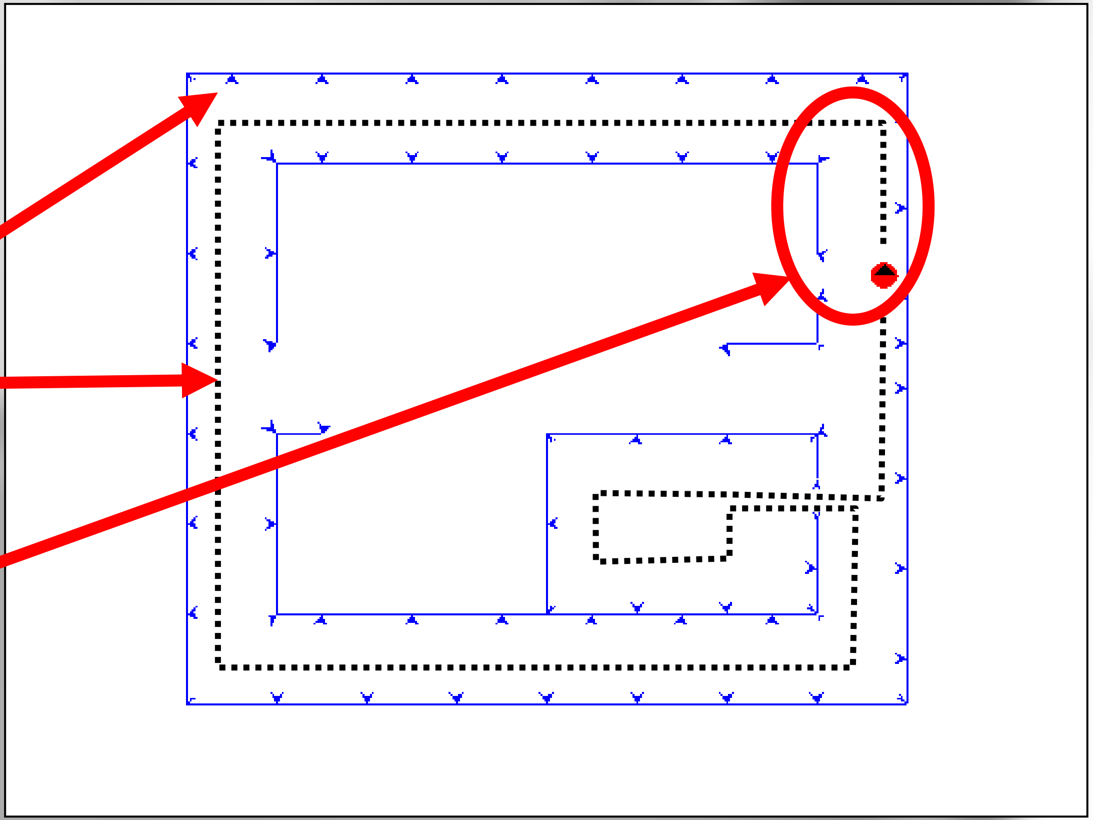
- continuously estimate a map from sensor data
- input (yellow):
  - landmark observations
  - odometry
- output (blue):
  - landmark positions
  - robot pose

Simultaneous Localization and Mapping

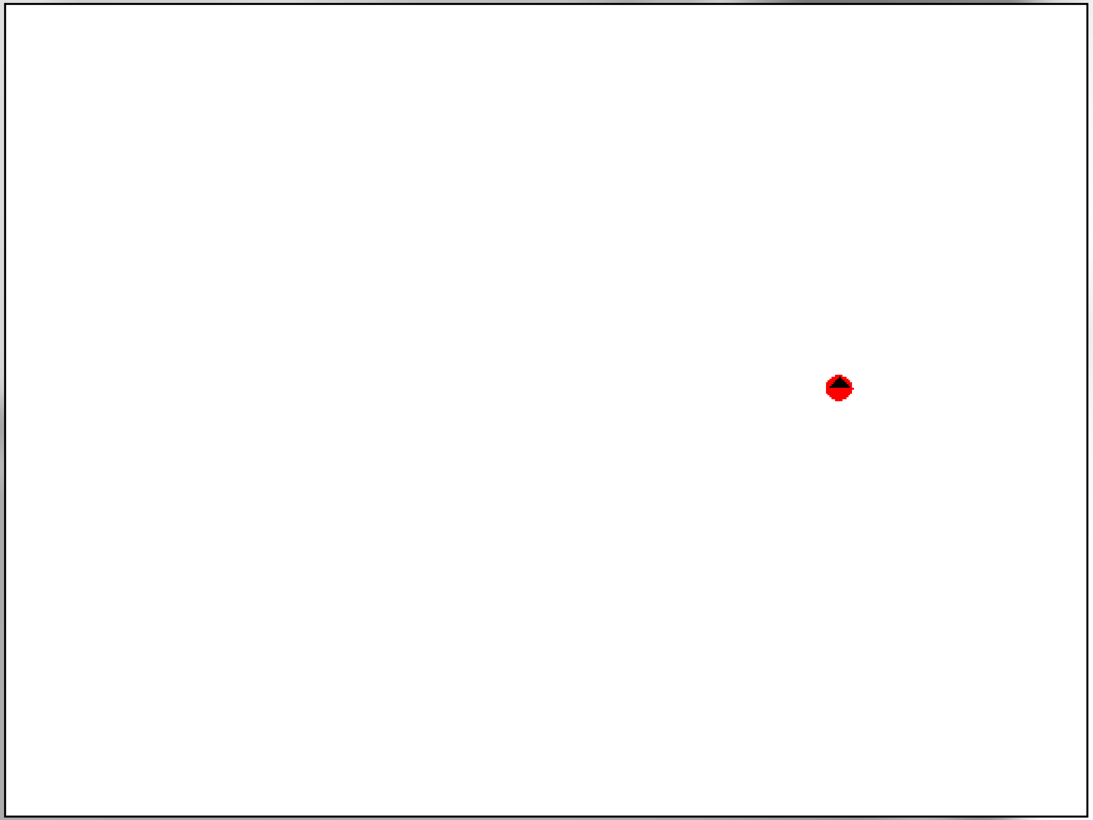


Simultaneous Localization and Mapping

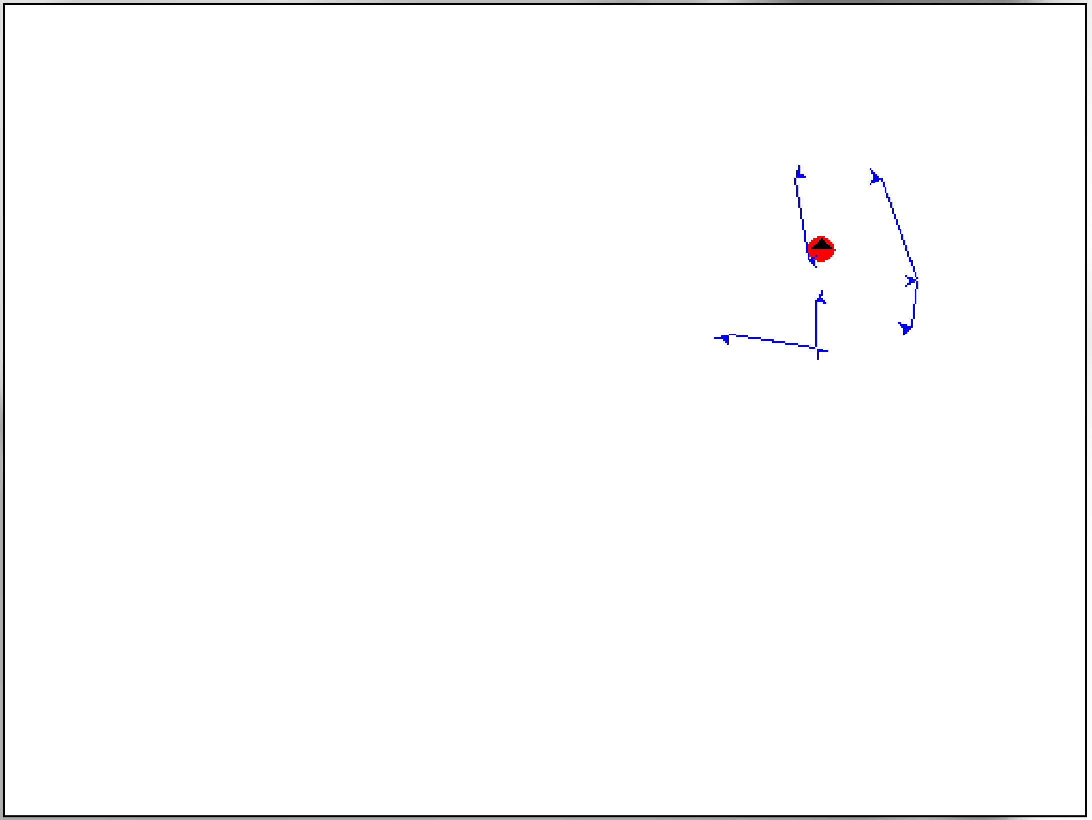
n	landmarks
p	robot poses
k	local
=O(1)	landmarks



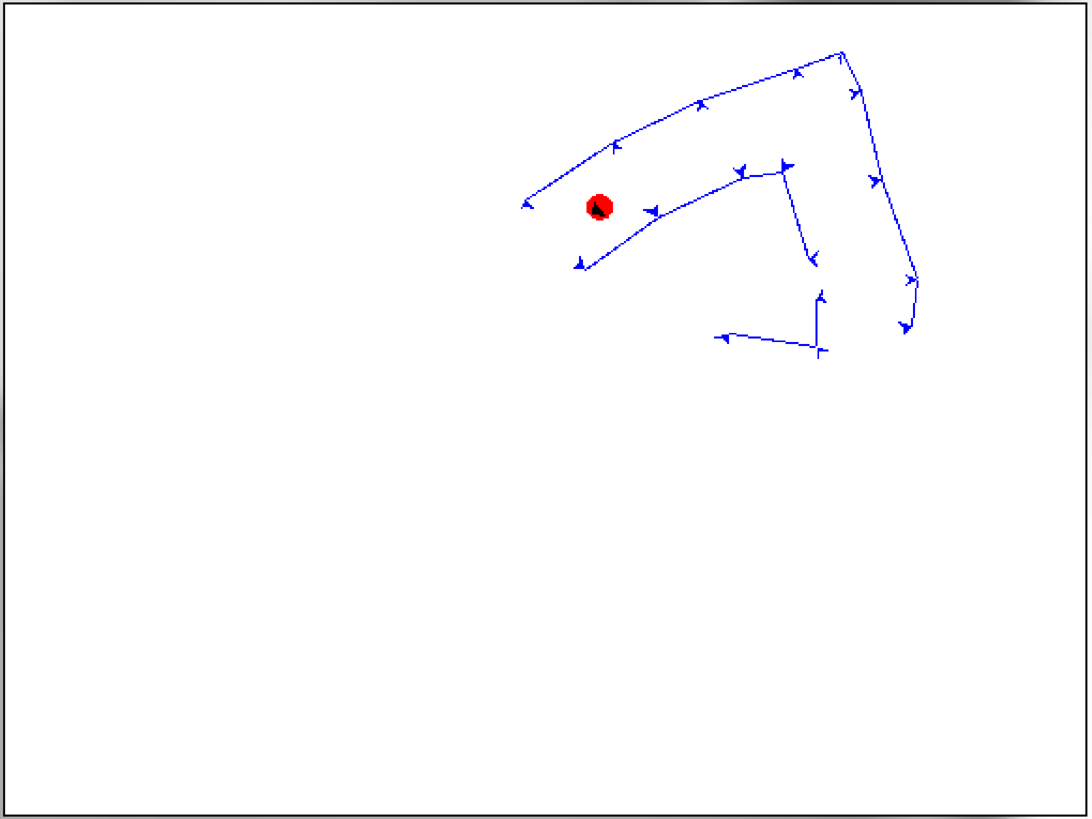
Simultaneous Localization and Mapping



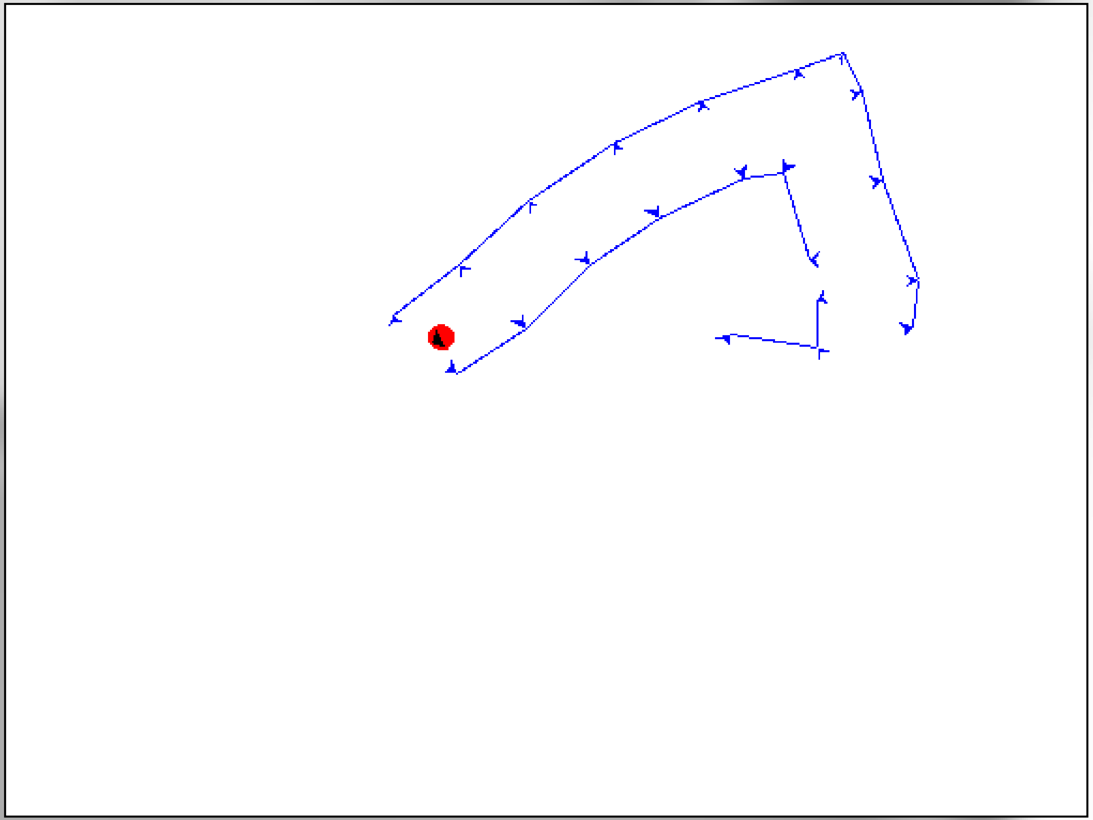
Simultaneous Localization and Mapping



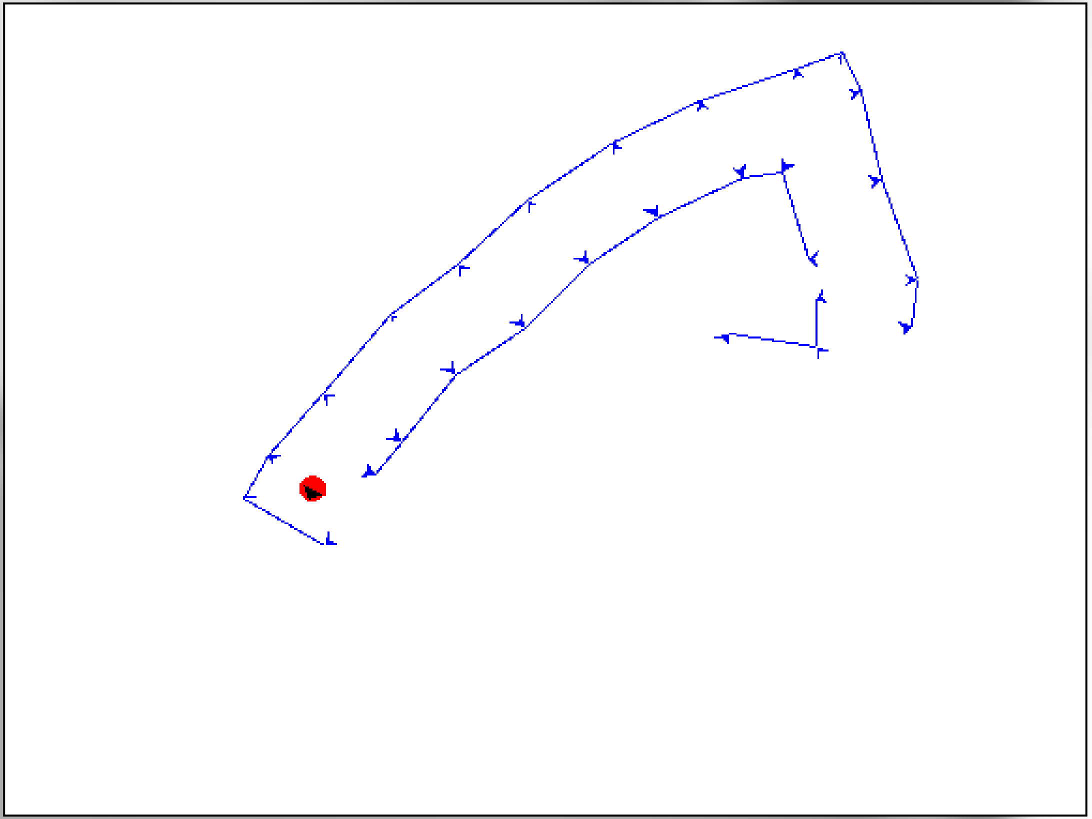
Simultaneous Localization and Mapping



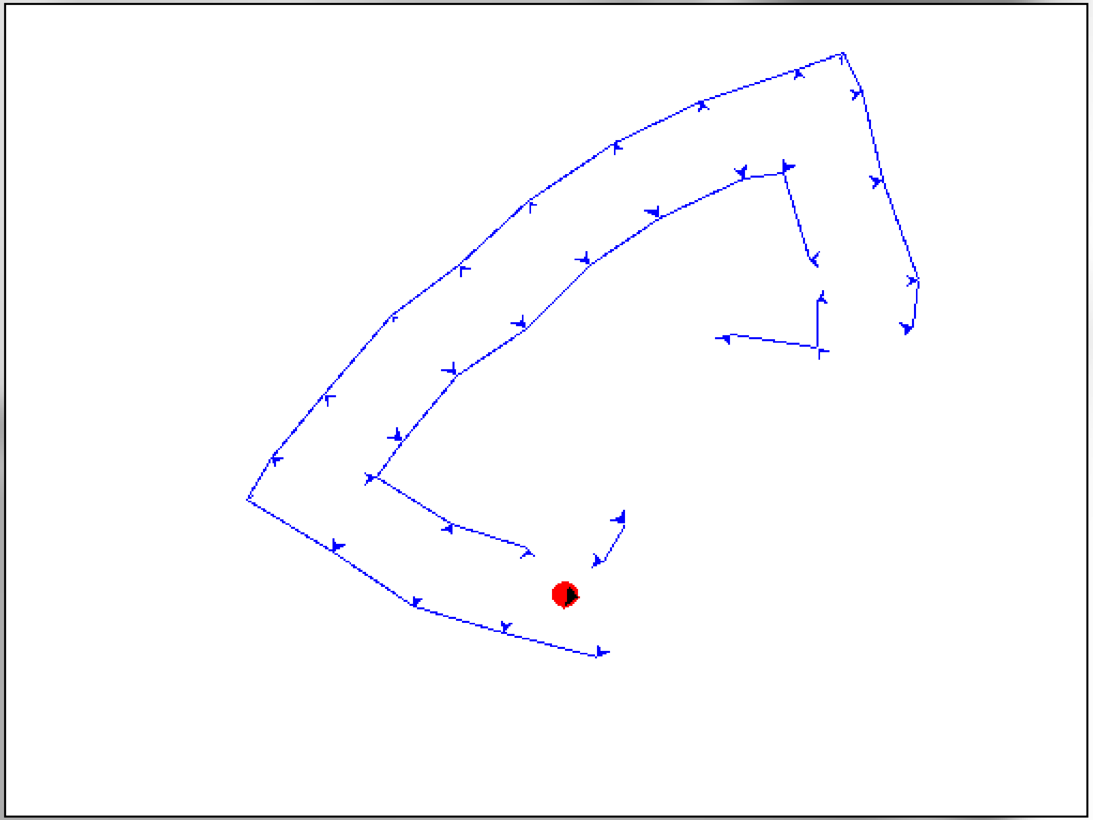
Simultaneous Localization and Mapping



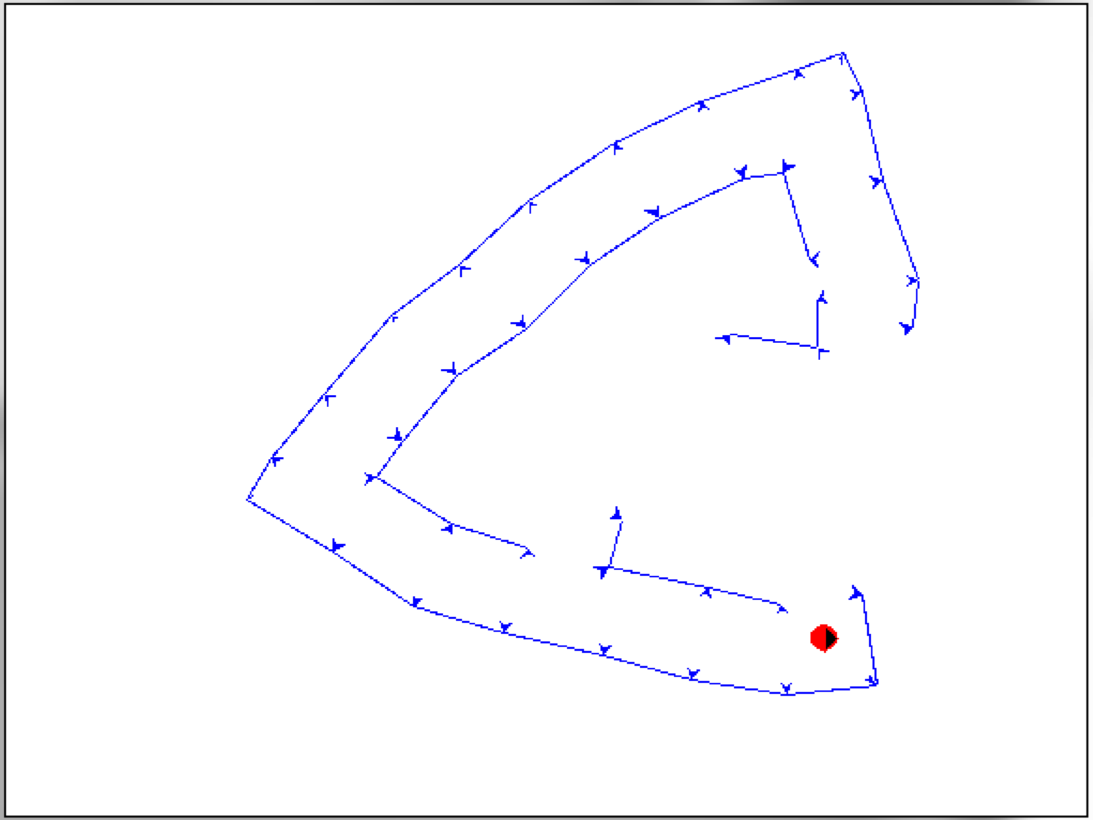
Simultaneous Localization and Mapping



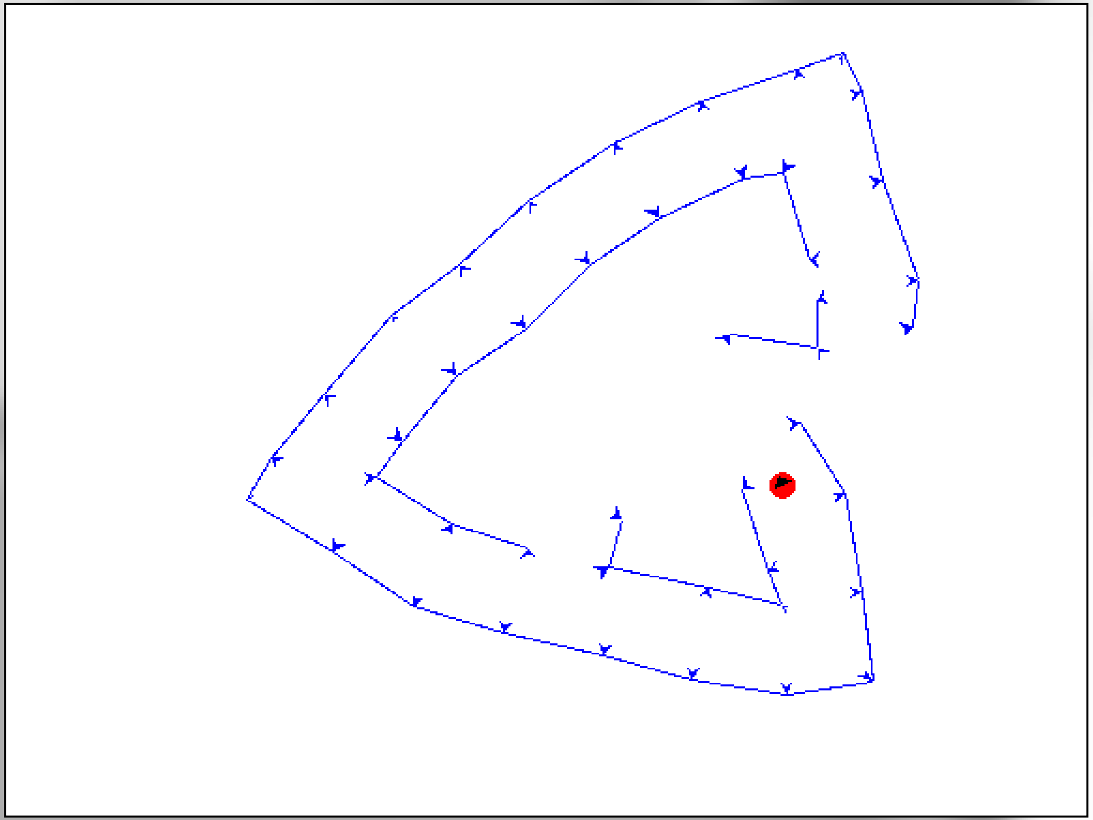
Simultaneous Localization and Mapping



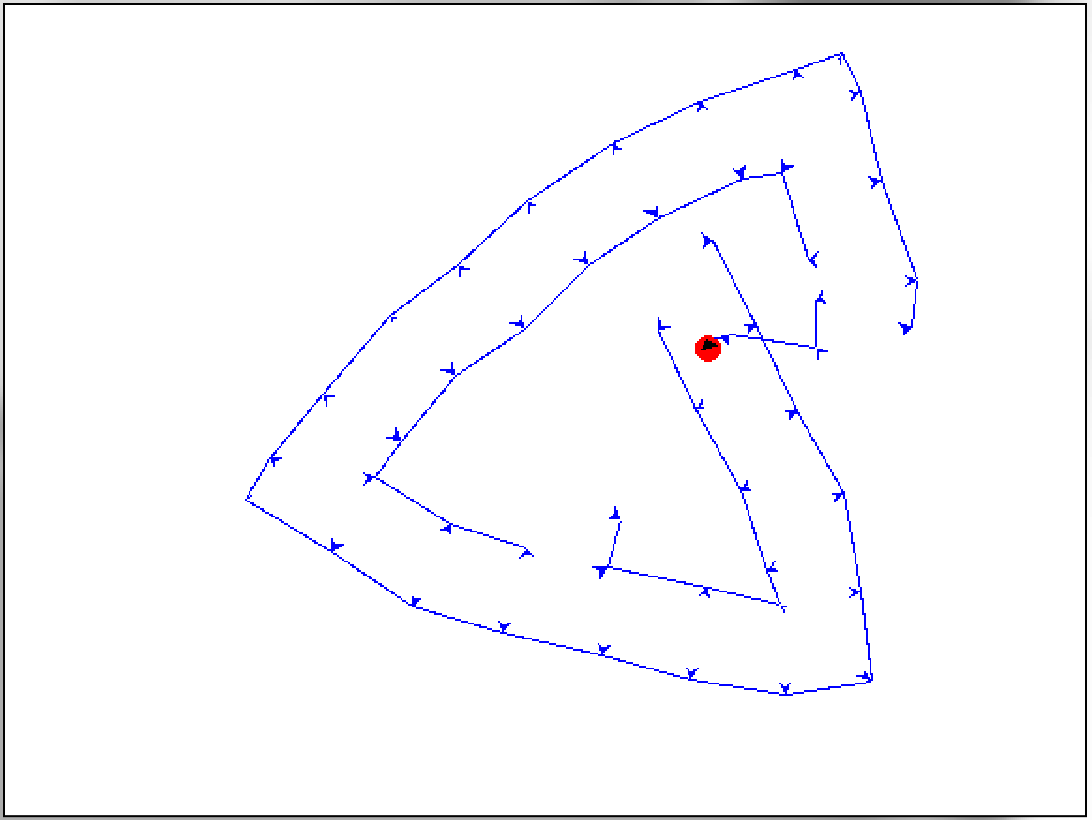
Simultaneous Localization and Mapping



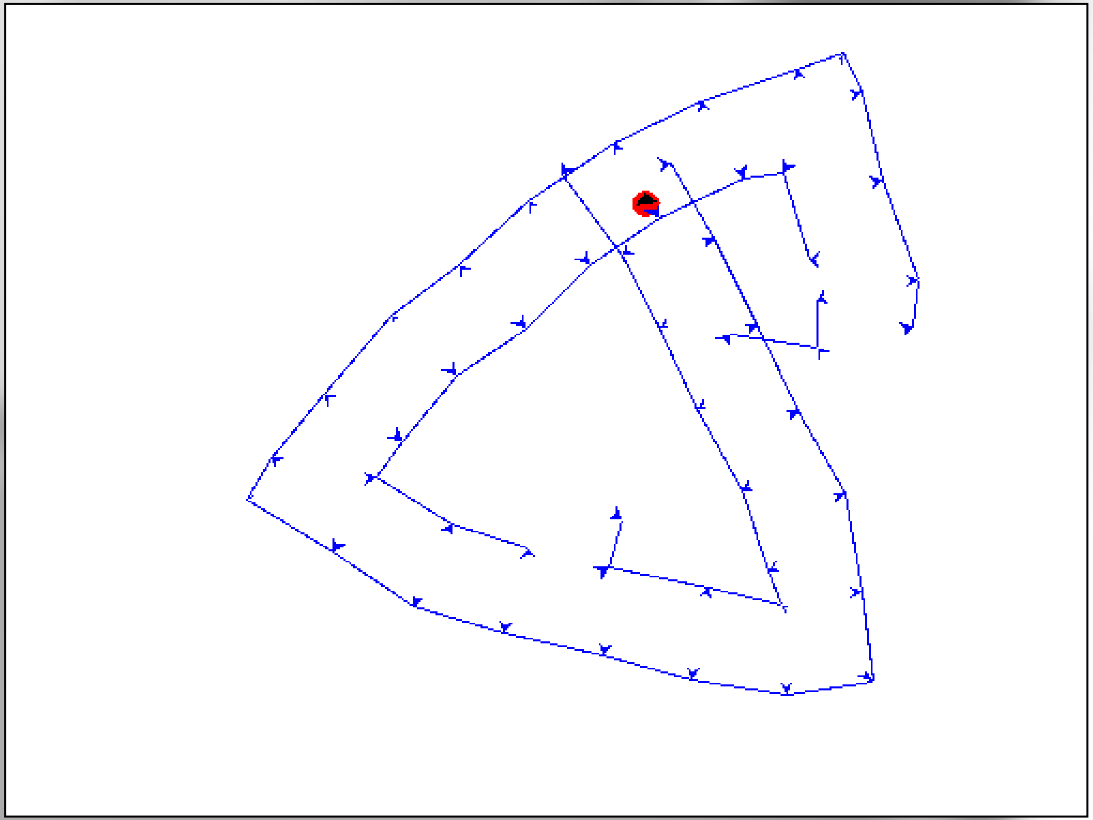
Simultaneous Localization and Mapping



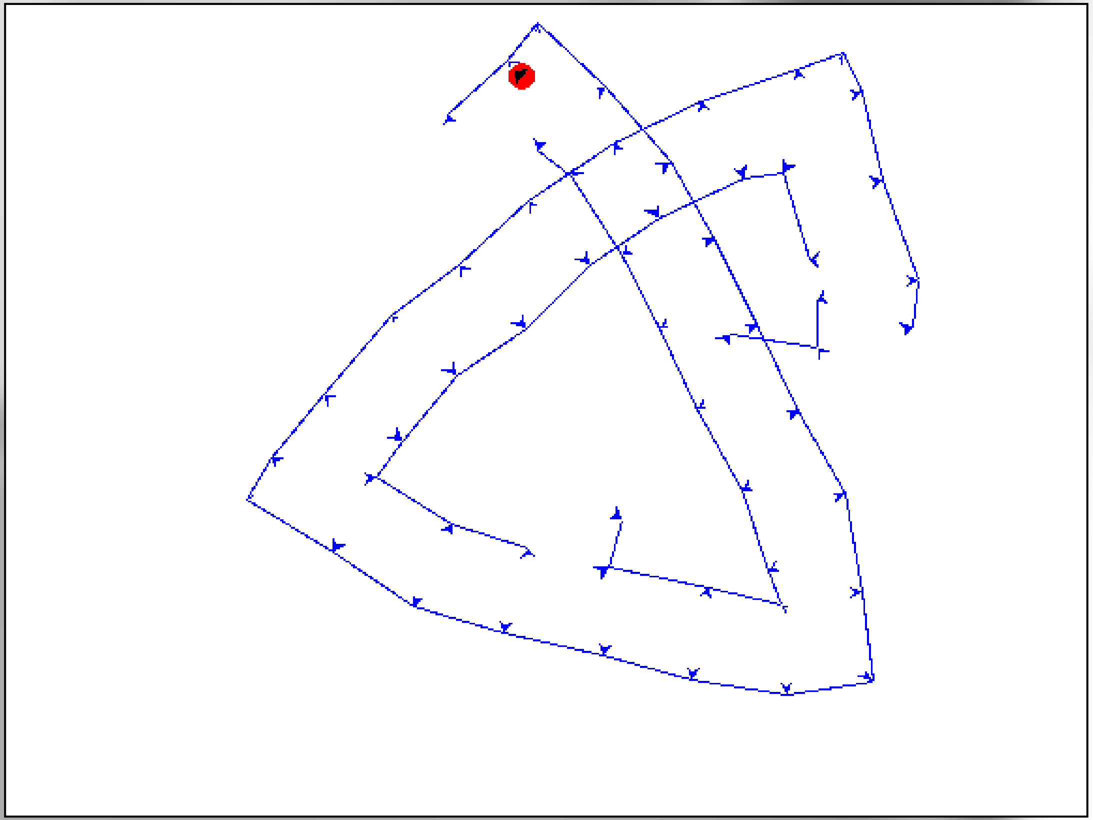
Simultaneous Localization and Mapping



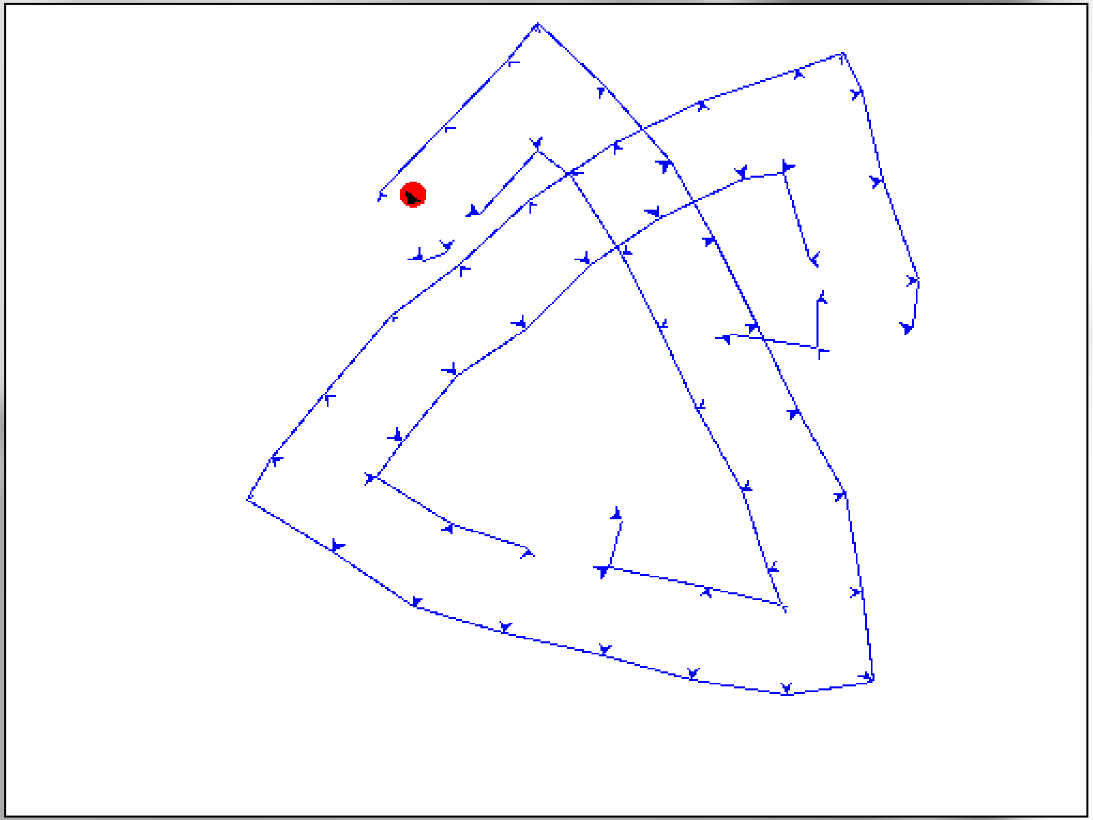
Simultaneous Localization and Mapping



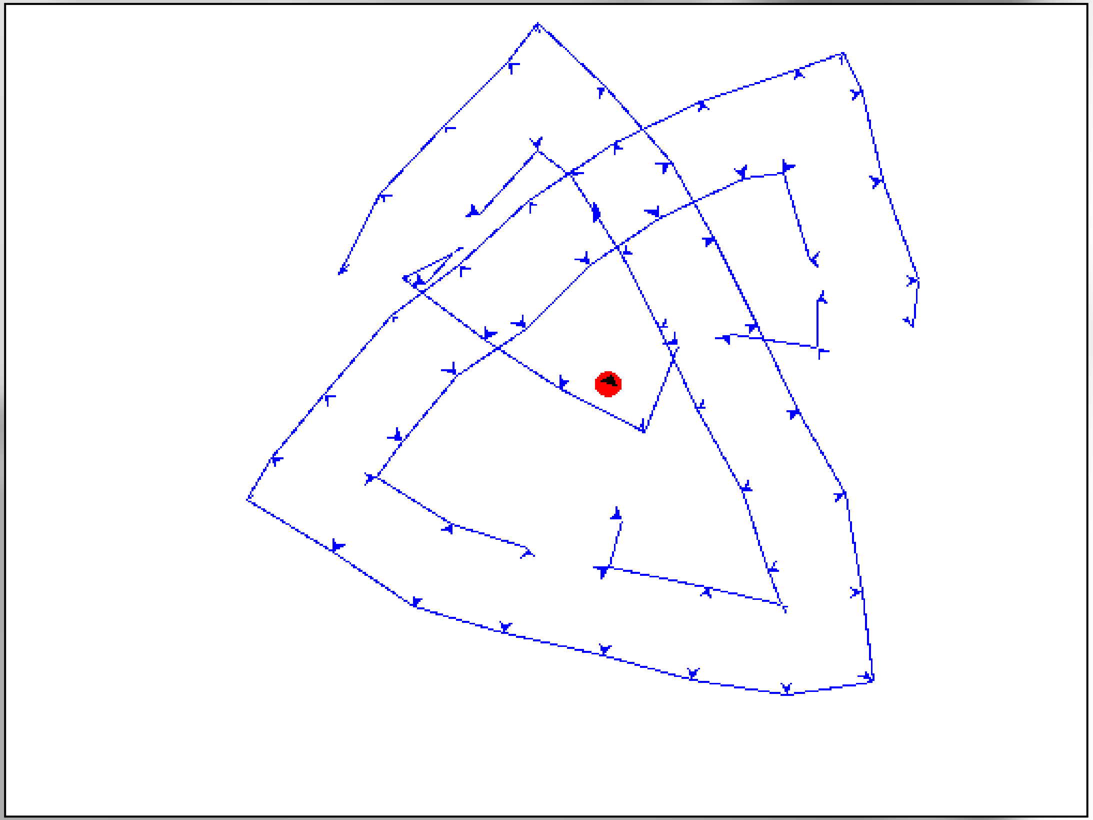
Simultaneous Localization and Mapping



Simultaneous Localization and Mapping

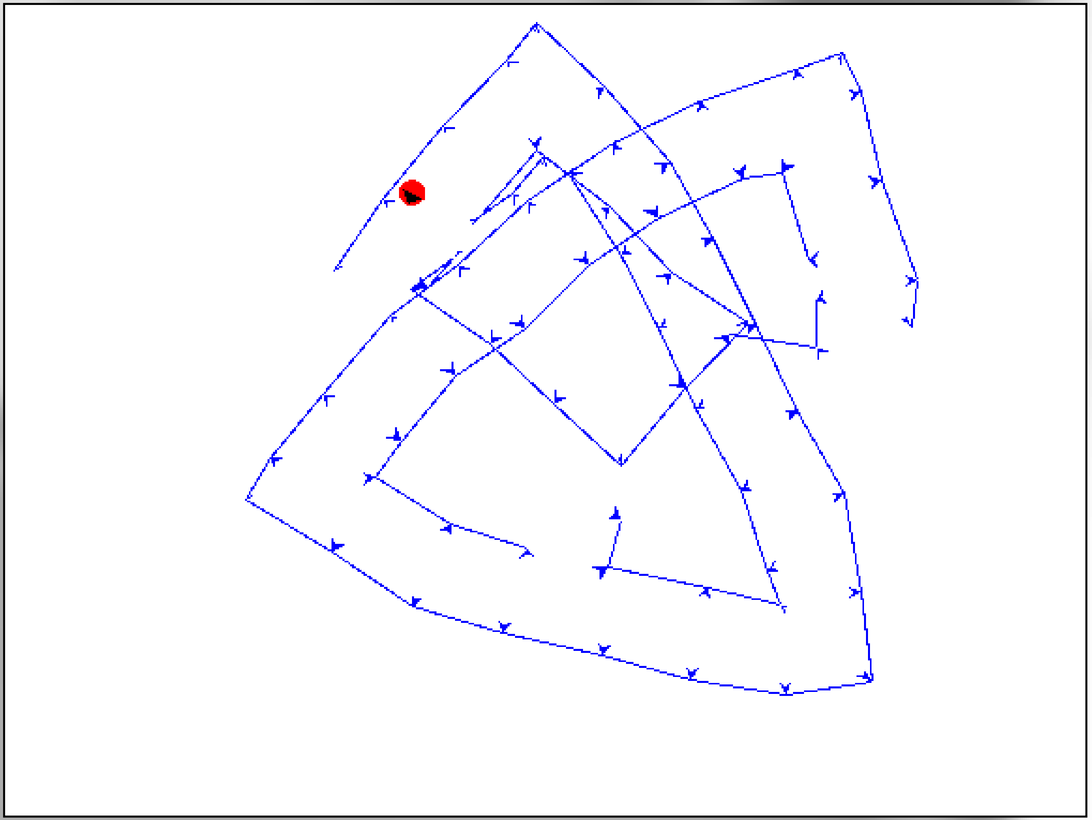


Simultaneous Localization and Mapping



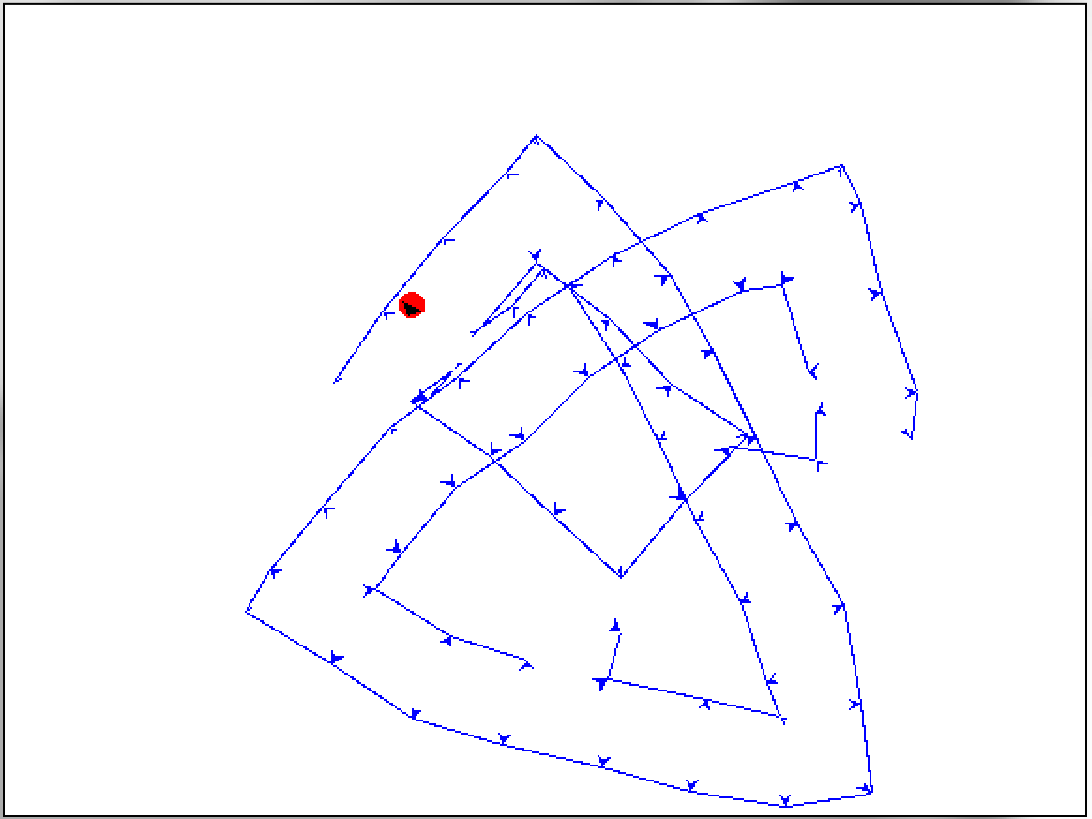
Simultaneous Localization and Mapping

- problem: accumulated error



Simultaneous Localization and Mapping

SLAM Uncertainty 1



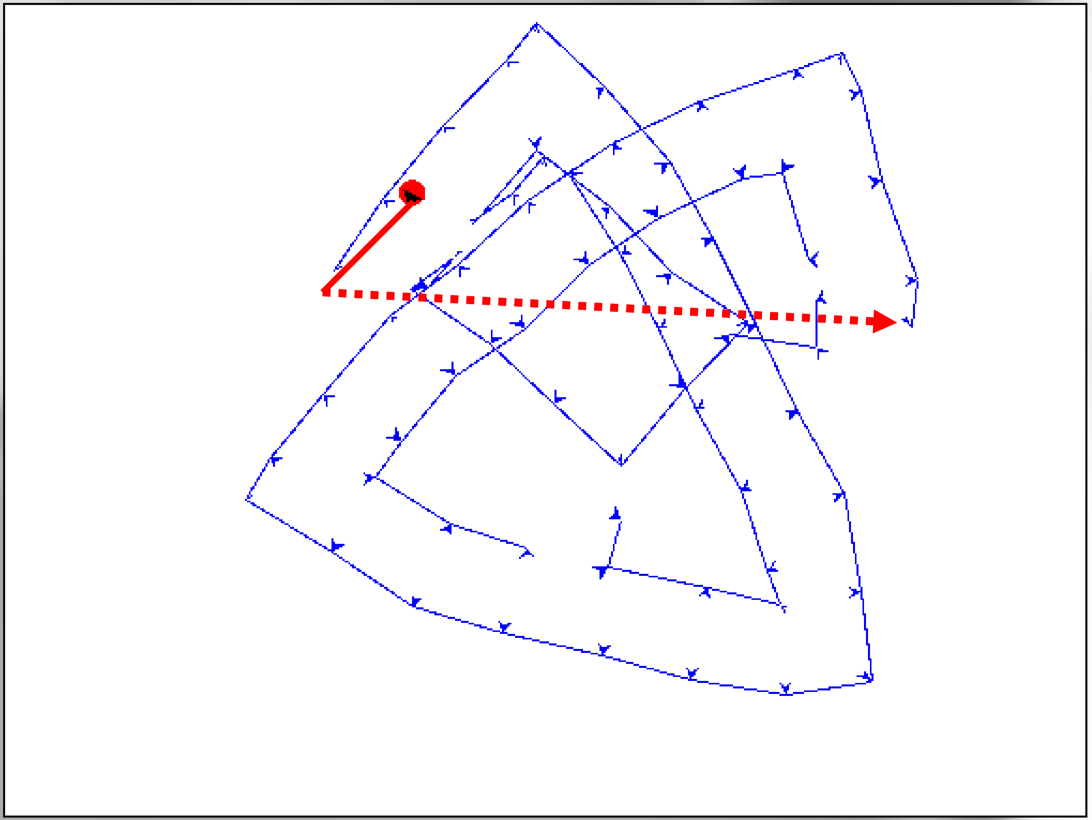
- accumulated error affects position not shape

„Certainty of Relations despite Uncertainty of Positions“

[1] U. Frese (2006), A Discussion of Simultaneous Localization and Mapping. In Autonomous Robots, 20 (1), pp. 25–42

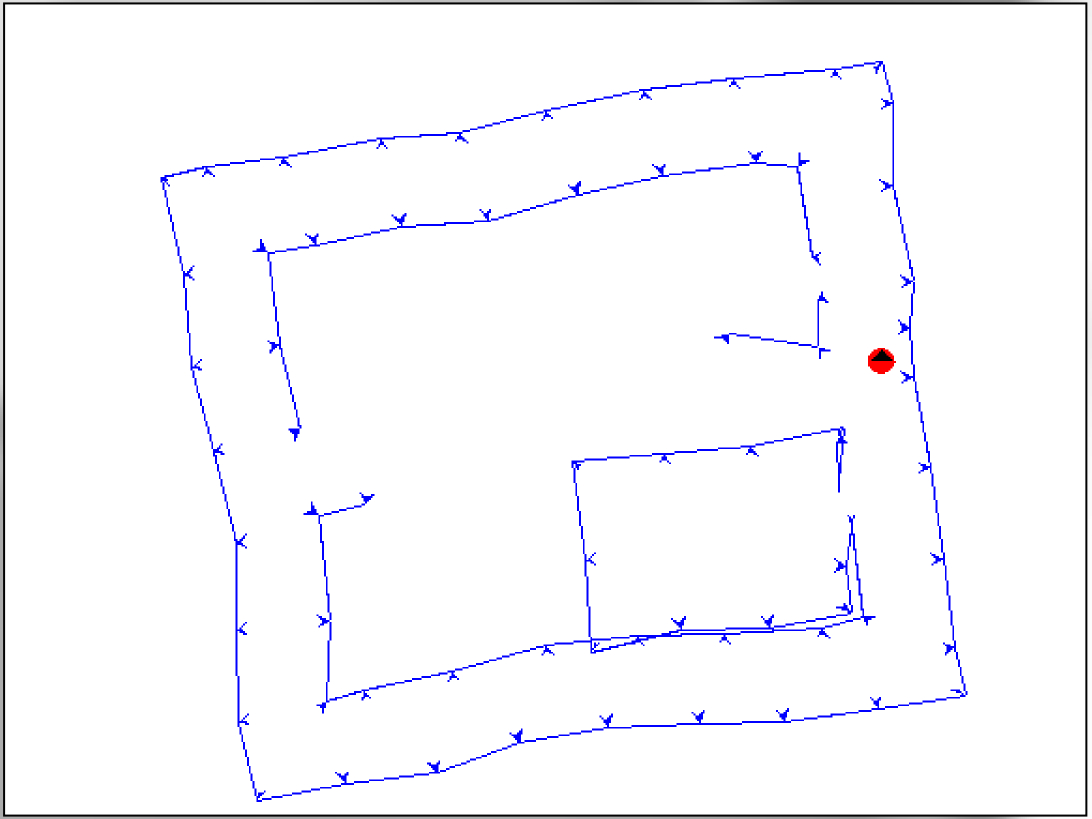
Simultaneous Localization and Mapping

- closing a loop by re-identifying a landmark
- „bending“ the map



Simultaneous Localization and Mapping

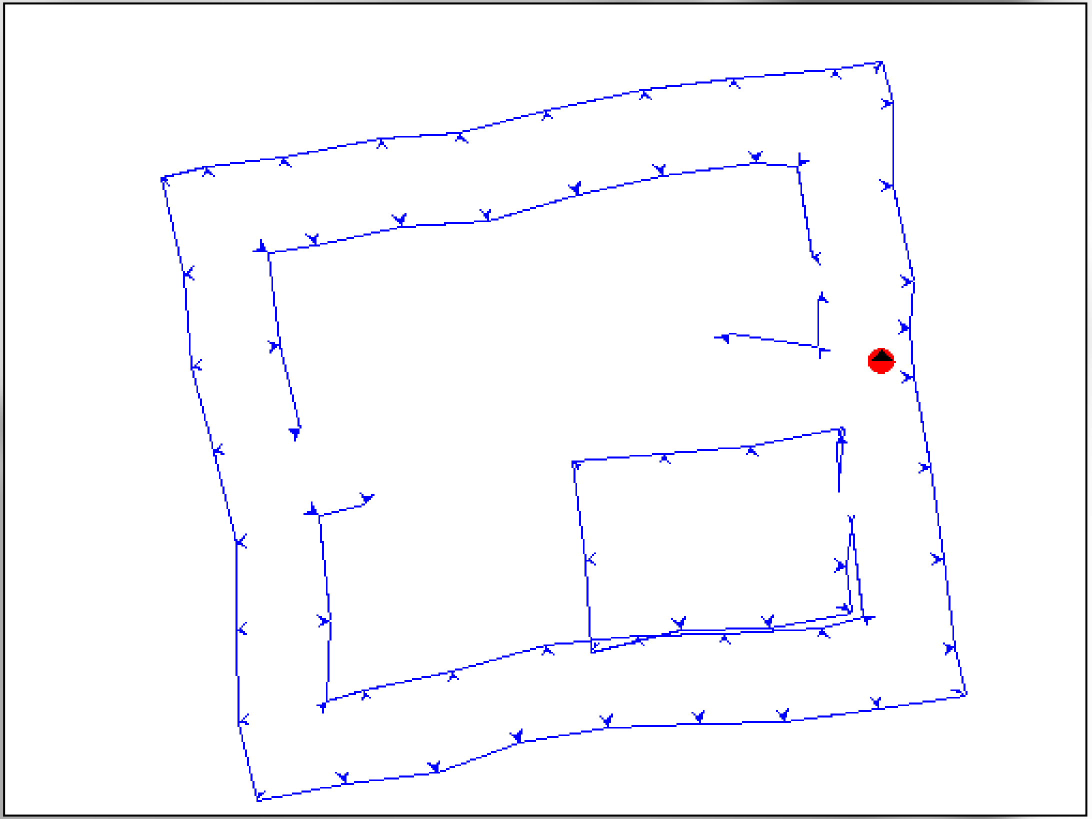
- implicitly done by proper statistical evaluation



Simultaneous Localization and Mapping

SLAM Uncertainty 2

- closing the loop: single measurement drastically reduces the overall error



Simultaneous Localization and Mapping

- optimal solution: (nonlinear) least square estimation following C.F. Gauss
- nonlinear maximum likelihood estimation
- linear equation system
- problem: computation time

Simultaneous Localization and Mapping

Algorithm	Quality	Storage	Computation time		
Max. Likel.	optimal	n+kp	$(n+p)^3$		
EKF	linear	n^2	n^2		
CEKF	linear	$n^{3/2}$	k^2	$kn^{3/2}$	
Treemap	**nonlin.**	**kn**	**k^2**	**$k^3 \log n$**	**kn**

n landmarks (725)

p robot poses (3297)

k local landmarks (15)

same region

new region

global

Simultaneous Localization and Mapping

Algorithm	Quality	Storage	Computation time		
Max. Likel.	optimal	n+kp	$(n+p)^3$		
EKF	linear	n^2	n^2		
CEKF	linear	$n^{3/2}$	k^2	$kn^{3/2}$	
Treemap	**nonlin.**	**kn**	**k^2**	**$k^3 \log n$**	**kn**

n landmarks (725)
p robot poses (3297)
k local landmarks (15)

Theoretical Highlight

Practical Highlight

Simultaneous Localization and Mapping

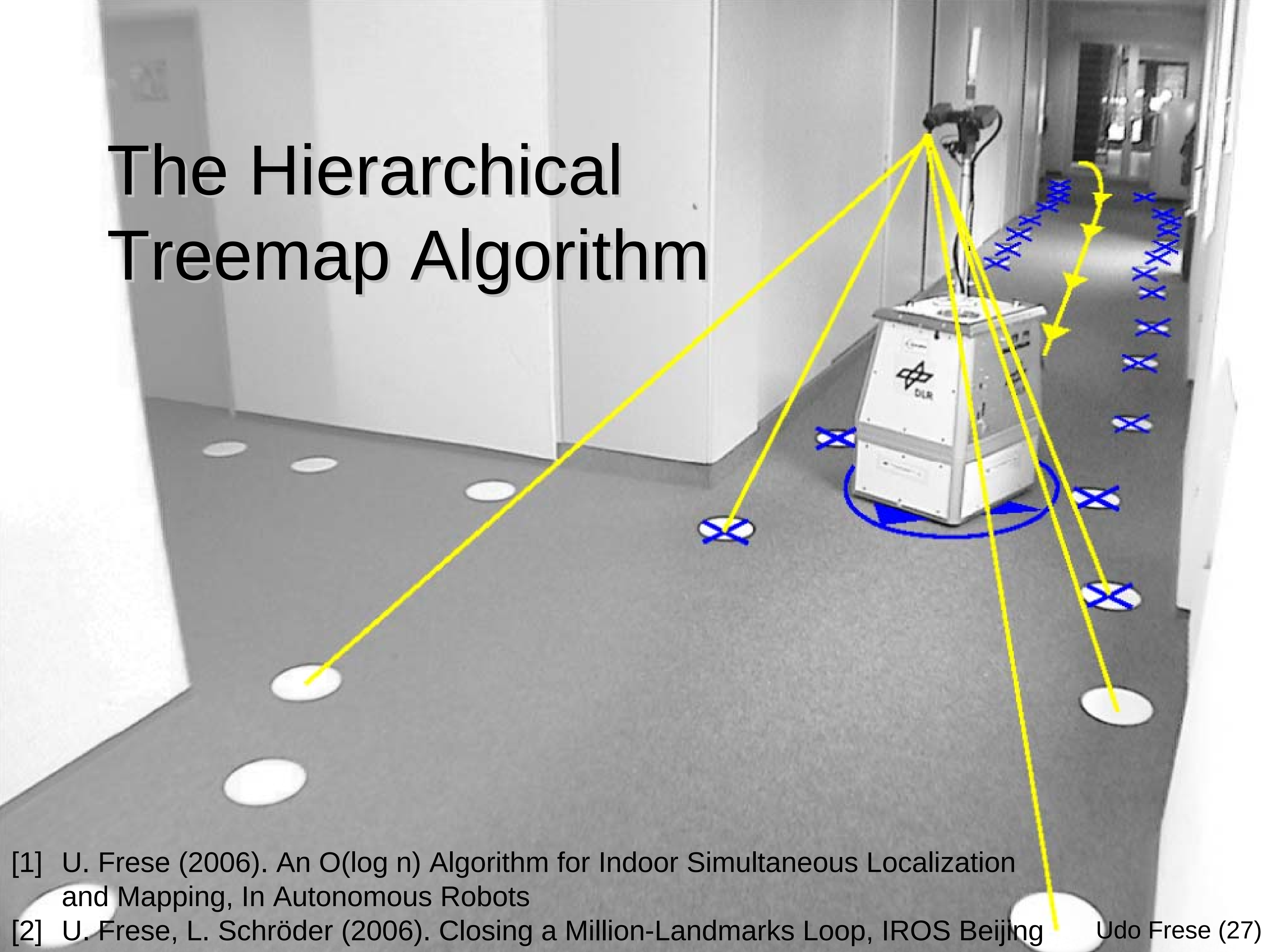
Algorithm	Quality	Storage	Computation time		
Max. Likel.	optimal	n+kp	$(n+p)^3$		
EKF	linear	n^2	n^2		
CEKF	linear	$n^{3/2}$	k^2	$kn^{3/2}$	
Treemap	**nonlin.**	**kn**	**k^2**	▨ **$k^3 \log n$**	**kn**

n landmarks (725)
p robot poses (3297)
k local landmarks (15)

Theoretical Highlight

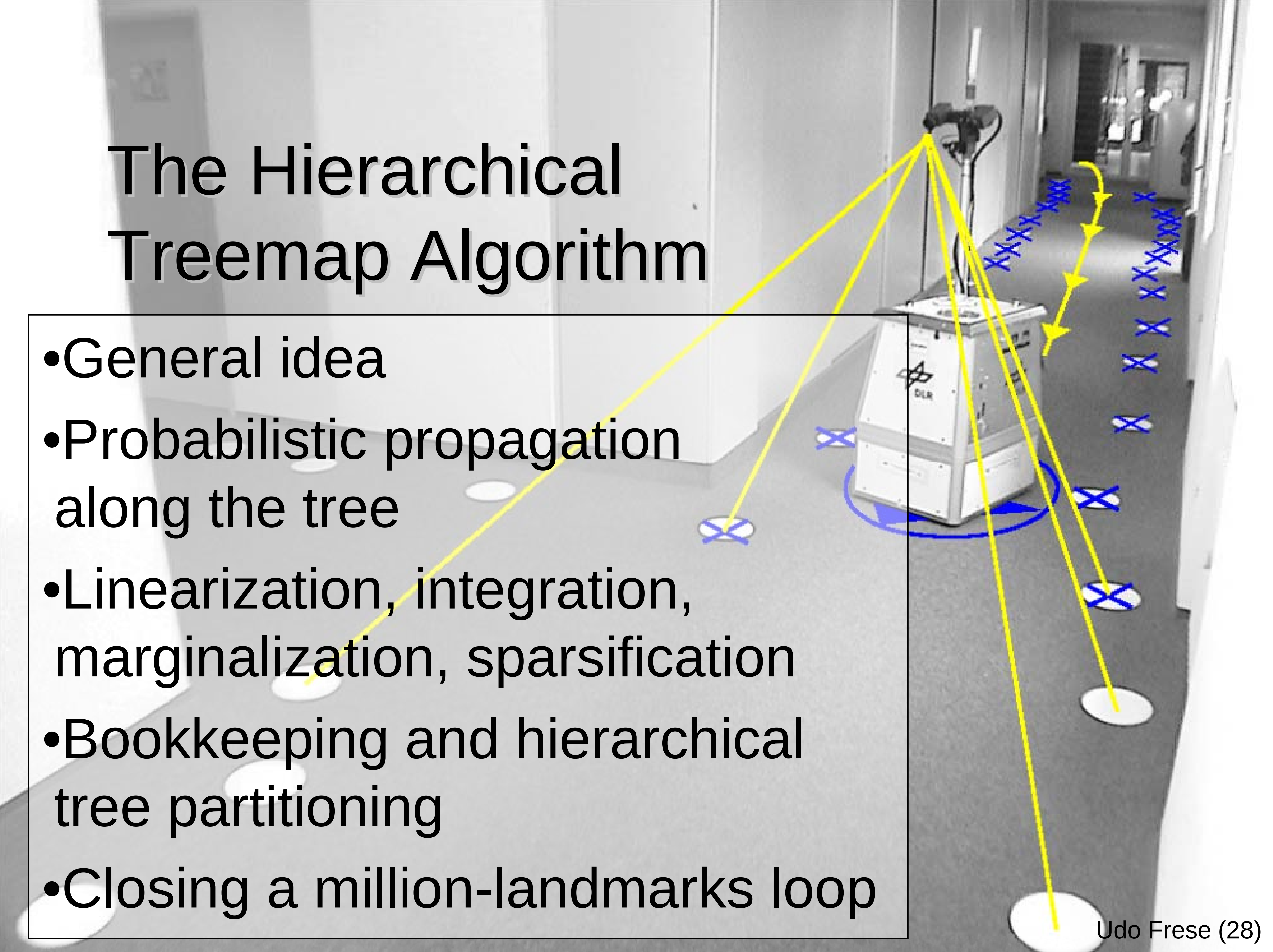
Practical Highlight

The Hierarchical Treemap Algorithm



[1] U. Frese (2006). An O(log n) Algorithm for Indoor Simultaneous Localization and Mapping, In Autonomous Robots

[2] U. Frese, L. Schröder (2006). Closing a Million-Landmarks Loop, IROS Beijing

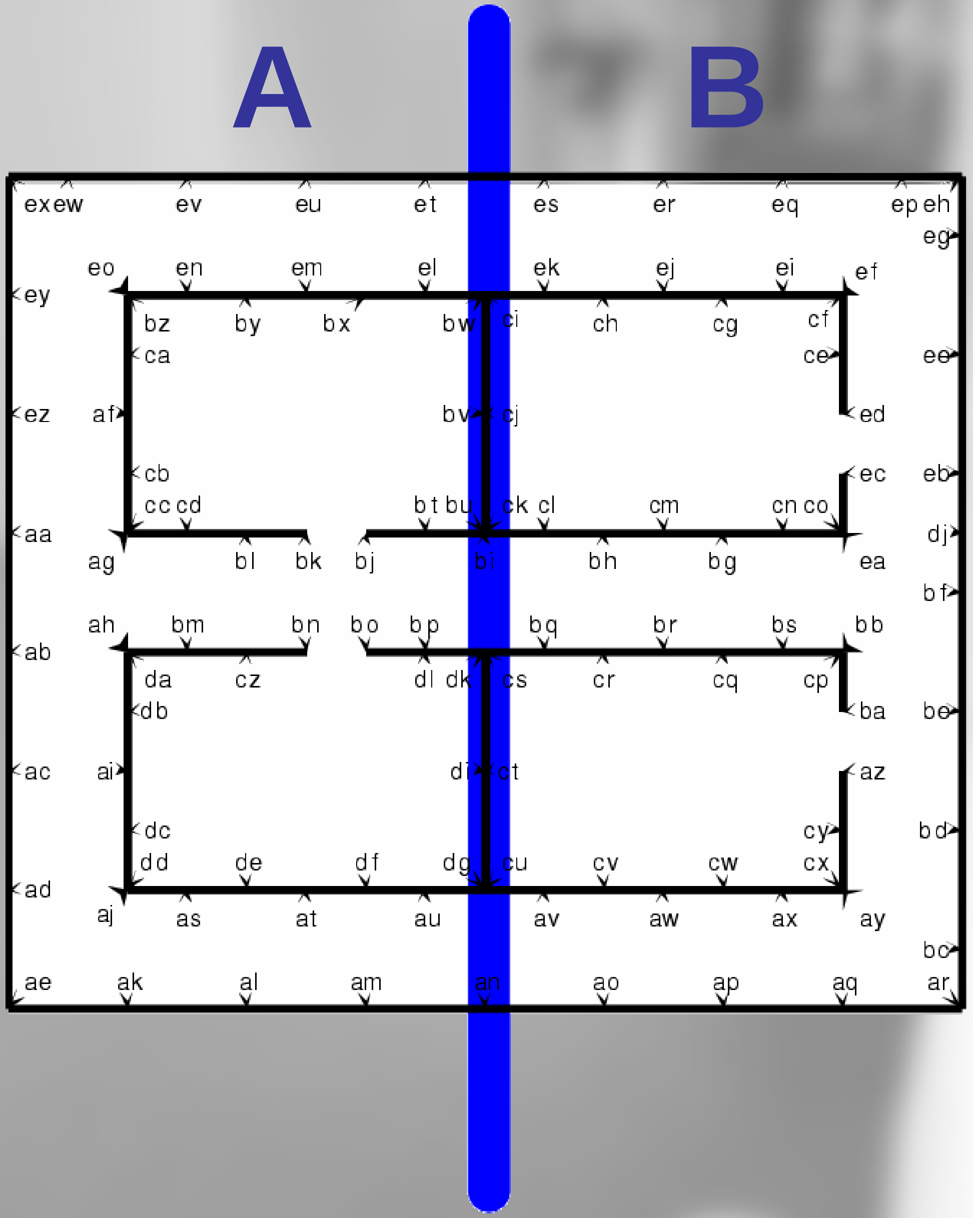


The Hierarchical Treemap Algorithm

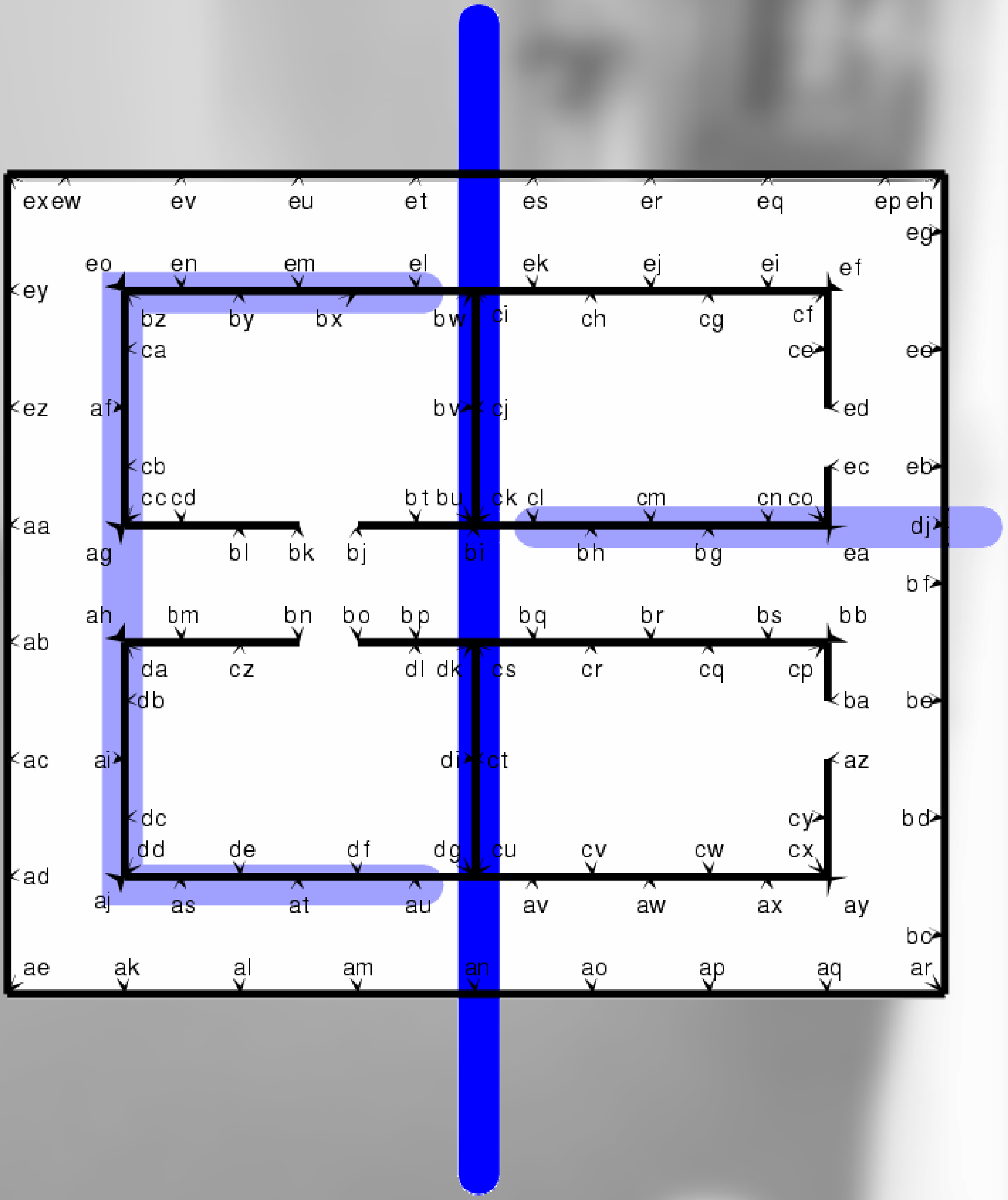
- General idea
- Probabilistic propagation along the tree
- Linearization, integration, marginalization, sparsification
- Bookkeeping and hierarchical tree partitioning
- Closing a million-landmarks loop

Treemap Algorithm

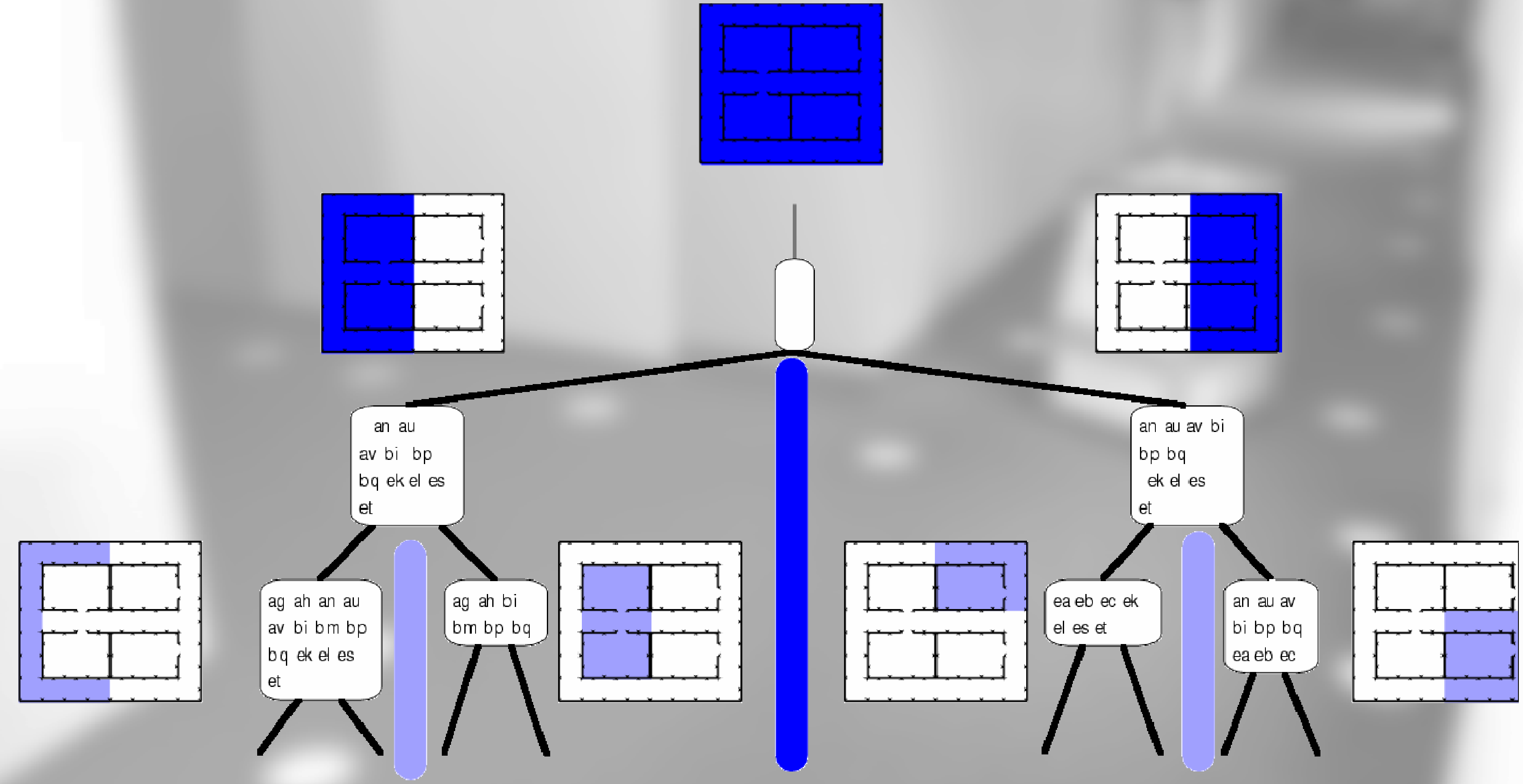
- If the robot is in part A, what is the information needed about B?
- Only the *marginal distribution* of landmarks observable from A conditioned on observations in B.



Treemap Algorithm



Treemap Algorithm



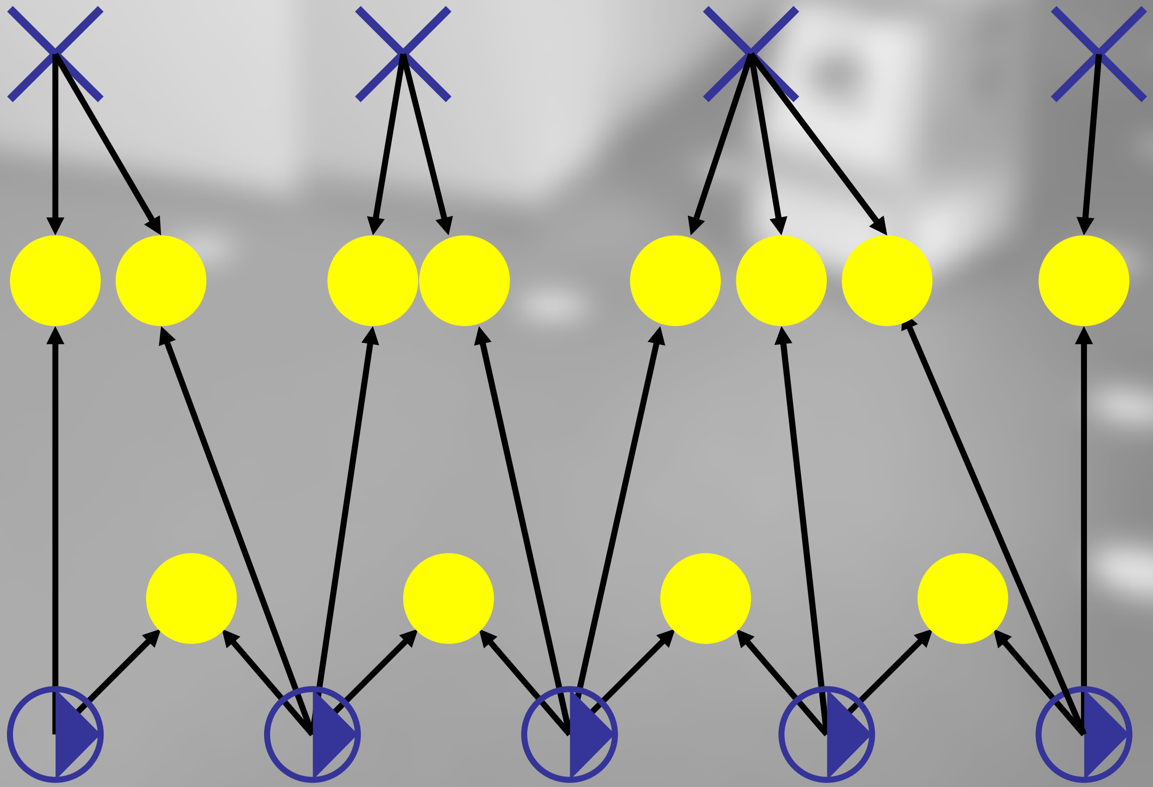
Treemap Algorithm

landmarks

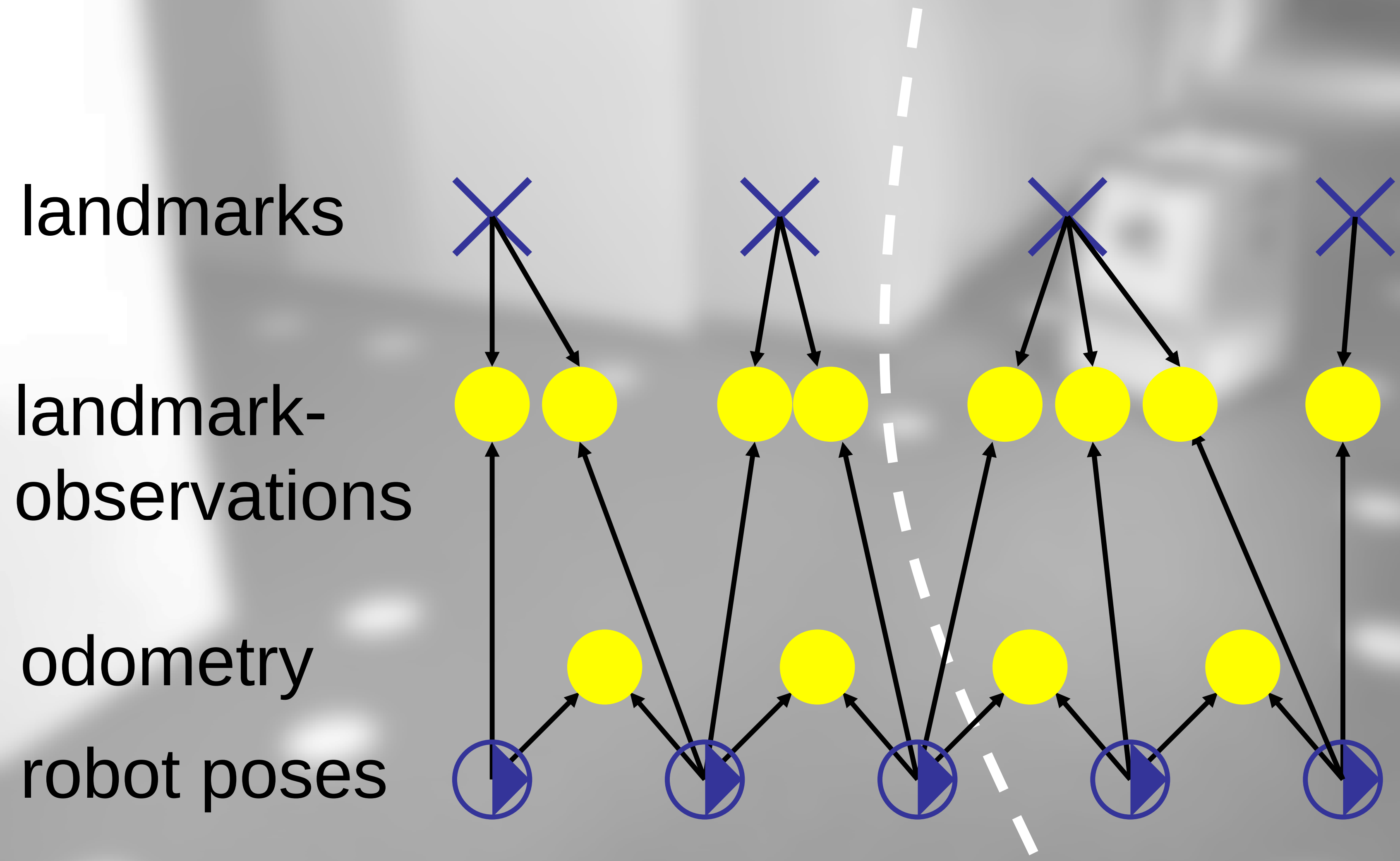
landmark-observations

odometry

robot poses



Treemap Algorithm



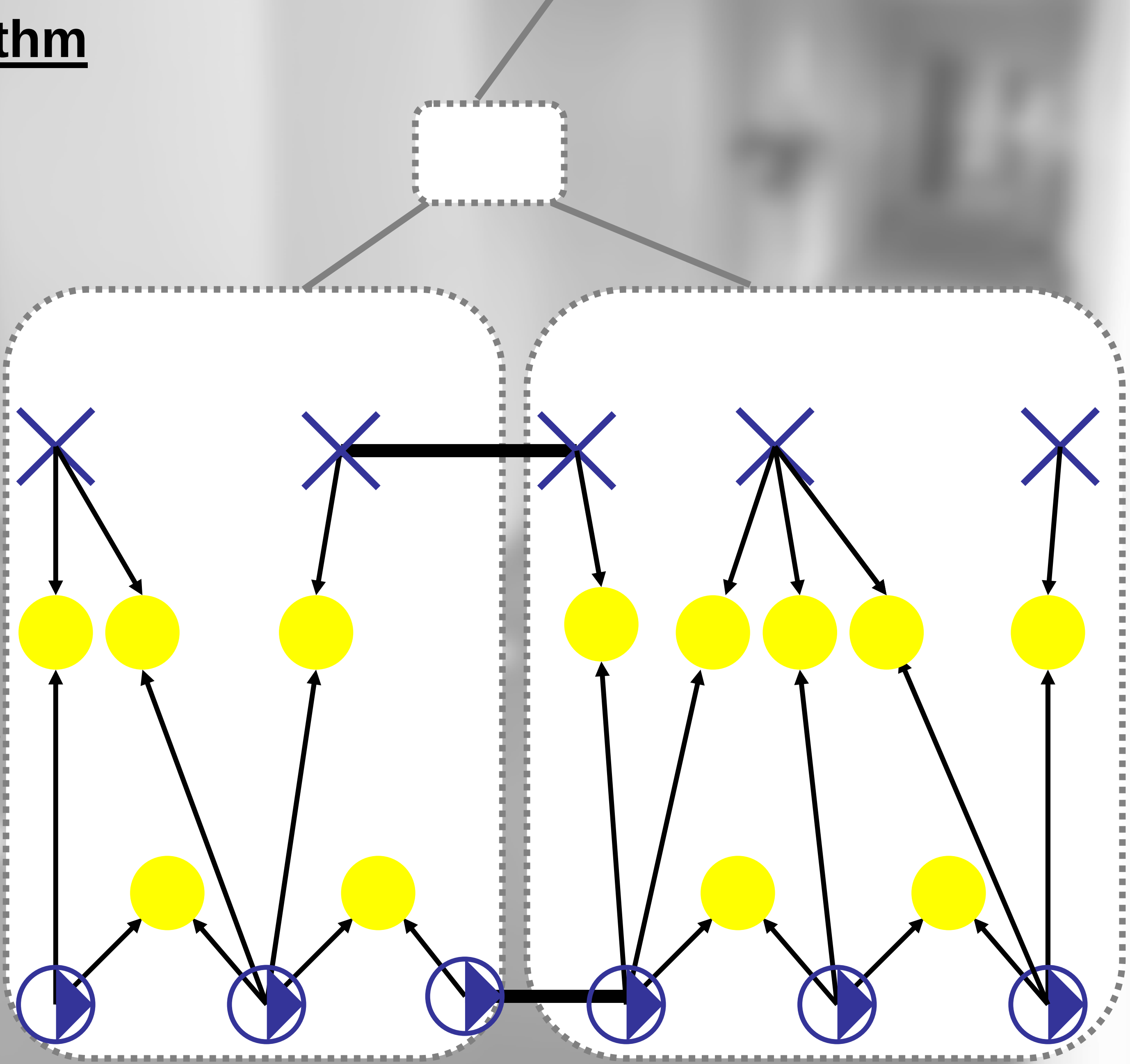
Treemap Algorithm

landmarks

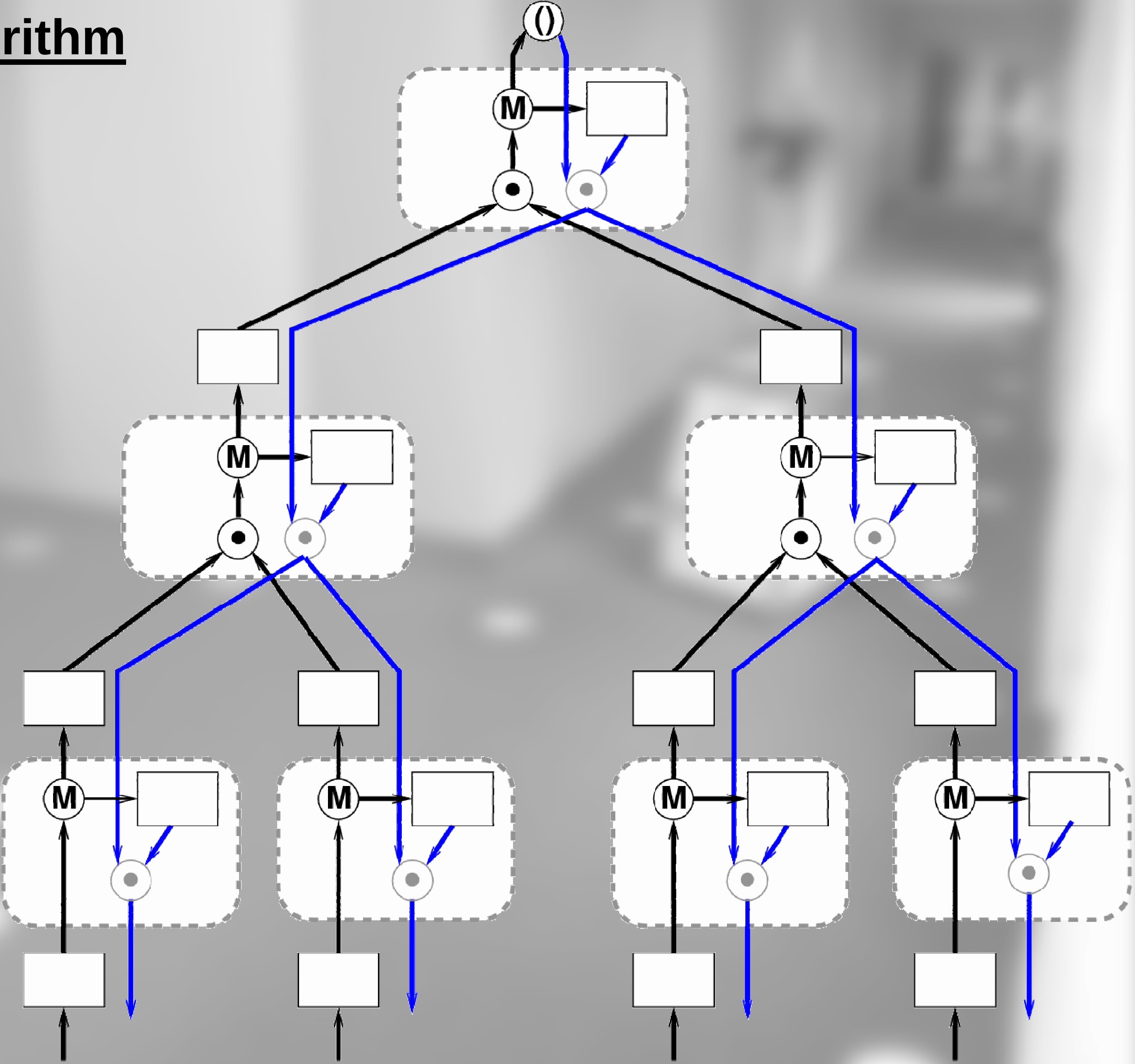
landmark-observations

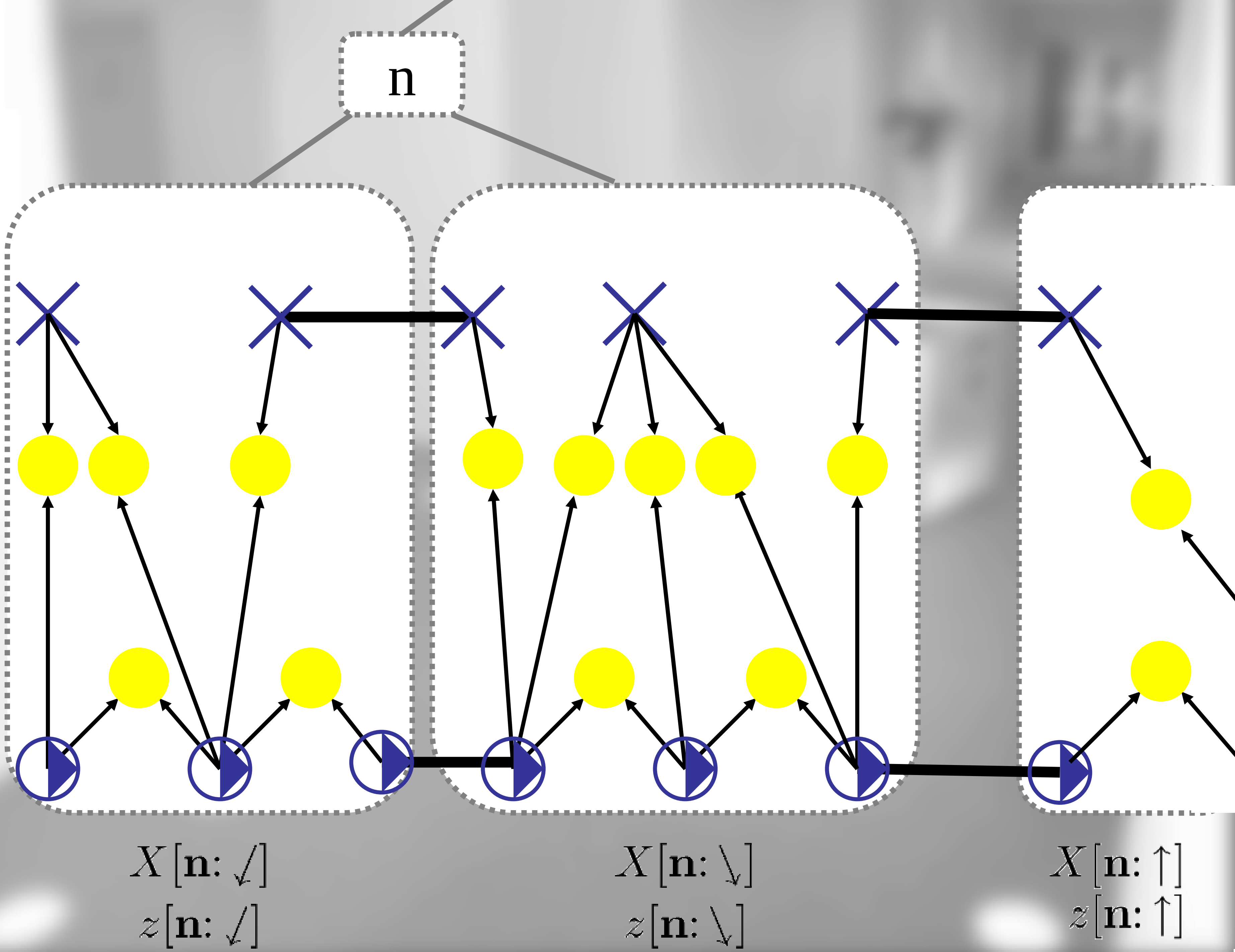
odometry

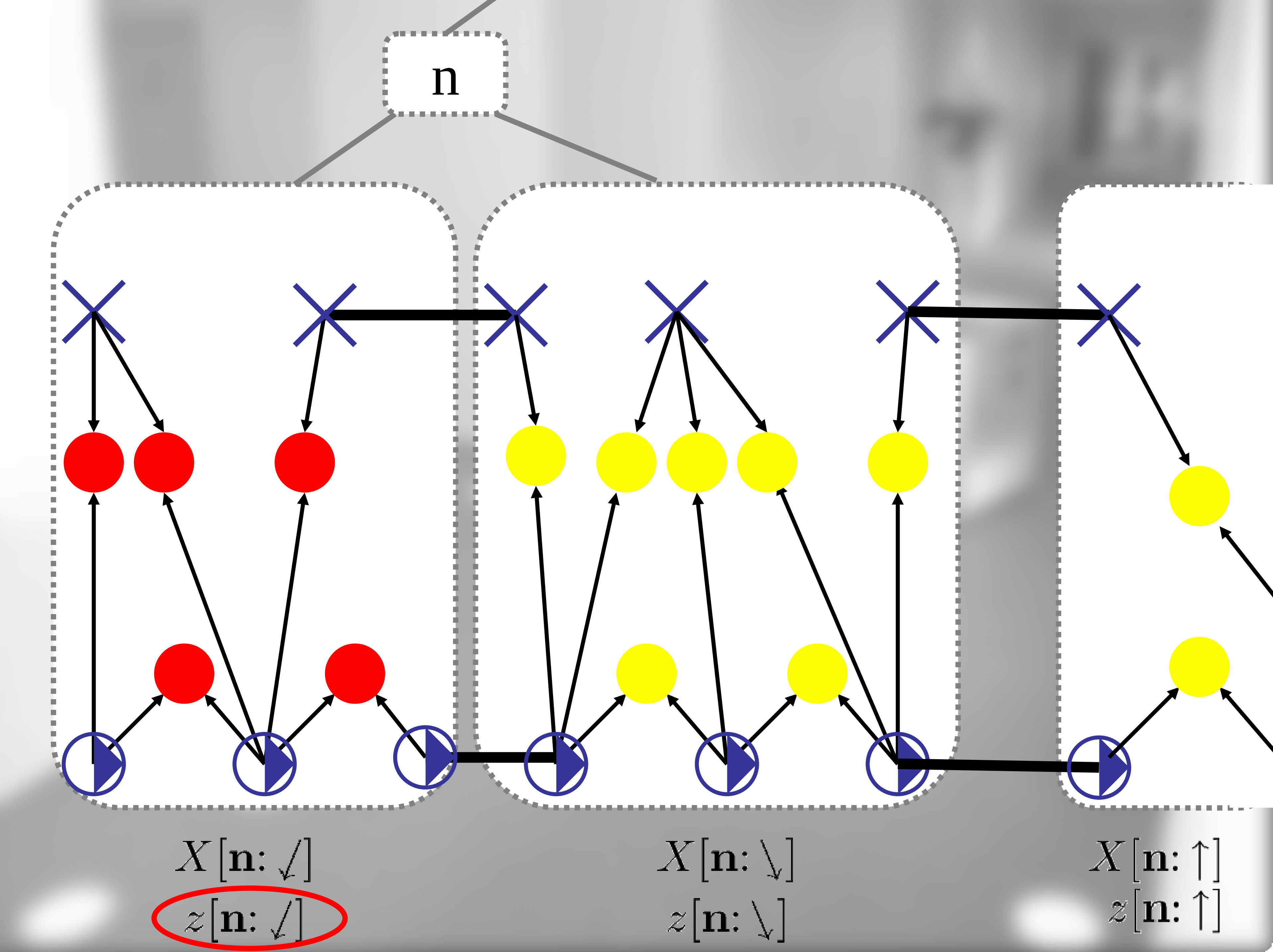
robot poses

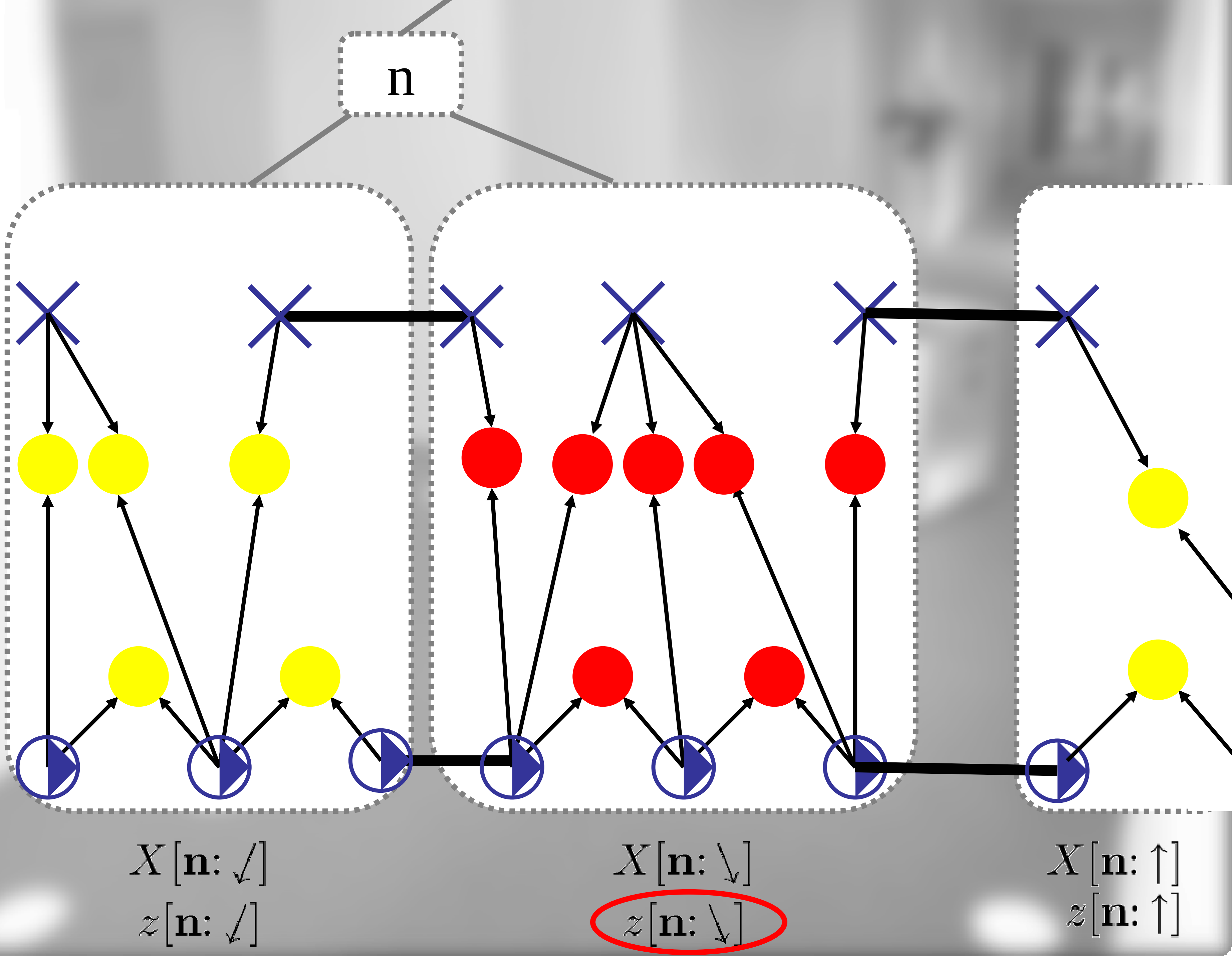


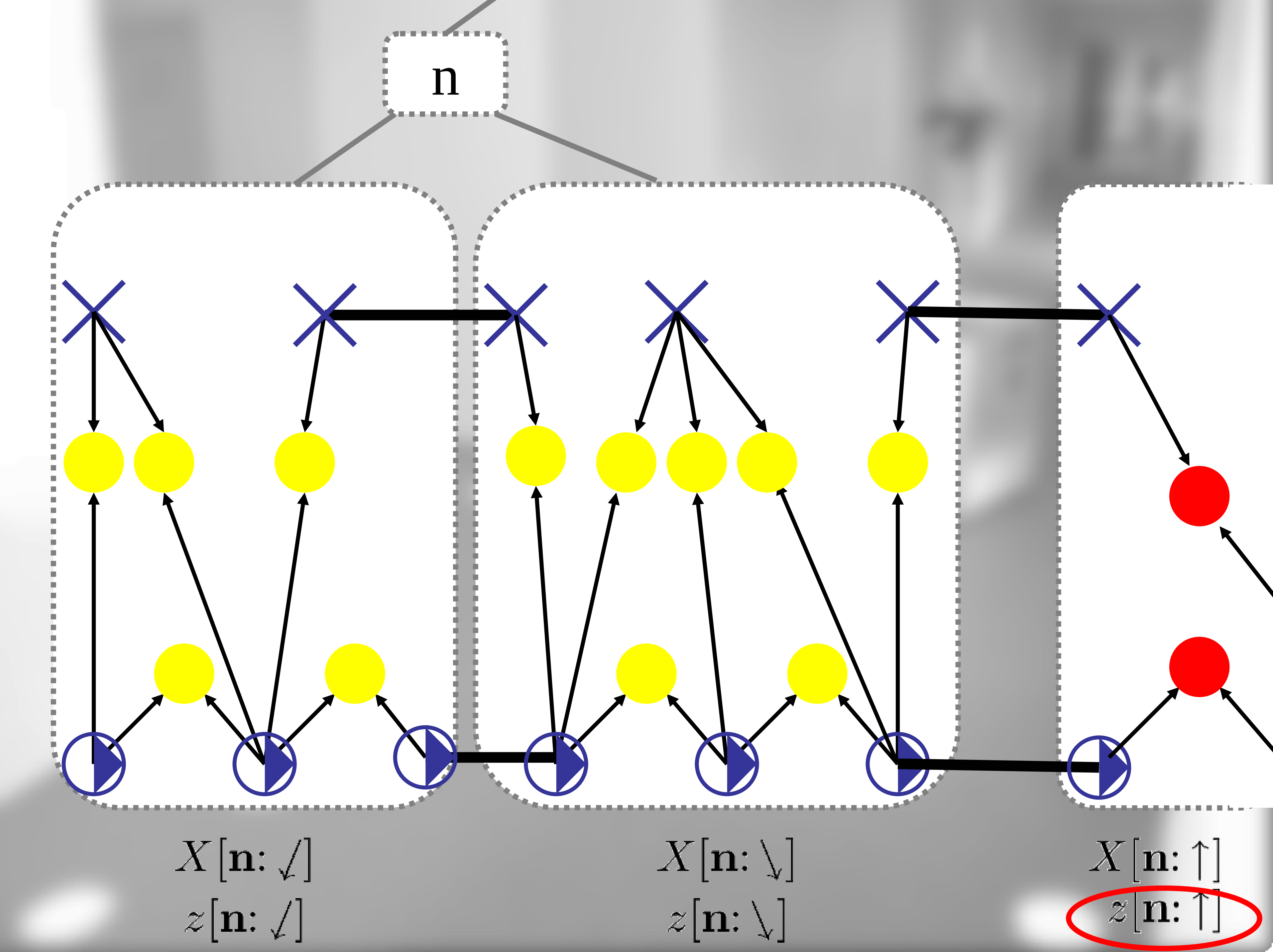
Treemap Algorithm

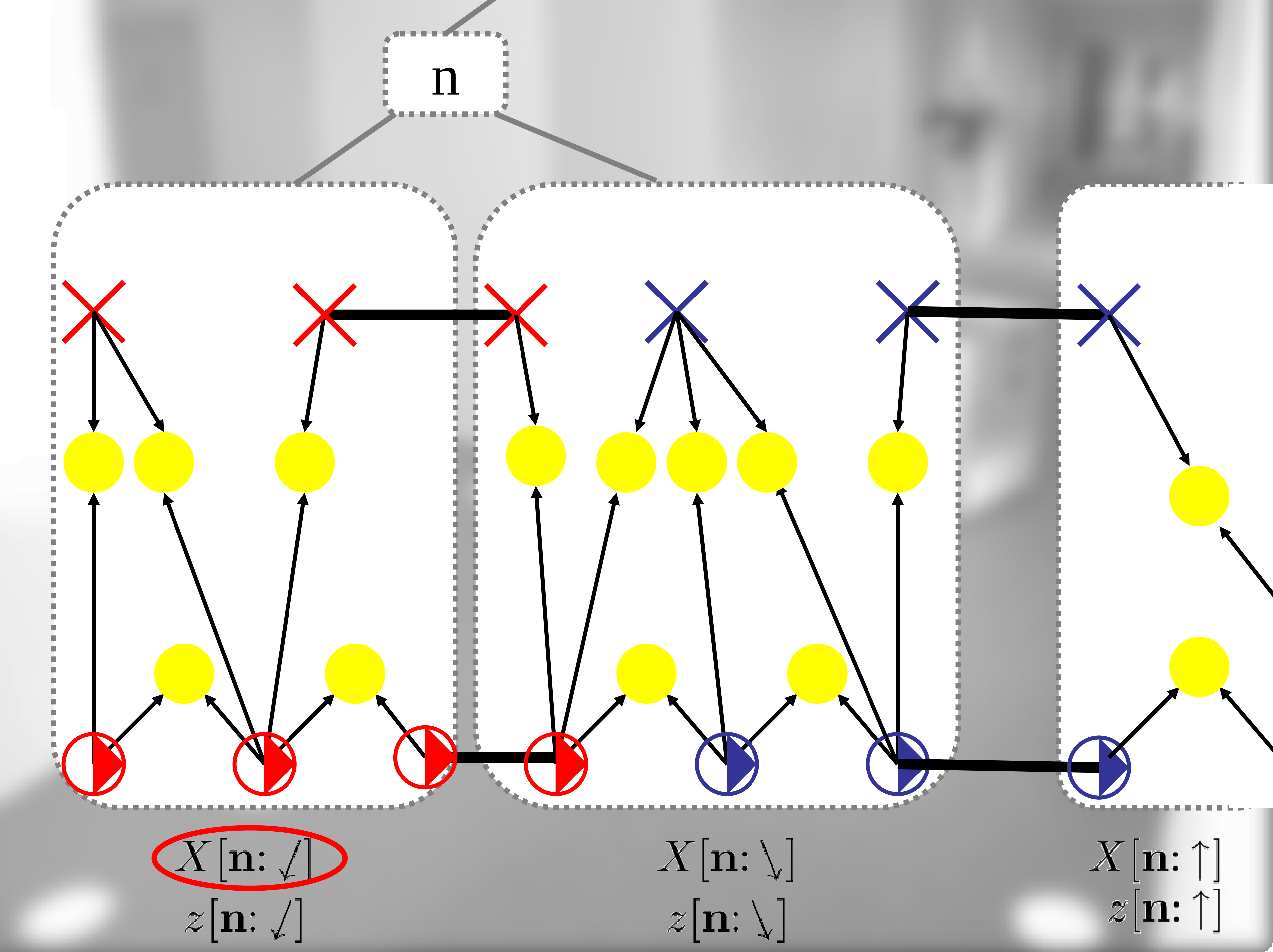


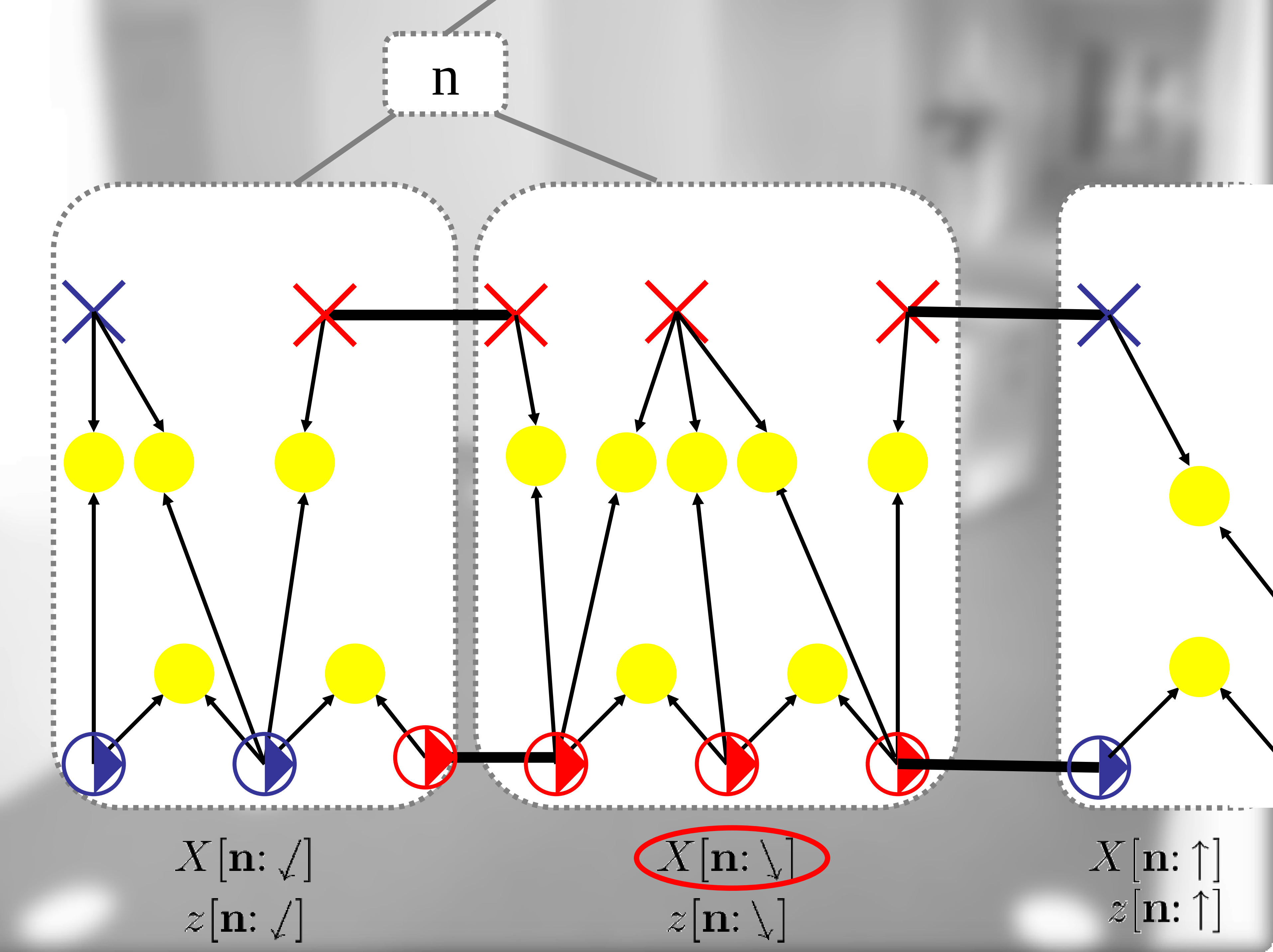

n
$X[\mathbf{n}\colon \swarrow]$
$z[\mathbf{n}\colon \swarrow]$
$X[\mathbf{n}\colon \searrow]$
$z[\mathbf{n}\colon \searrow]$
$X[\mathbf{n}\colon \uparrow]$
$z[\mathbf{n}\colon \uparrow]$

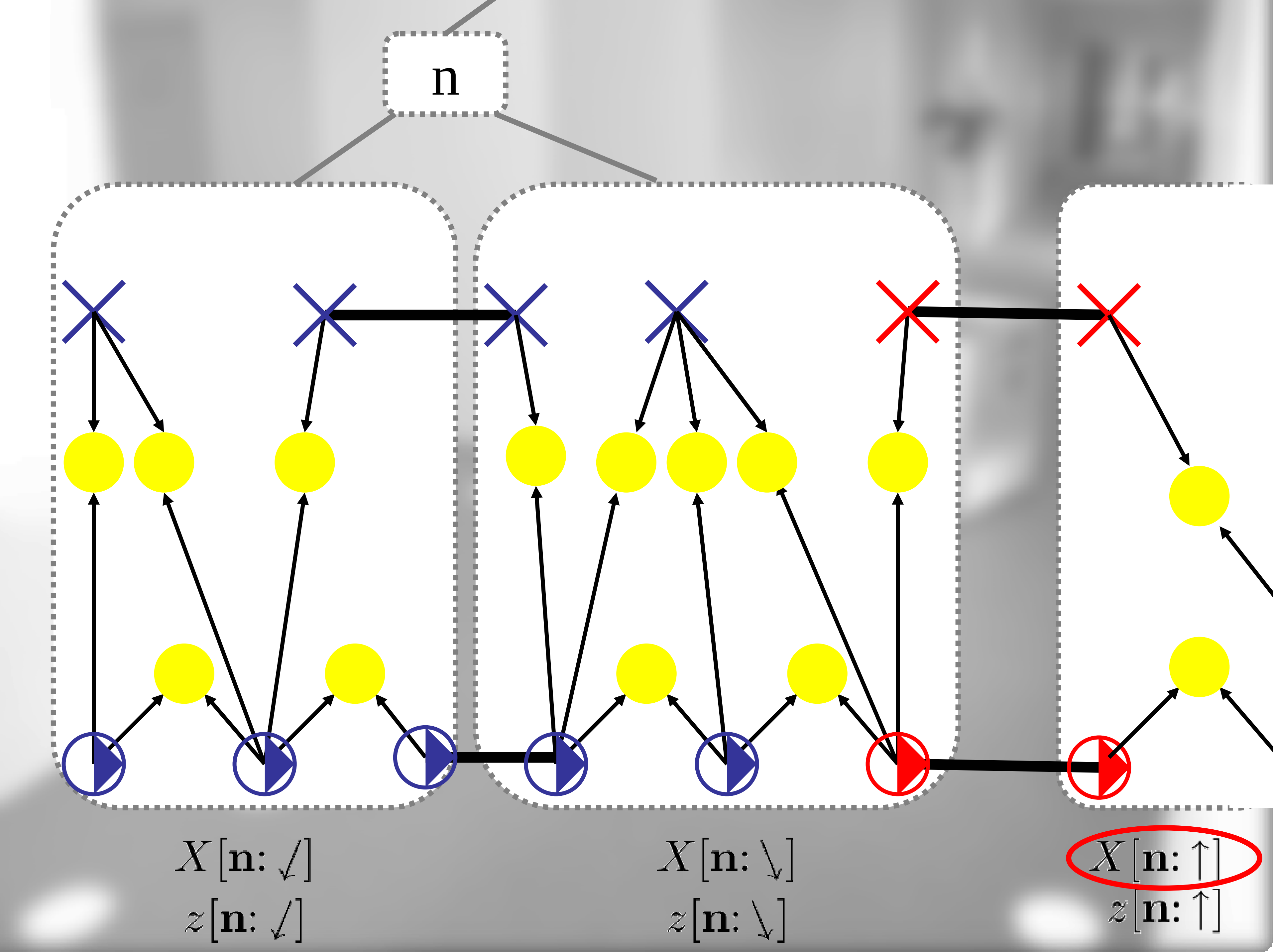

n
$X[\mathbf{n}: \swarrow]$
$z[\mathbf{n}: \swarrow]$
$X[\mathbf{n}: \searrow]$
$z[\mathbf{n}: \searrow]$
$X[\mathbf{n}: \uparrow]$
$z[\mathbf{n}: \uparrow]$


n
X[n: ↙]
z[n: ↙]
X[n: ↘]
z[n: ↘]
X[n: ↑]
z[n: ↑]


n
$X[\mathbf{n}\colon \swarrow]$
$z[\mathbf{n}\colon \swarrow]$
$X[\mathbf{n}\colon \searrow]$
$z[\mathbf{n}\colon \searrow]$
$X[\mathbf{n}\colon \uparrow]$
$z[\mathbf{n}\colon \uparrow]$


n
$X[\mathbf{n}\colon \swarrow]$
$z[\mathbf{n}\colon \swarrow]$
$X[\mathbf{n}\colon \searrow]$
$z[\mathbf{n}\colon \searrow]$
$X[\mathbf{n}\colon \uparrow]$
$z[\mathbf{n}\colon \uparrow]$


n
$X[\mathbf{n}\colon \swarrow]$
$z[\mathbf{n}\colon \swarrow]$
$X[\mathbf{n}\colon \searrow]$
$z[\mathbf{n}\colon \searrow]$
$X[\mathbf{n}\colon \uparrow]$
$z[\mathbf{n}\colon \uparrow]$

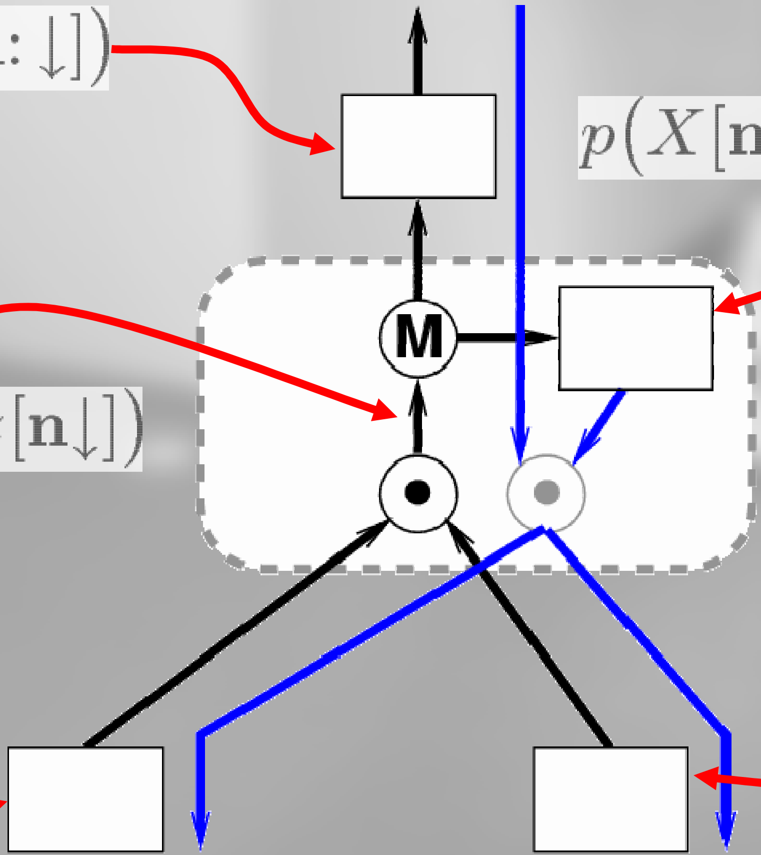

n
$X[\mathbf{n}\colon \swarrow]$
$z[\mathbf{n}\colon \swarrow]$
$X[\mathbf{n}\colon \searrow]$
$z[\mathbf{n}\colon \searrow]$
$X[\mathbf{n}\colon \uparrow]$
$z[\mathbf{n}\colon \uparrow]$

Treemap Algorithm

$$p\left(X[\mathbf{n}\colon\downarrow\uparrow] \,\middle|\, z[\mathbf{n}\colon\downarrow]\right)$$

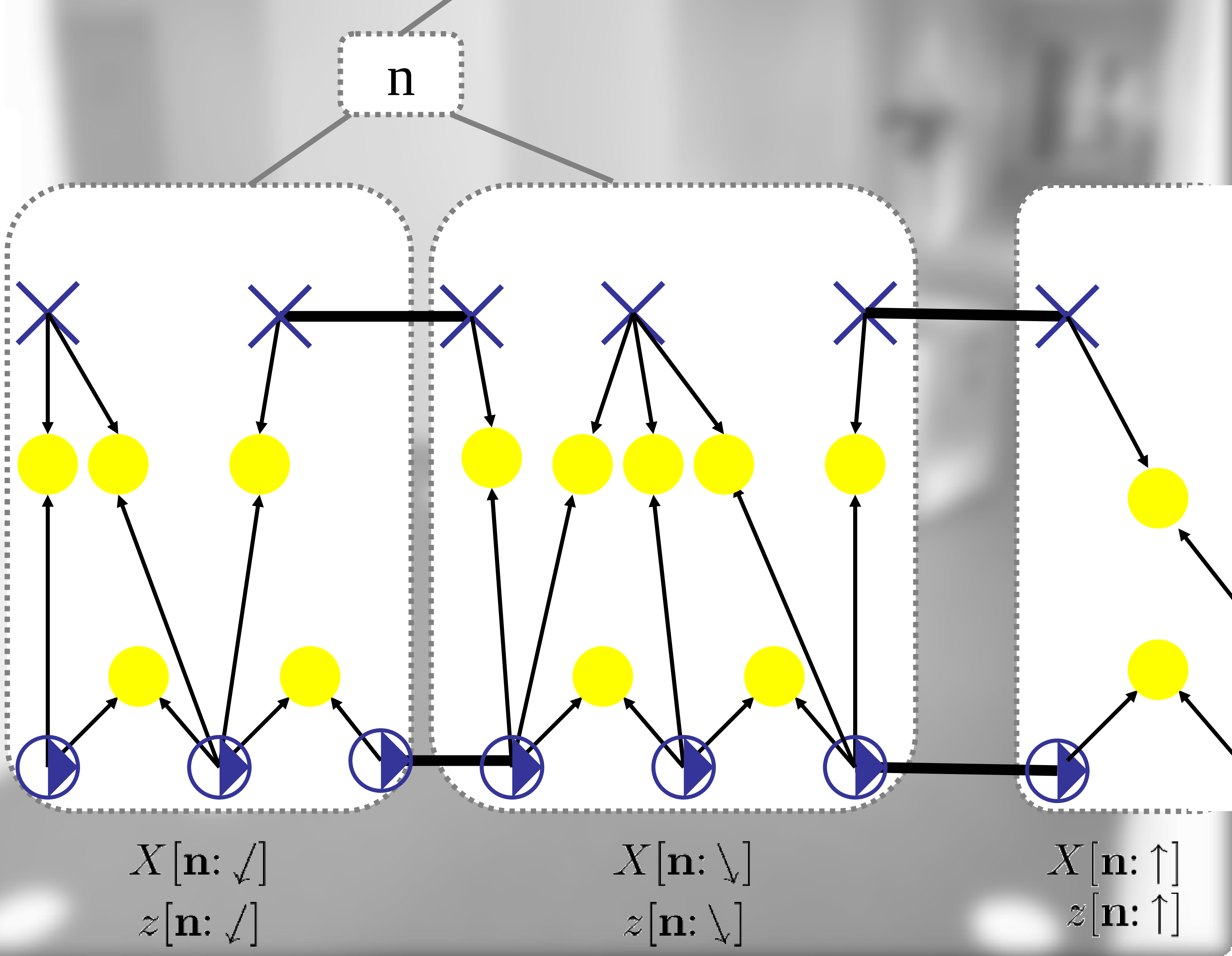
$$p\left(X[\mathbf{n}\colon \swarrow\searrow\not\uparrow] \,\middle|\, X[\mathbf{n}\colon\downarrow\uparrow], z\right)$$

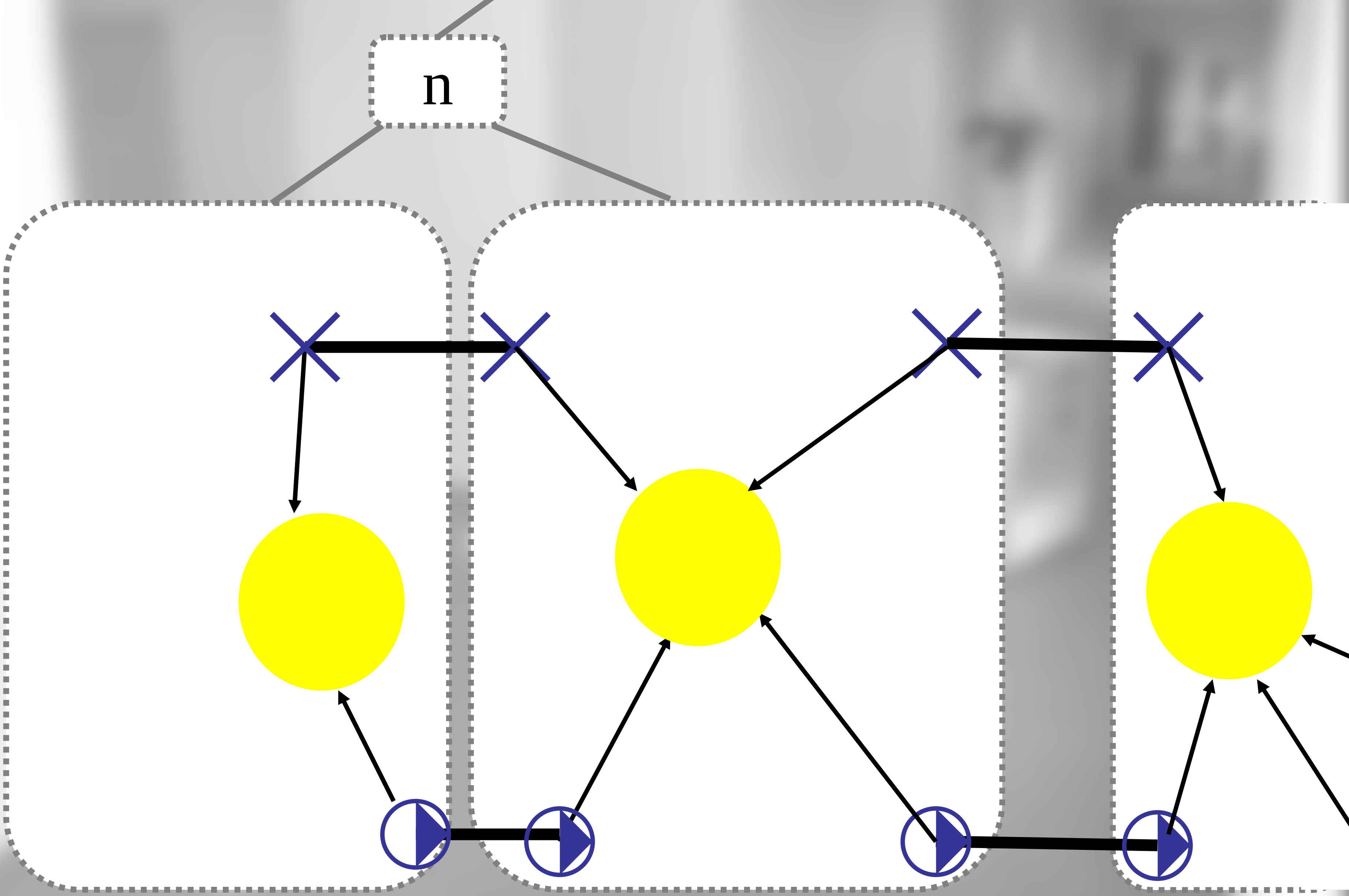
$$p\left(X[\mathbf{n}\colon\downarrow\uparrow \vee \swarrow\searrow\not\uparrow] \,\middle|\, z[\mathbf{n}\downarrow]\right)$$



$$p\left(X[\mathbf{n}_{\swarrow}\colon\downarrow\uparrow] \,\middle|\, z[\mathbf{n}_{\swarrow}\colon\downarrow]\right) = p\left(X[\mathbf{n}\colon\swarrow\uparrow \vee \swarrow\searrow\not\uparrow] \,\middle|\, z[\mathbf{n}\colon\swarrow]\right)$$

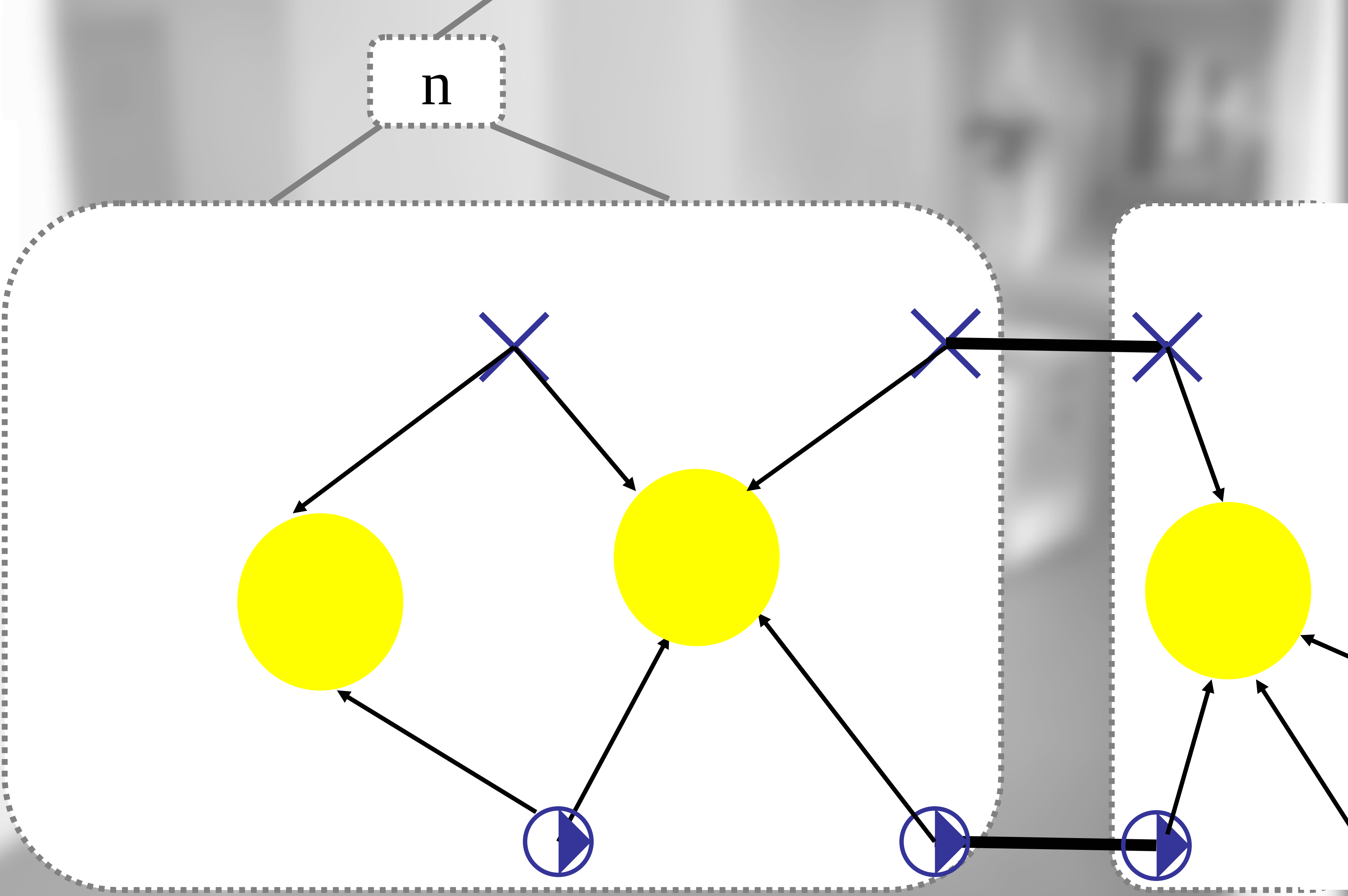
$$p\left(X[\mathbf{n}_{\searrow}\colon\downarrow\uparrow] \,\middle|\, z[\mathbf{n}_{\searrow}\colon\downarrow]\right) = p\left(X[\mathbf{n}\colon\searrow\uparrow \vee \swarrow\searrow\not\uparrow] \,\middle|\, z[\mathbf{n}\colon\searrow]\right)$$


n
$X[\mathbf{n}: \swarrow]$
$z[\mathbf{n}: \swarrow]$
$X[\mathbf{n}: \searrow]$
$z[\mathbf{n}: \searrow]$
$X[\mathbf{n}: \uparrow]$
$z[\mathbf{n}: \uparrow]$

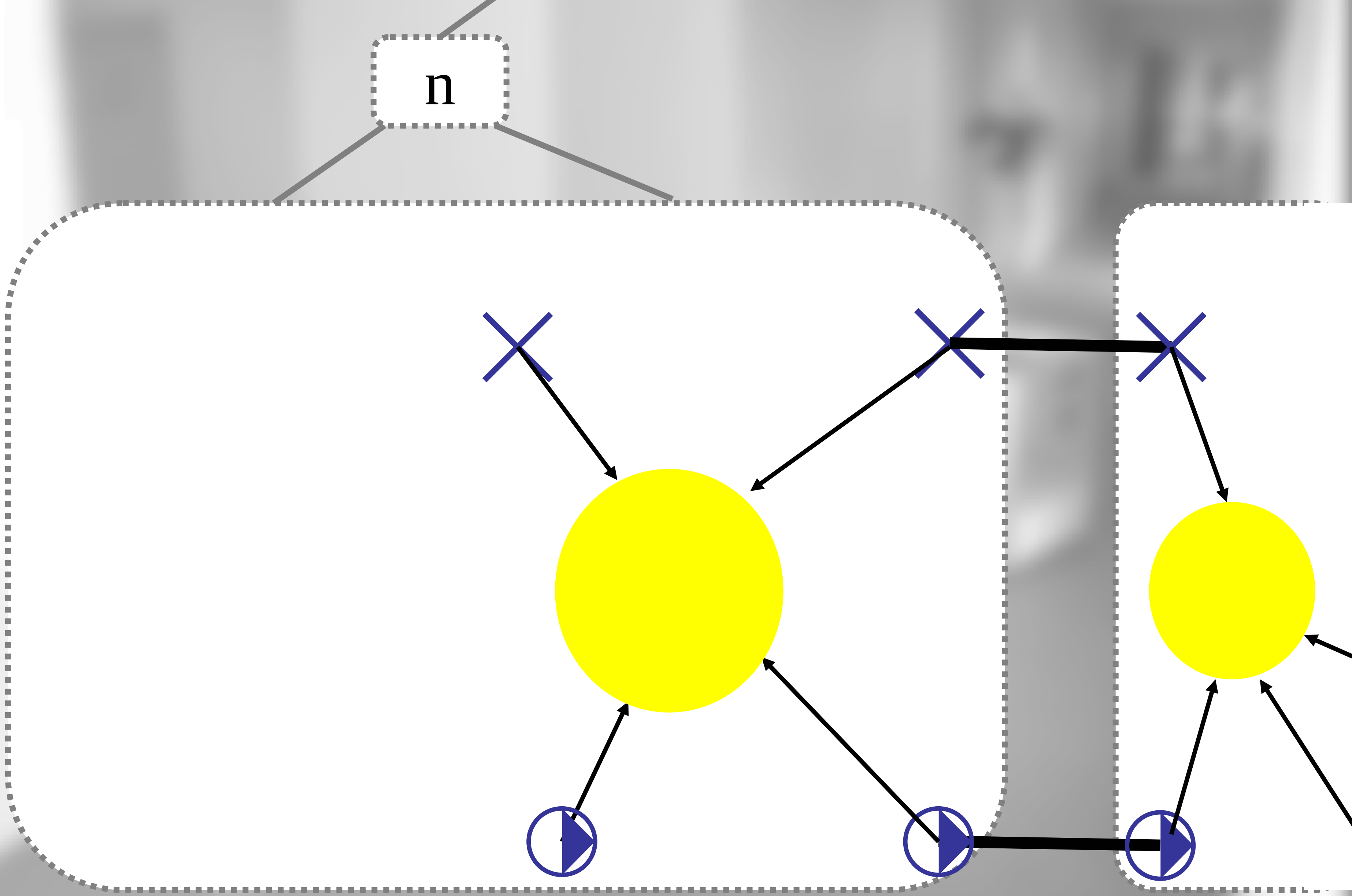


$$p\big(X[\mathbf{n}_{\swarrow}:\downarrow\uparrow]\,\big|\,z[\mathbf{n}_{\swarrow}:\downarrow]\big) = p\big(X[\mathbf{n}:\swarrow\uparrow\vee\nearrow\searrow\Uparrow]\,\big|\,z[\mathbf{n}:\swarrow]\big)$$

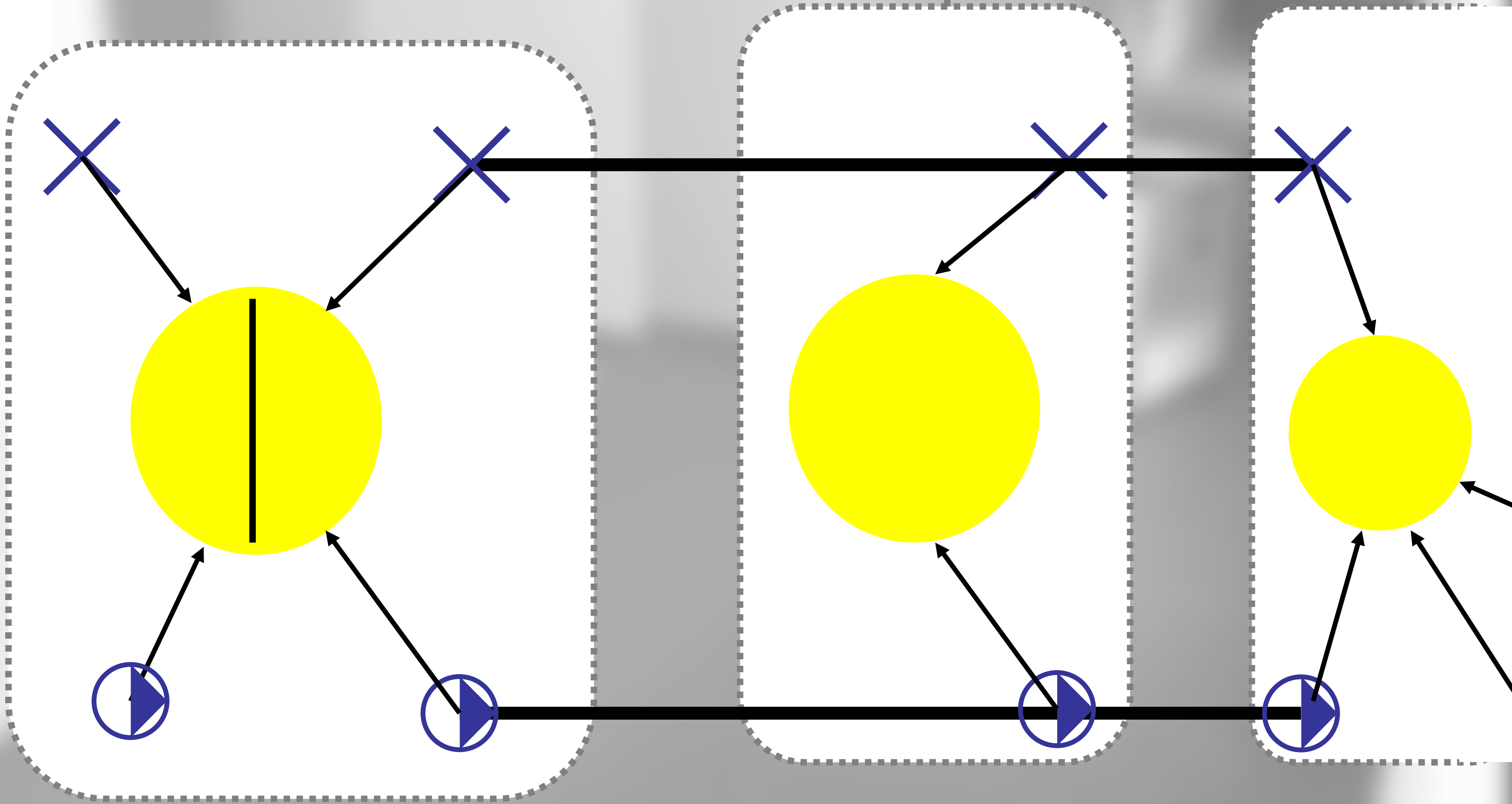
$$p\big(X[\mathbf{n}_{\searrow}:\downarrow\uparrow]\,\big|\,z[\mathbf{n}_{\searrow}:\downarrow]\big) = p\big(X[\mathbf{n}:\searrow\uparrow\vee\nearrow\searrow\Uparrow]\,\big|\,z[\mathbf{n}:\searrow]\big)$$



$$p\big(X[\mathbf{n}\colon \downarrow\uparrow \vee \swarrow\searrow \Uparrow] \,\big|\, z[\mathbf{n}\downarrow]\big)$$



$$p\big(X[\mathbf{n}{:}\downarrow\uparrow \vee \swarrow\searrow \not\uparrow]\,\big|\,z[\mathbf{n}\downarrow]\big)$$



$p\left(X[\mathbf{n}\colon \swarrow\searrow\upuparrows] \middle| X[\mathbf{n}\colon \downarrow\uparrow], z\right)$

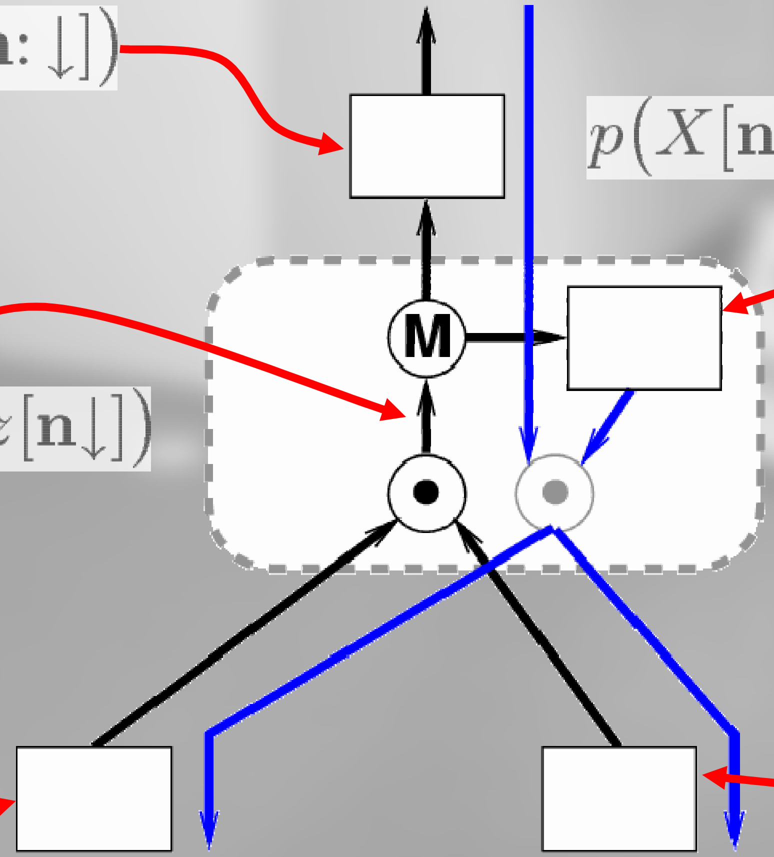
$p\left(X[\mathbf{n}\colon \downarrow\uparrow] \middle| z[\mathbf{n}\colon \downarrow]\right)$

Treemap Algorithm

$p\left(X[\mathbf{n}\colon \downarrow\uparrow] \,\middle|\, z[\mathbf{n}\colon \downarrow]\right)$

$p\left(X[\mathbf{n}\colon \swarrow\searrow \not\uparrow] \,\middle|\, X[\mathbf{n}\colon \downarrow\uparrow], z\right)$

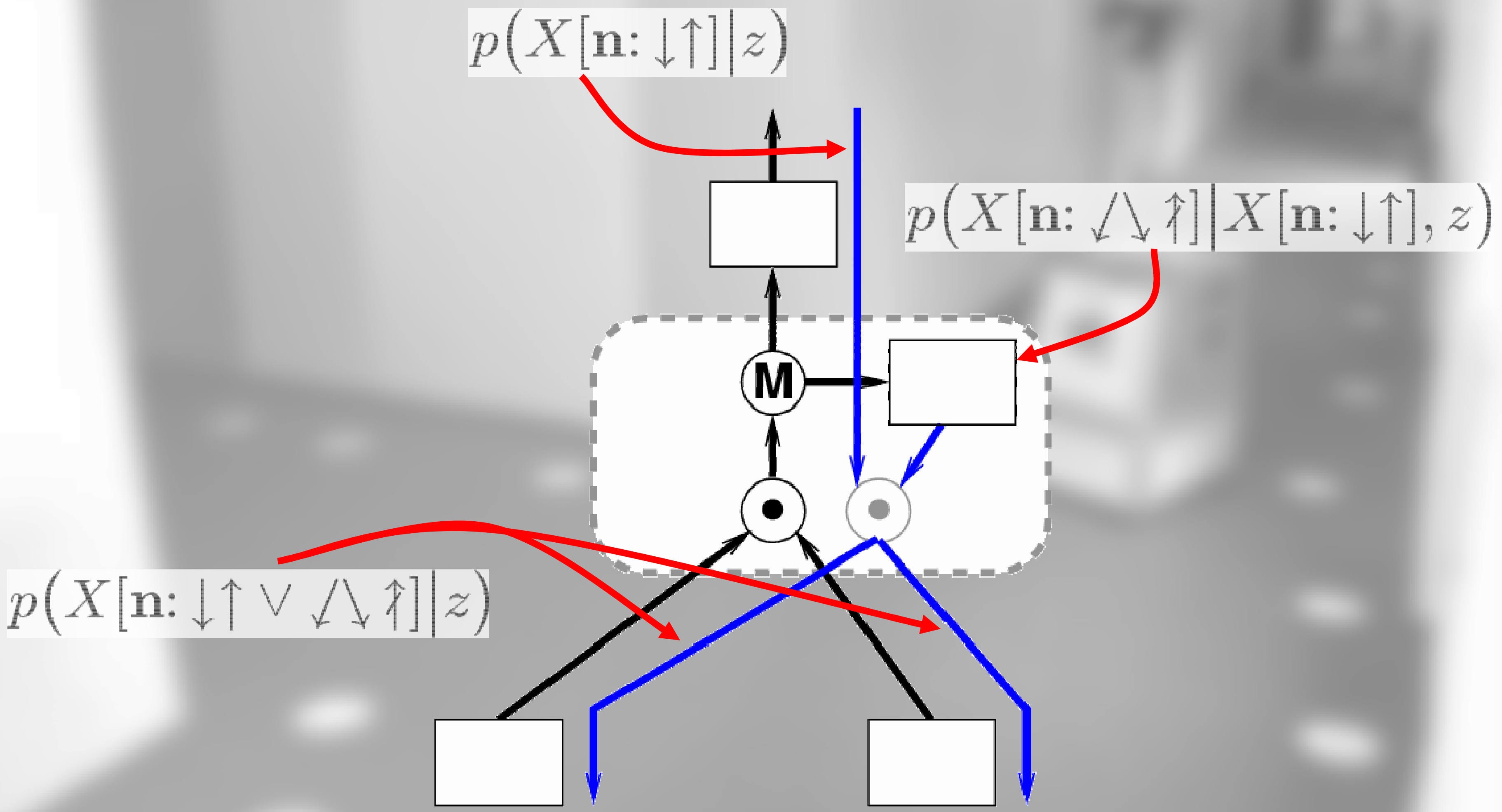
$p\left(X[\mathbf{n}\colon \downarrow\uparrow \vee \swarrow\searrow \not\uparrow] \,\middle|\, z[\mathbf{n}\downarrow]\right)$



$$p\left(X[\mathbf{n}_\swarrow\colon \downarrow\uparrow] \,\middle|\, z[\mathbf{n}_\swarrow\colon \downarrow]\right) = p\left(X[\mathbf{n}\colon \swarrow\uparrow \vee \swarrow\searrow \not\uparrow] \,\middle|\, z[\mathbf{n}\colon \swarrow]\right)$$

$$p\left(X[\mathbf{n}_\searrow\colon \downarrow\uparrow] \,\middle|\, z[\mathbf{n}_\searrow\colon \downarrow]\right) = p\left(X[\mathbf{n}\colon \searrow\uparrow \vee \swarrow\searrow \not\uparrow] \,\middle|\, z[\mathbf{n}\colon \searrow]\right)$$

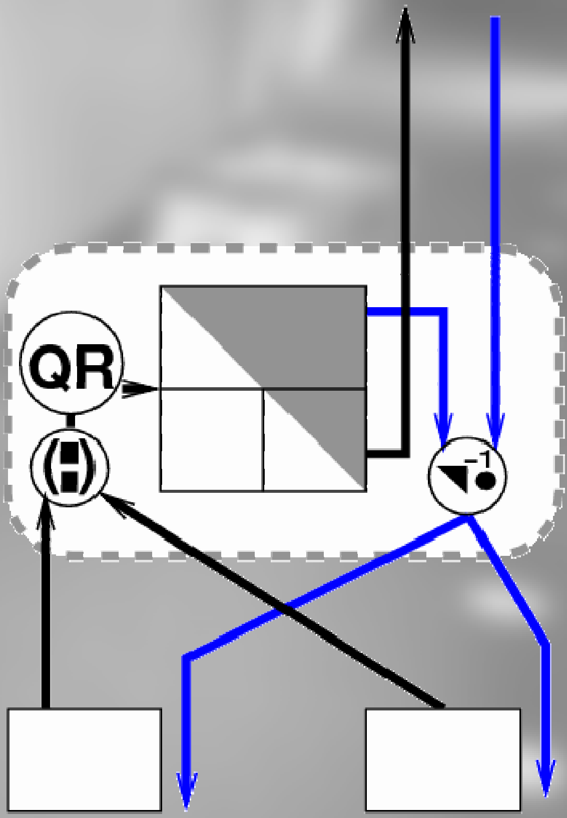
Treemap Algorithm


$p(X[\mathbf{n}{:}\downarrow\uparrow] \mid z)$
$p(X[\mathbf{n}{:}\swarrow\searrow\Uparrow] \mid X[\mathbf{n}{:}\downarrow\uparrow], z)$
M
$p(X[\mathbf{n}{:}\downarrow\uparrow \vee \swarrow\searrow\Uparrow] \mid z)$

Treemap Algorithm

Actual Implementation

- Gaussians defined by square-root information matrix.
- Upwards (•) by stacking.
- (M) by QR-decomposition
- Downwards (•) by back-substitution, i.e. solving a triangular equation system

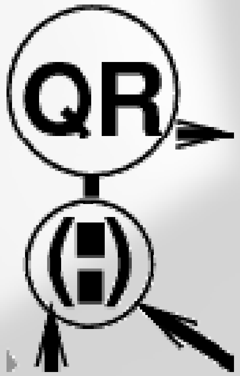


Treemap Algorithm

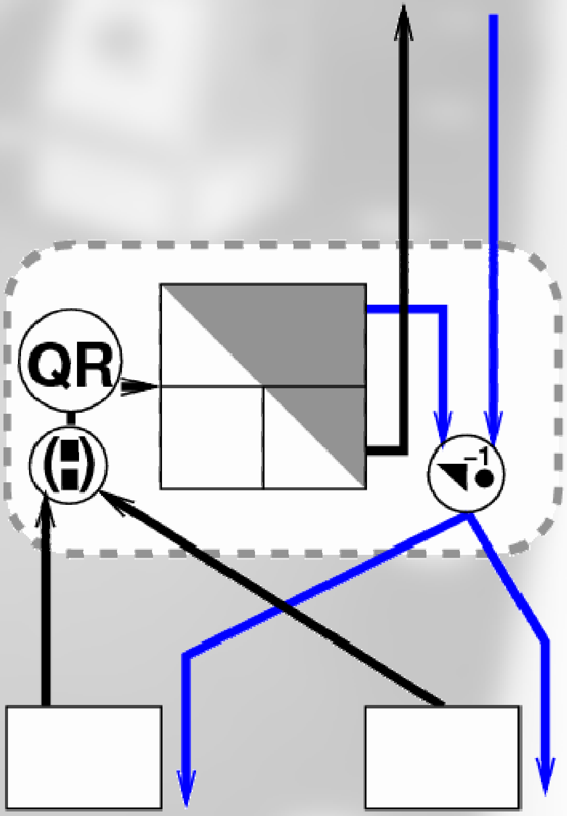
$$\chi^2(x) = x^T A x + x^T b + \gamma$$

$$= \begin{pmatrix} x \\ 1 \end{pmatrix}^T \underbrace{\begin{pmatrix} A & b/2 \\ b^T/2 & \gamma \end{pmatrix}}_{A'} \underbrace{\begin{pmatrix} x \\ 1 \end{pmatrix}}_{x'}$$

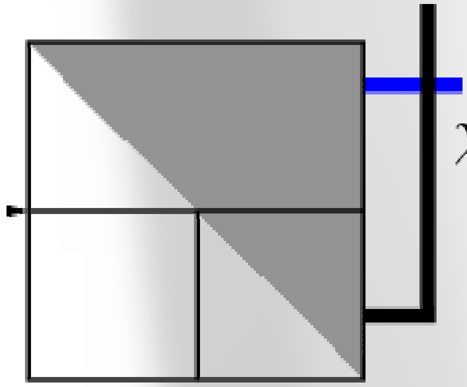
$$\chi^2(x) = \|Rx'\|_2^2, \quad A' = R^T R$$



$$\chi^2(x') = \chi_1^2(x') + \chi_2^2(x')$$
$$= \|R_1 x'\|_2^2 + \|R_2 x'\|_2^2$$
$$= \left\| \begin{pmatrix} R_1 \\ R_2 \end{pmatrix} x' \right\|_2^2$$
$$= \|Rx'\|_2^2, \quad \begin{pmatrix} R_1 \\ R_2 \end{pmatrix} = QR$$



Treemap Algorithm

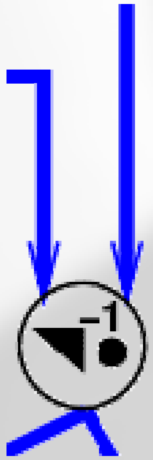


$$\chi^2\begin{pmatrix} y \\ z' \end{pmatrix} = \left\| \begin{pmatrix} C & D \\ 0 & E \end{pmatrix} \begin{pmatrix} y \\ z' \end{pmatrix} \right\|_2^2$$

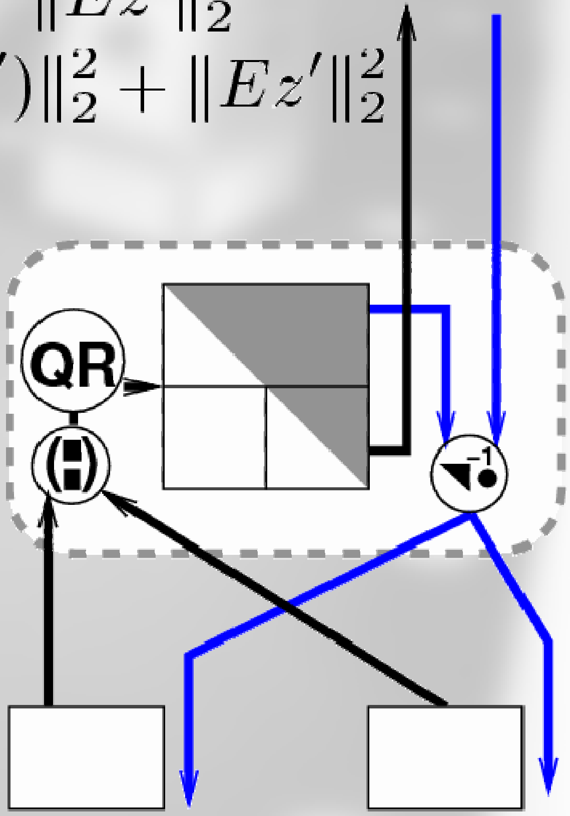
$$= \left\| \begin{pmatrix} C & D \end{pmatrix} \begin{pmatrix} y \\ z' \end{pmatrix} \right\|_2^2 + \left\| \begin{pmatrix} 0 & E \end{pmatrix} \begin{pmatrix} y \\ z' \end{pmatrix} \right\|_2^2$$

$$= \left\| \begin{pmatrix} C & D \end{pmatrix} \begin{pmatrix} y \\ z' \end{pmatrix} \right\|_2^2 + \| E z' \|_2^2$$

$$\chi^2\begin{pmatrix} y \\ z' \end{pmatrix} = \| C(y - C^{-1} D z') \|_2^2 + \| E z' \|_2^2$$



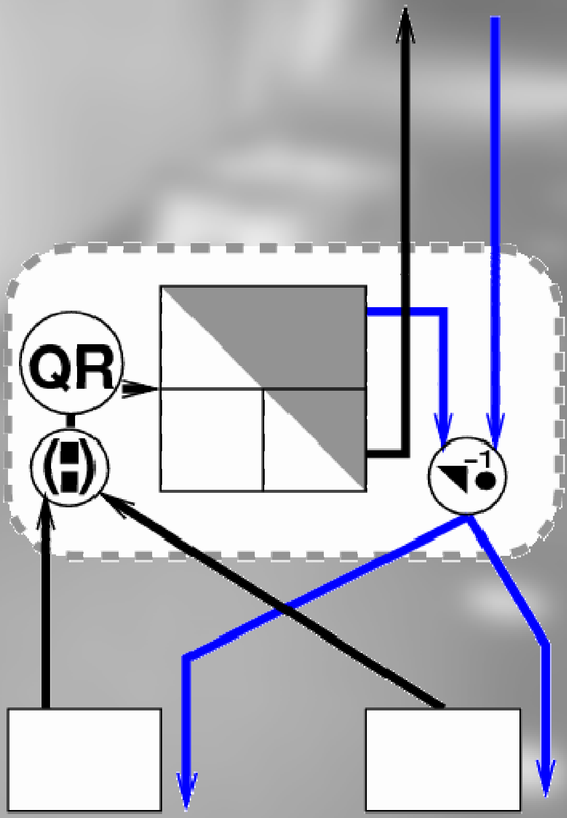
$$y_i = -\frac{1}{R_{ii}} \sum_{j=i+1}^{\dim y} R_{ij} y_j$$



Treemap Algorithm

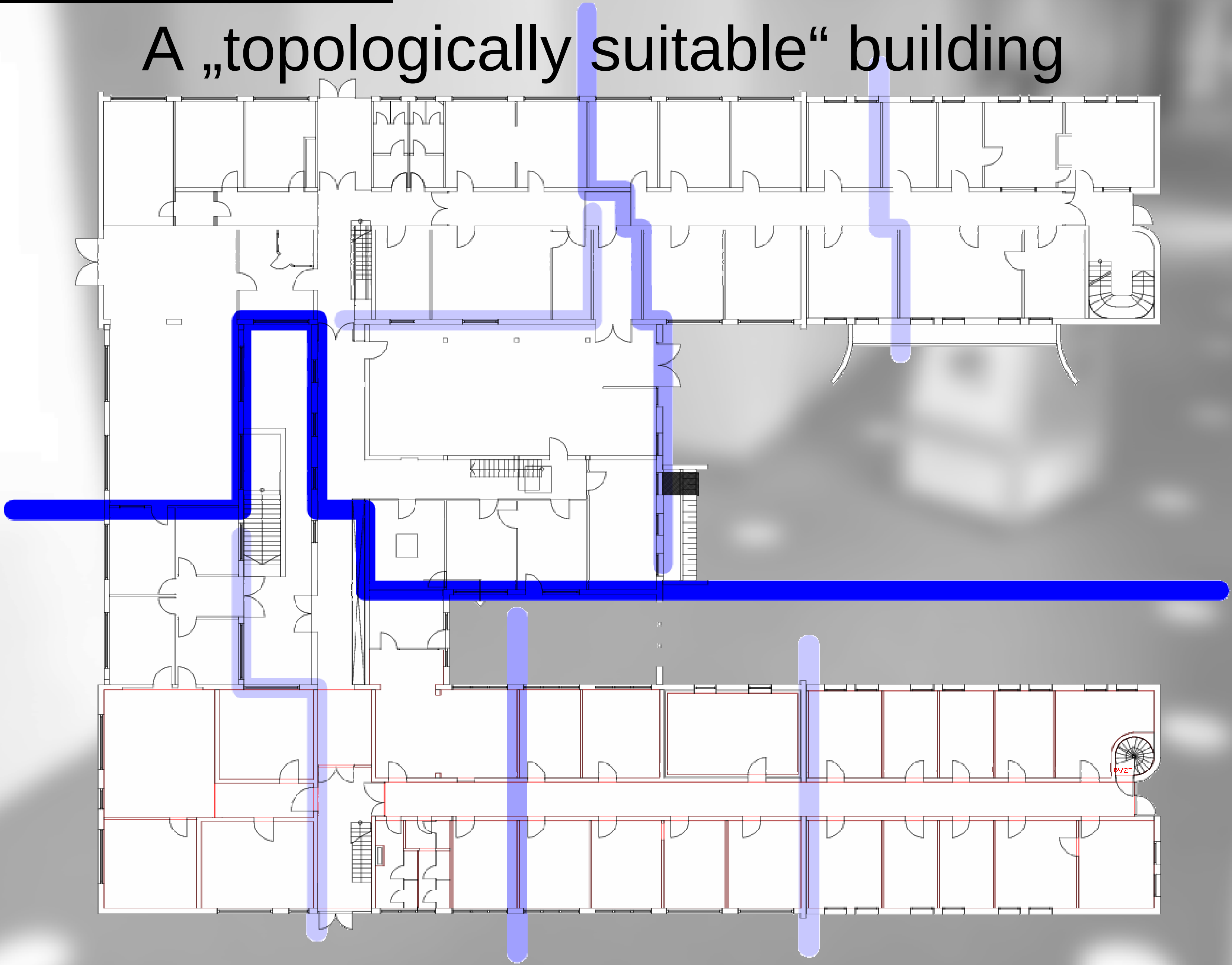
Why is it fast?

- Many small matrices instead of one large matrix.
- Update only O(log n) nodes upwards.
- Downwards (•) operation is extremely fast.
- Requires topologically suitable building.

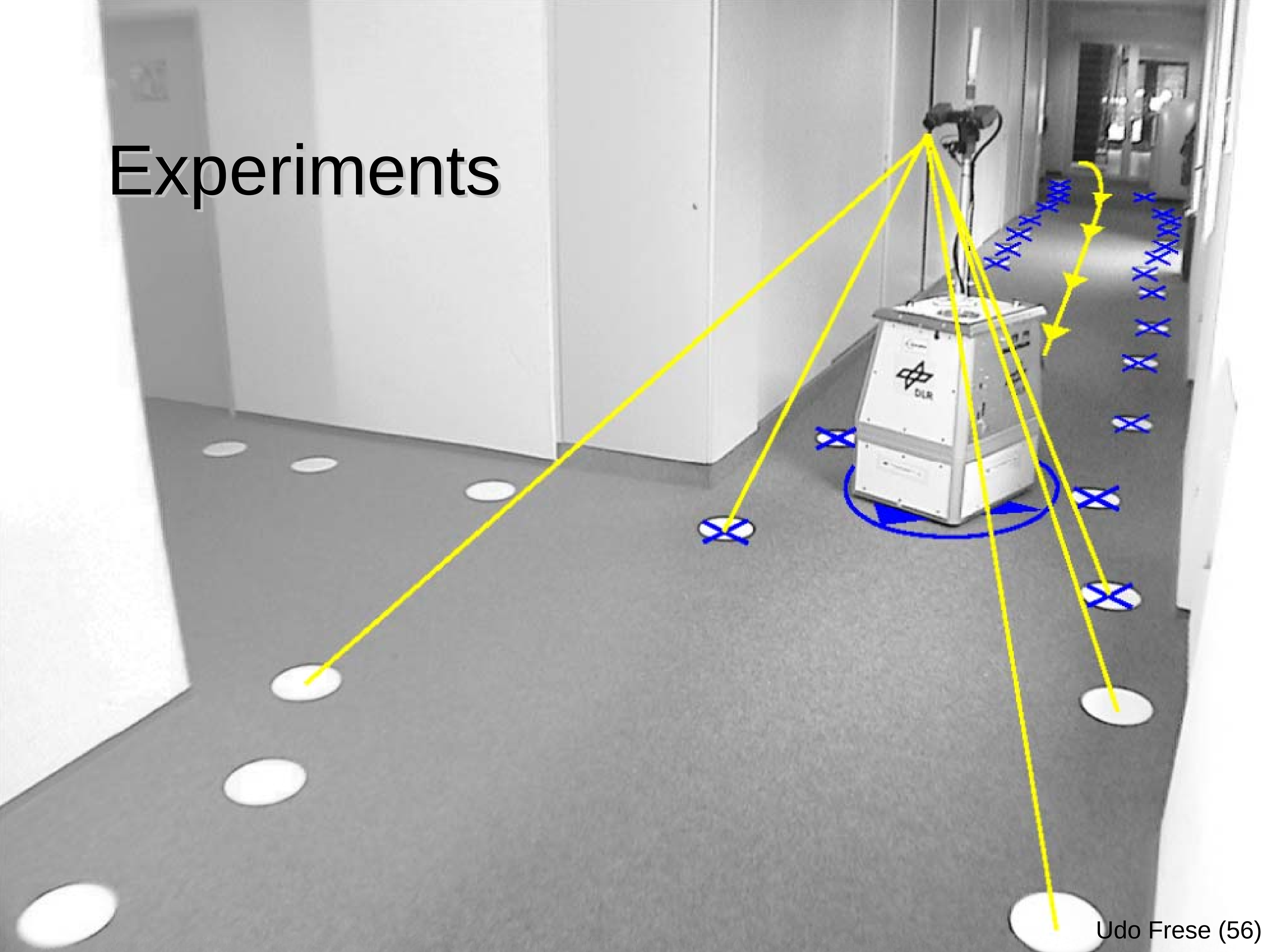


Treemap Algorithm

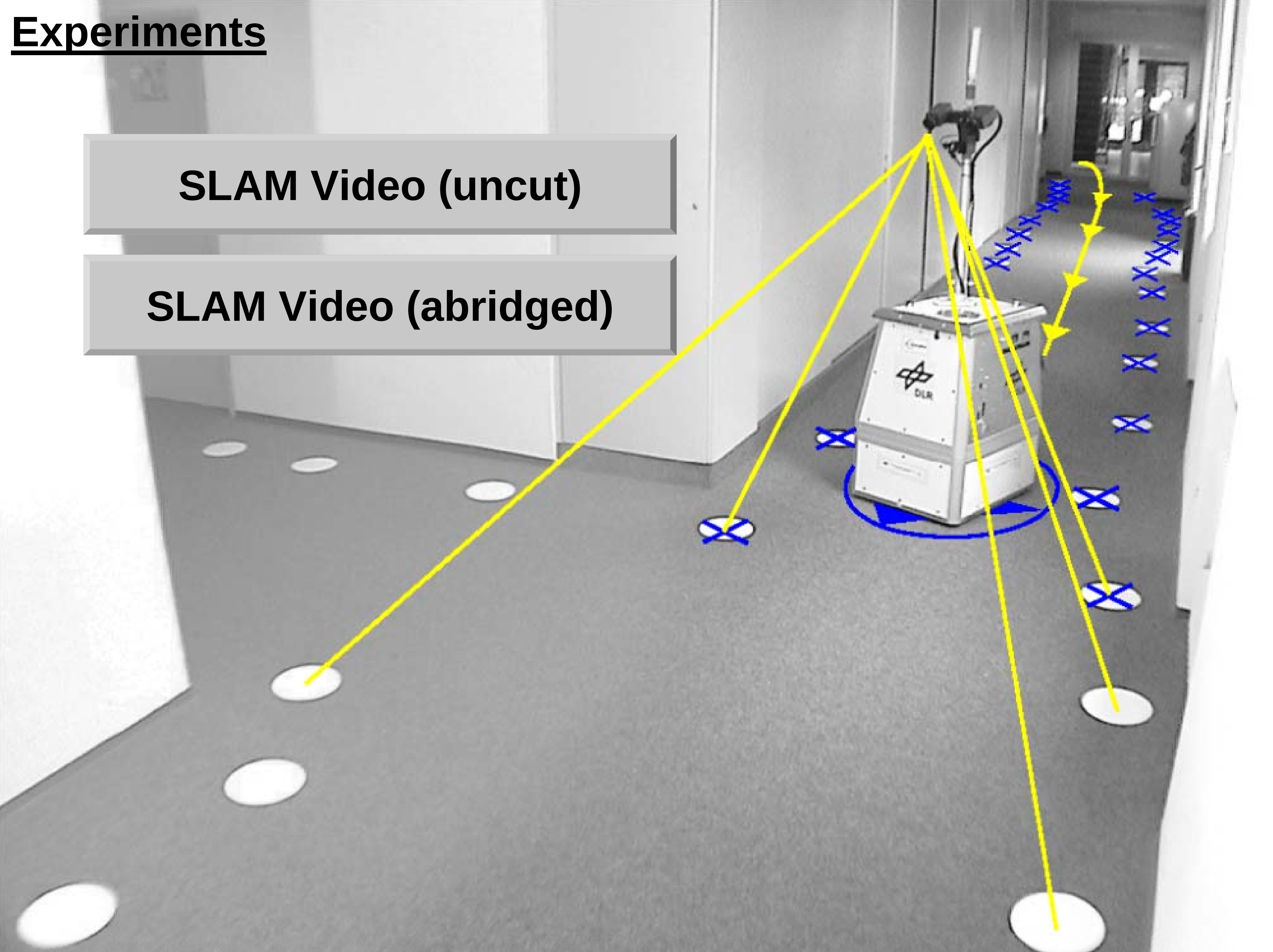
A „topologically suitable“ building



Experiments



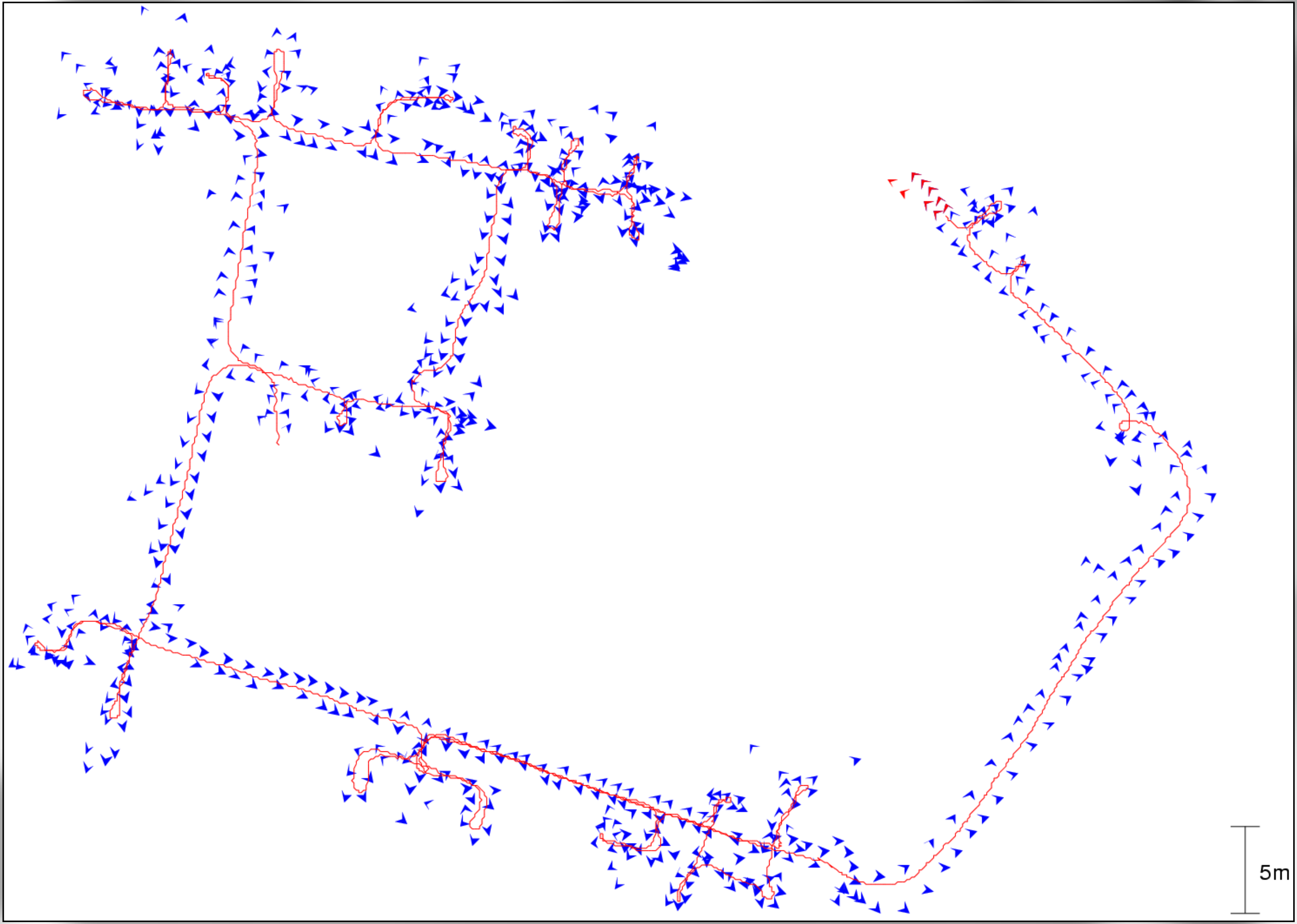
Experiments



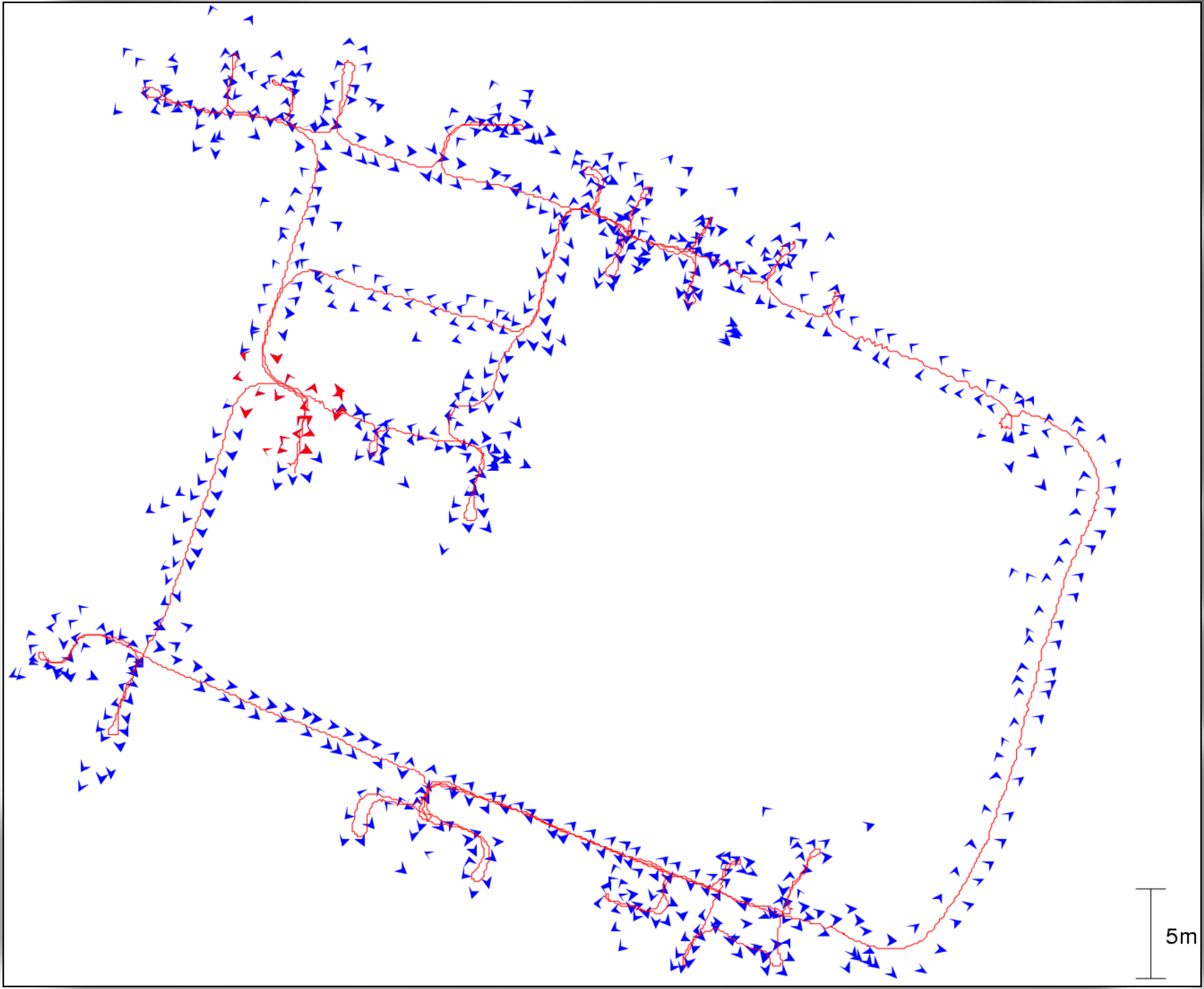
SLAM Video (uncut)

SLAM Video (abridged)

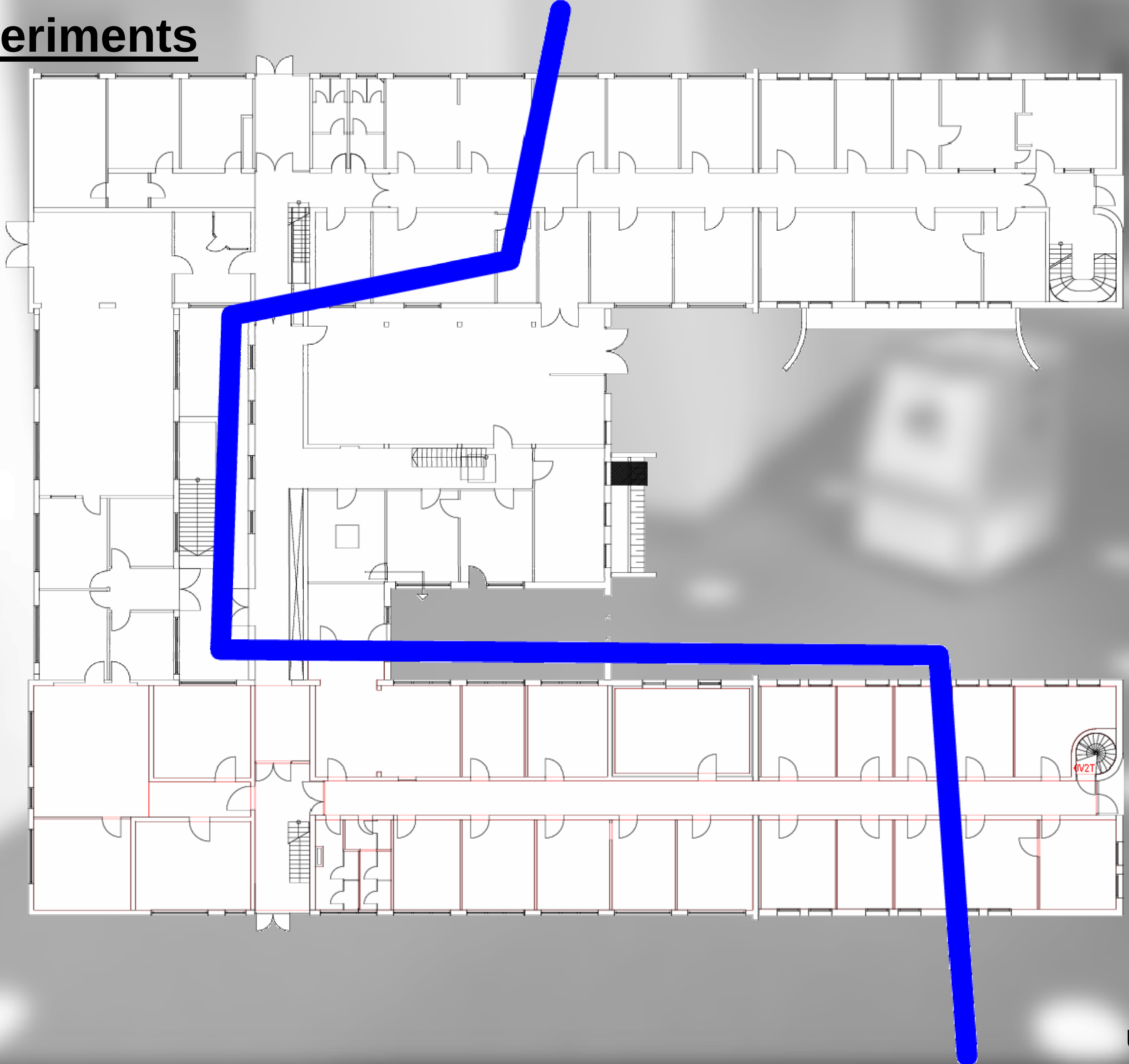
Experiments



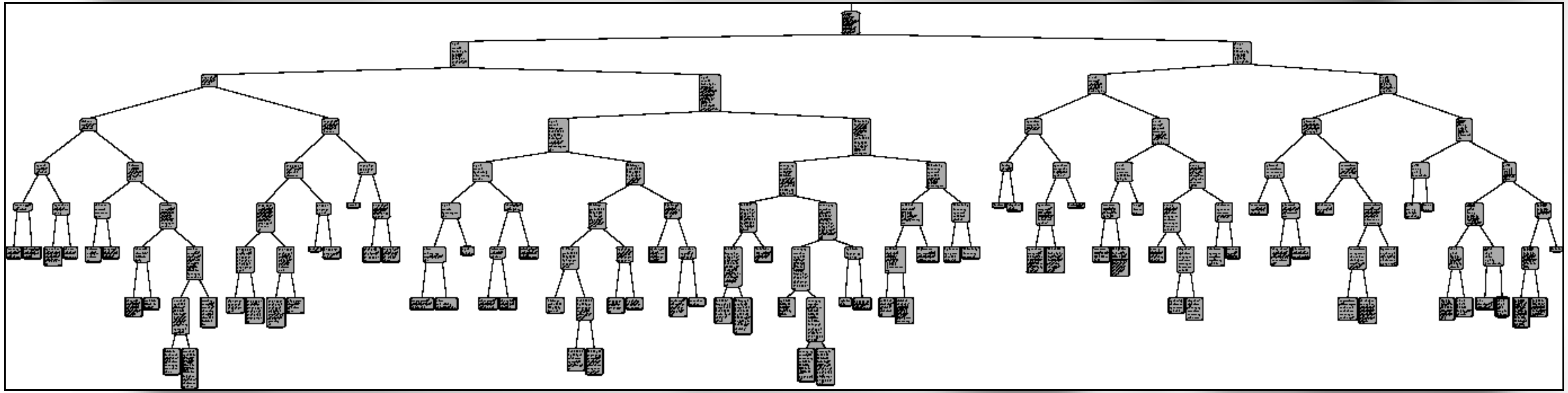
Experiments



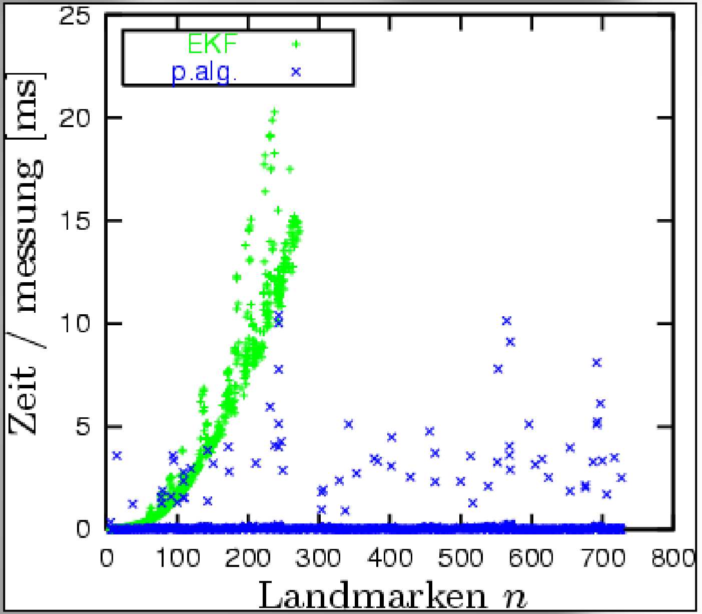
Experiments



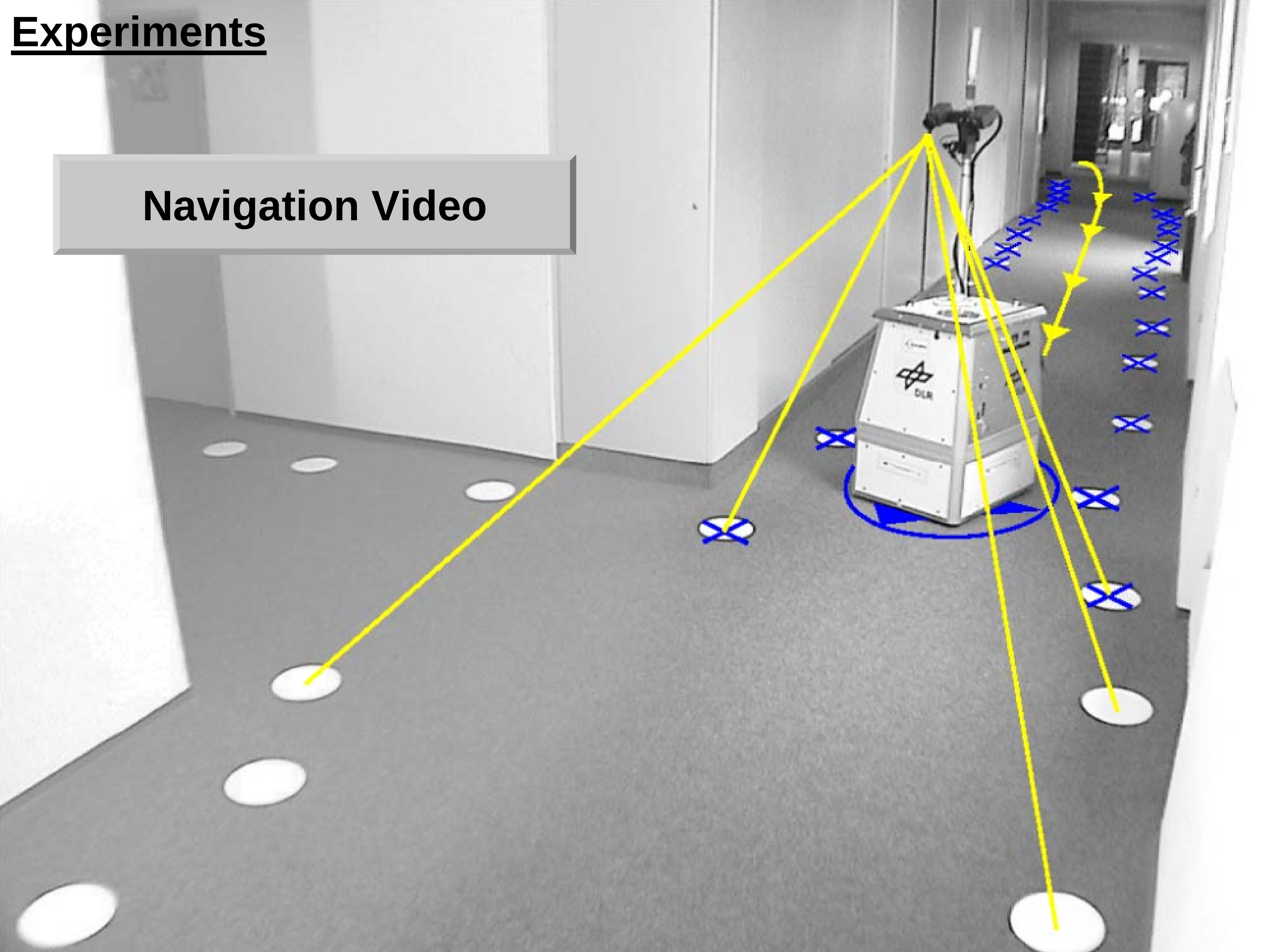
Experiments



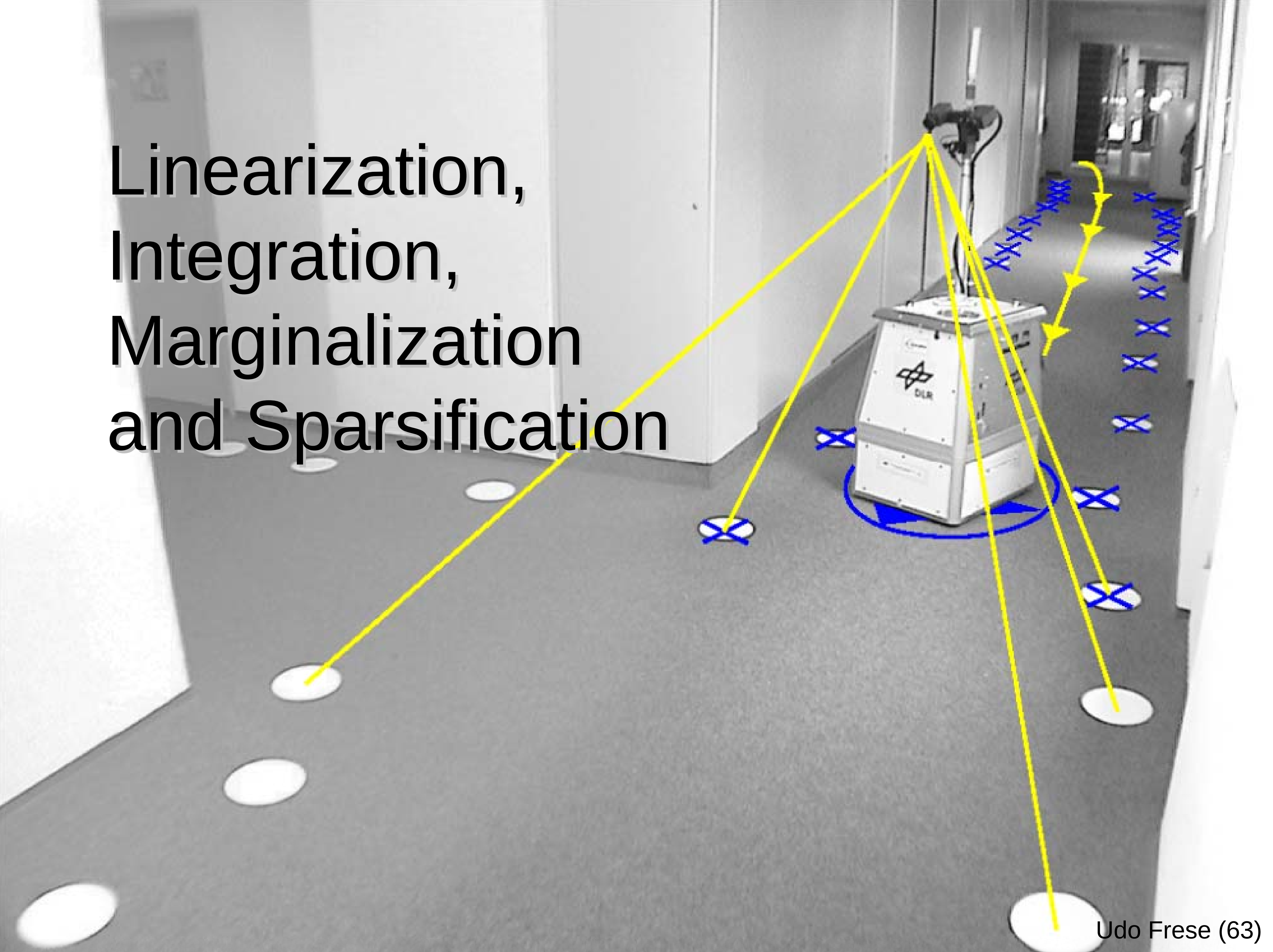
building	**60m × 45m**
rooms	**29**
distance traveled	**505m**
large loops	**3**
landmarks	**n = 725**
measurements	**m = 29142**
robot poses	**p = 3297**
local landmarks	**k ≅ 16**



Experiments



Navigation Video



Linearization, Integration, Marginalization and Sparsification

Different Levels of Approximation

- keep all non-linear measurements
  - recompute Jacobians every time you need.
- linearize
  - integrate a whole region into one matrix
- marginalize
  - marginalize out old poses inside a region
- sparsify
  - duplicate some old poses and marginalize out
  - cutting odometry (like ESDS-Filter)

Different Levels of Approximation

- keep all non-linear measurements
  - recompute Jacobians every time you need.
- linearize
  - integrate a whole region into one matrix
- marginalize
  - marginalize out old poses
- sparsify
  - duplicate some old poses and marginalize out
  - cutting odometry (like ESDS-Filter)

Only approximations in treemap.

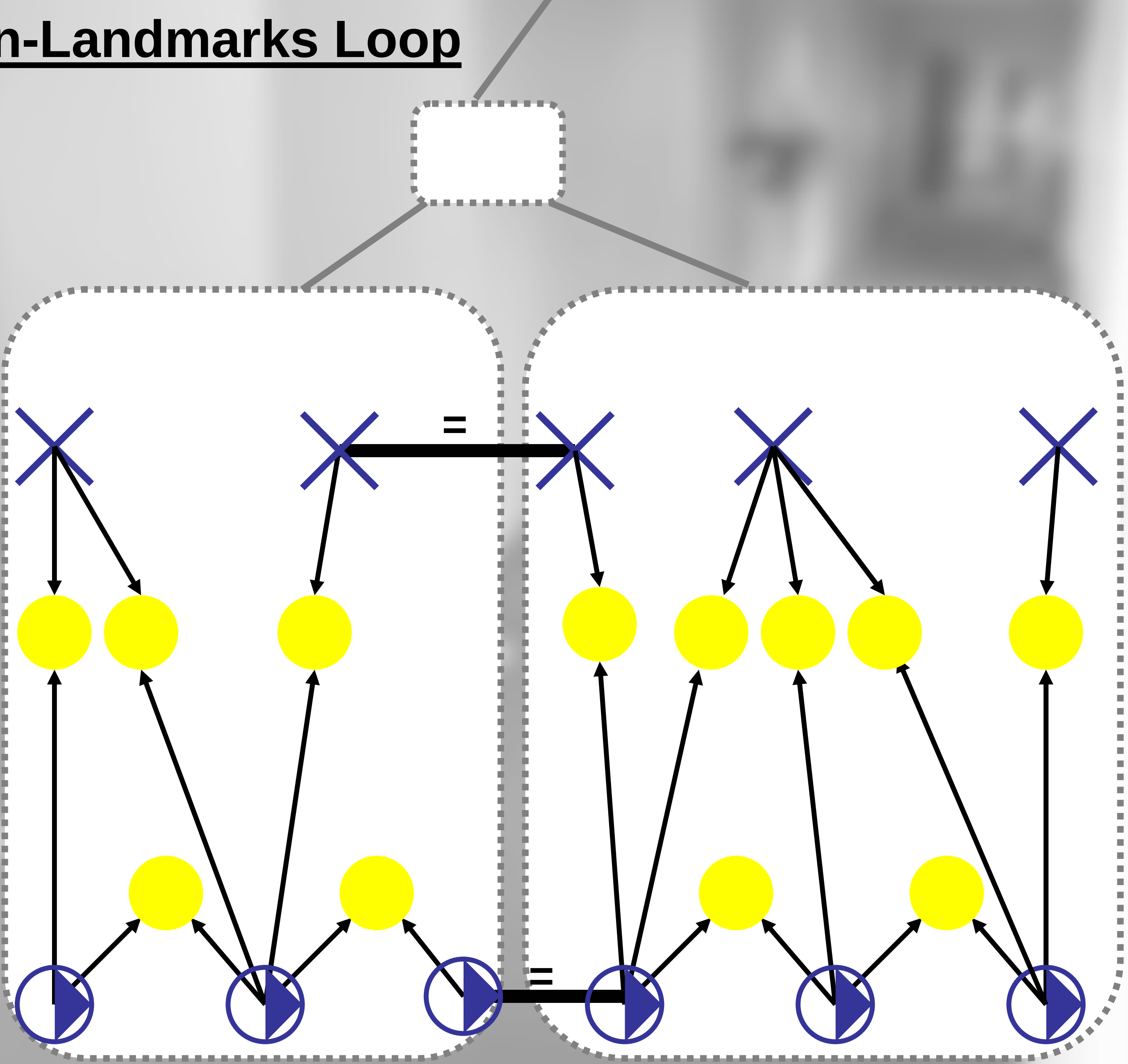
Closing a Million-Landmarks Loop

landmarks

landmark-
observations

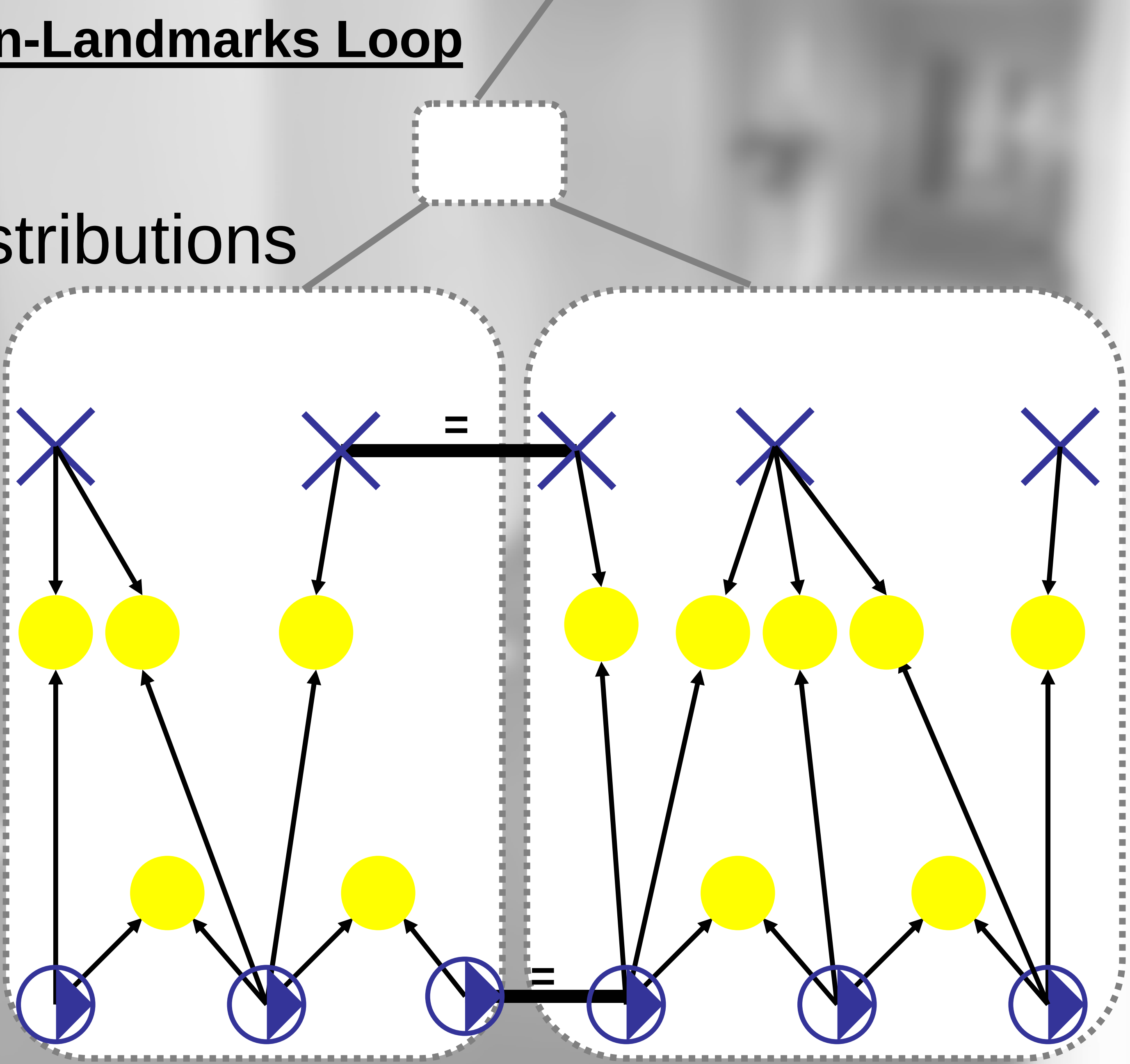
odometry

robot poses



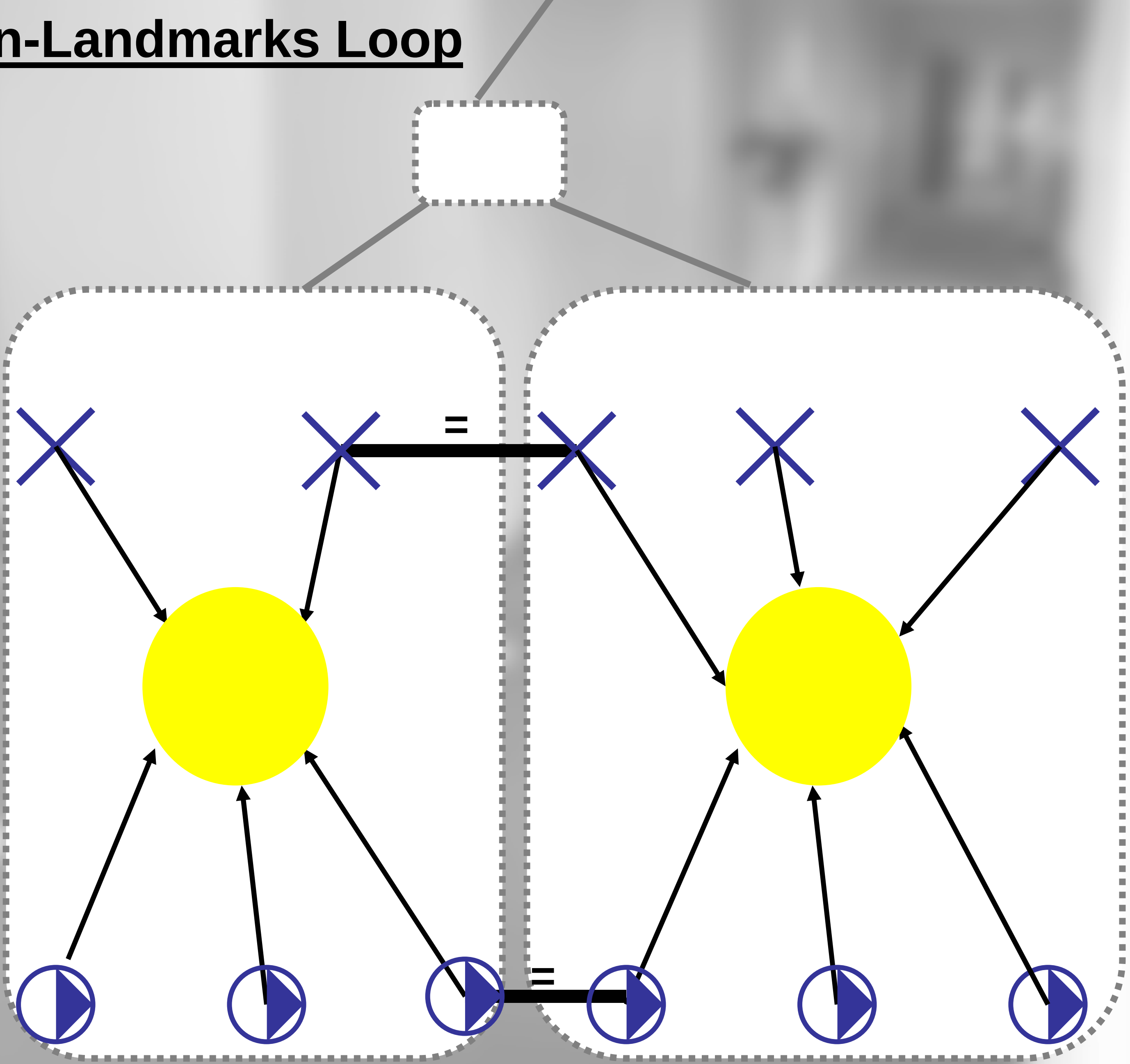
Closing a Million-Landmarks Loop

A:
Nonlinear distributions



Closing a Million-Landmarks Loop

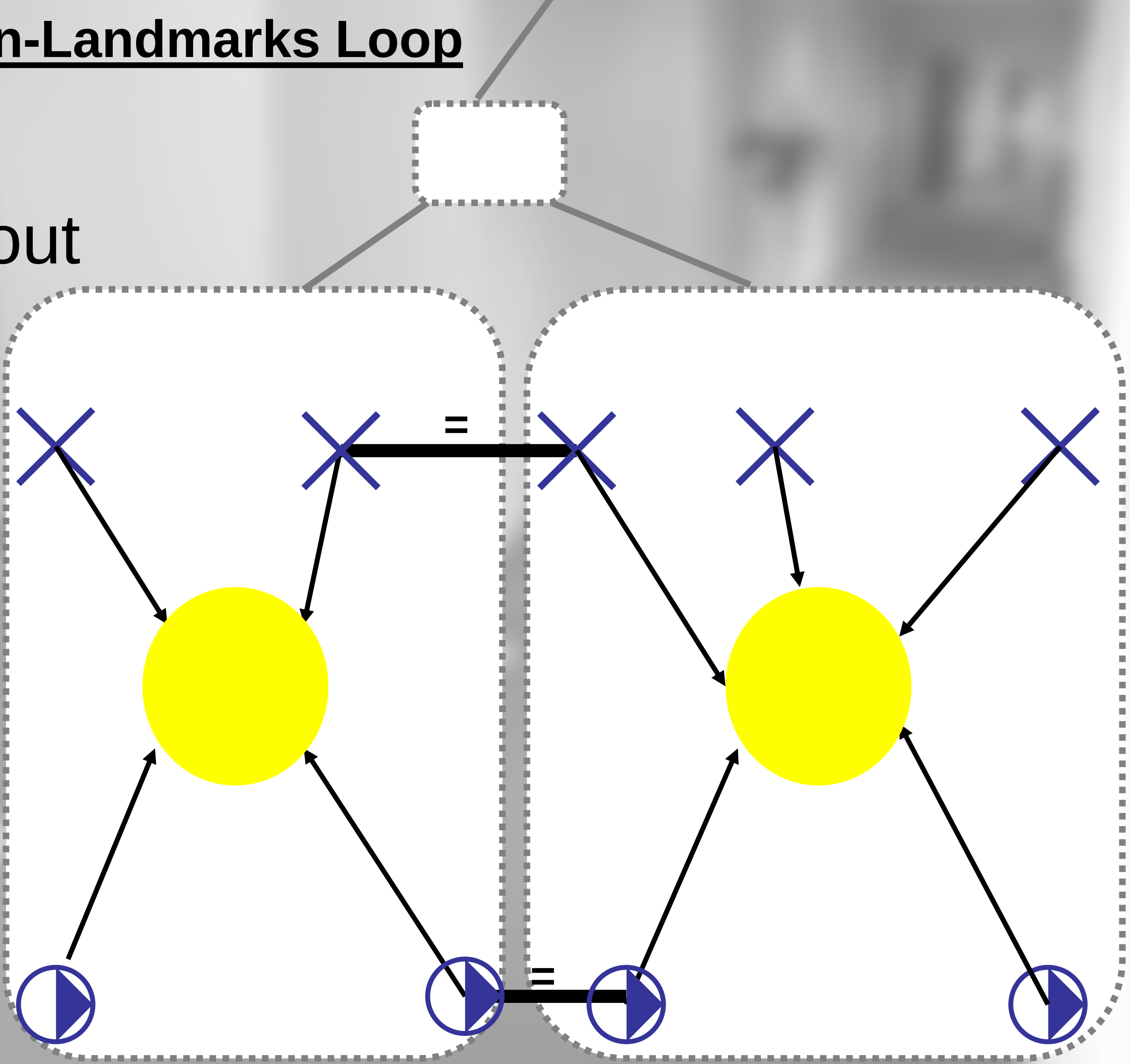
B:
Linearize



Closing a Million-Landmarks Loop

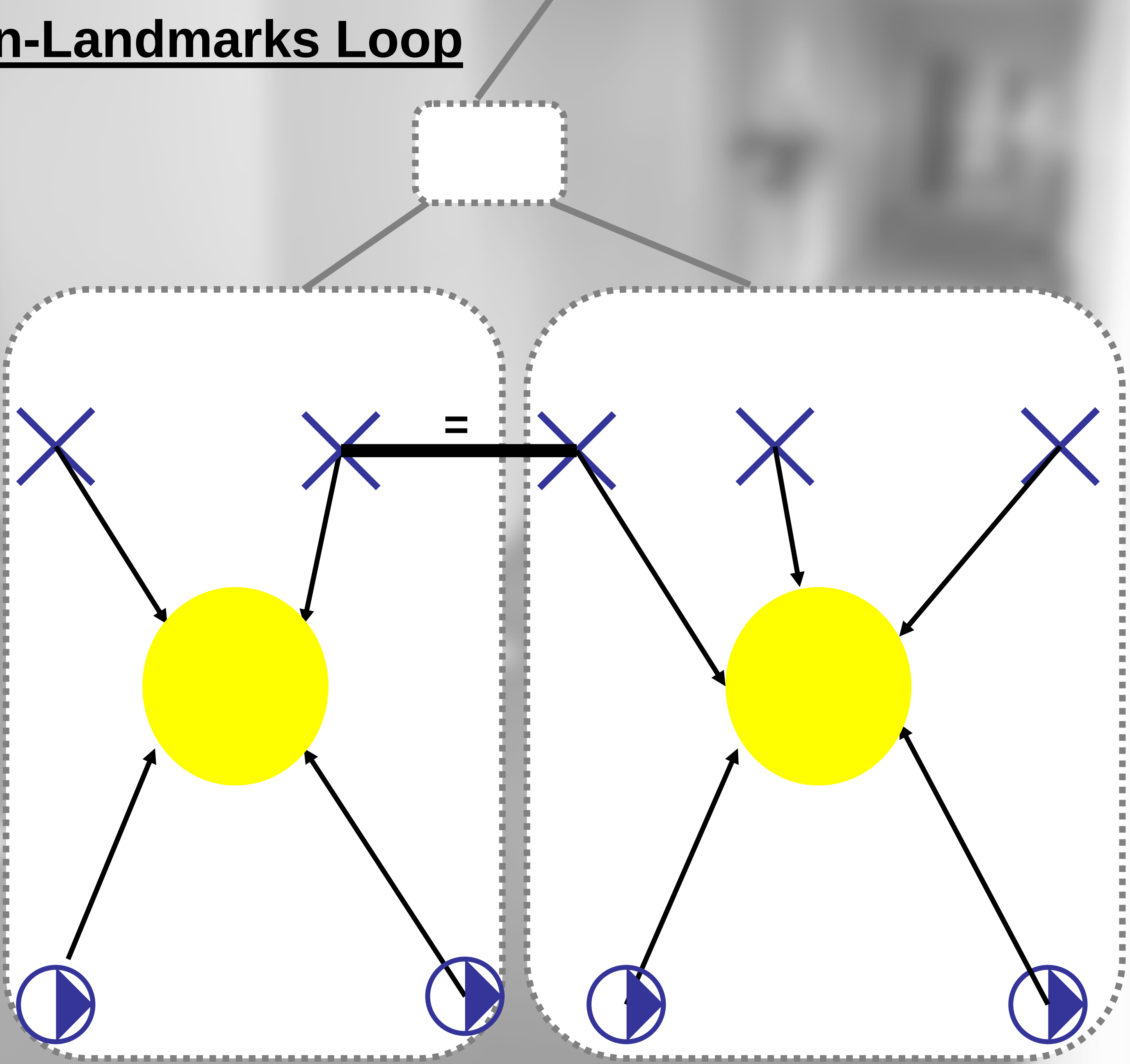
C:

Marginalize out inner poses



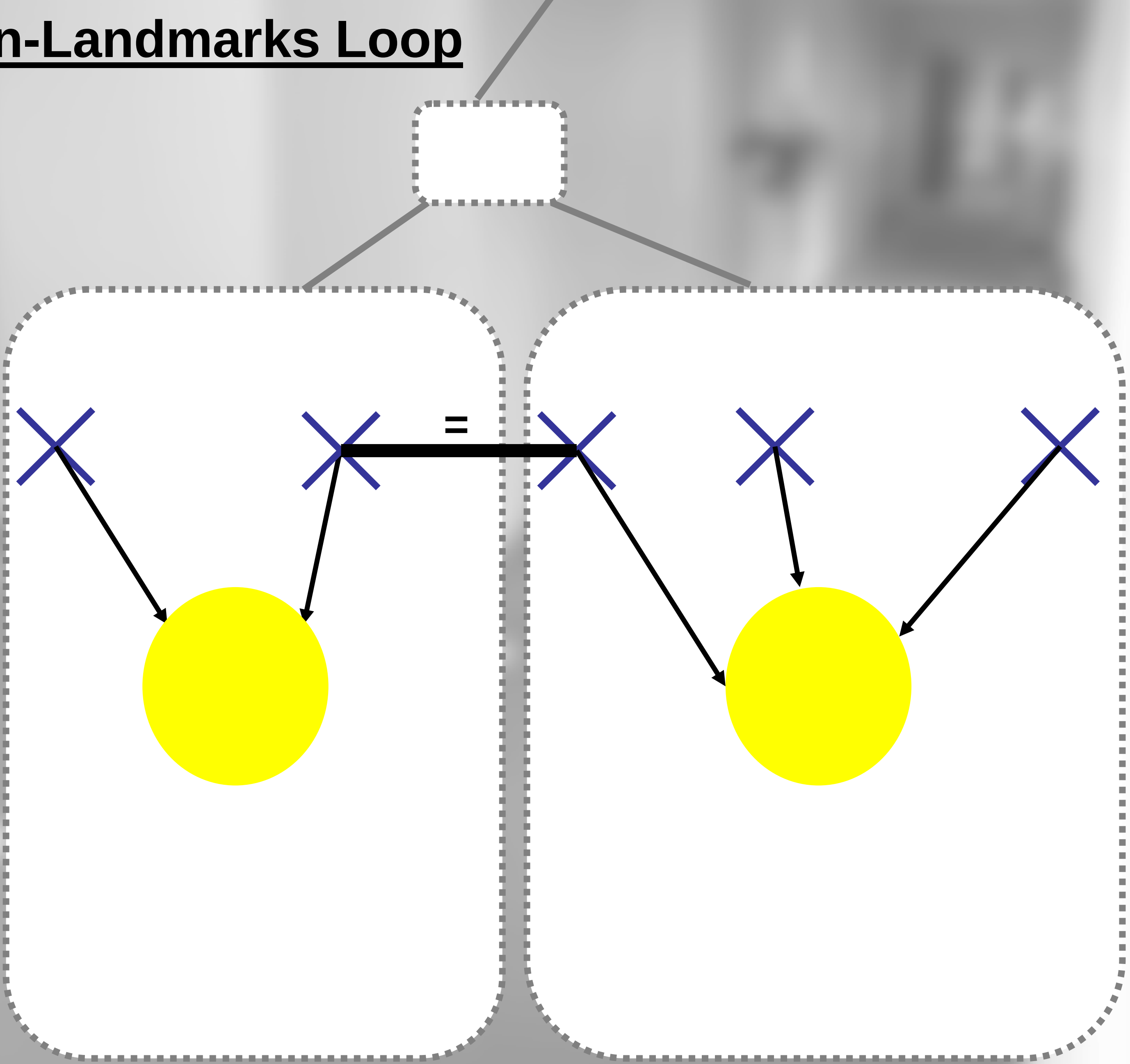
Closing a Million-Landmarks Loop

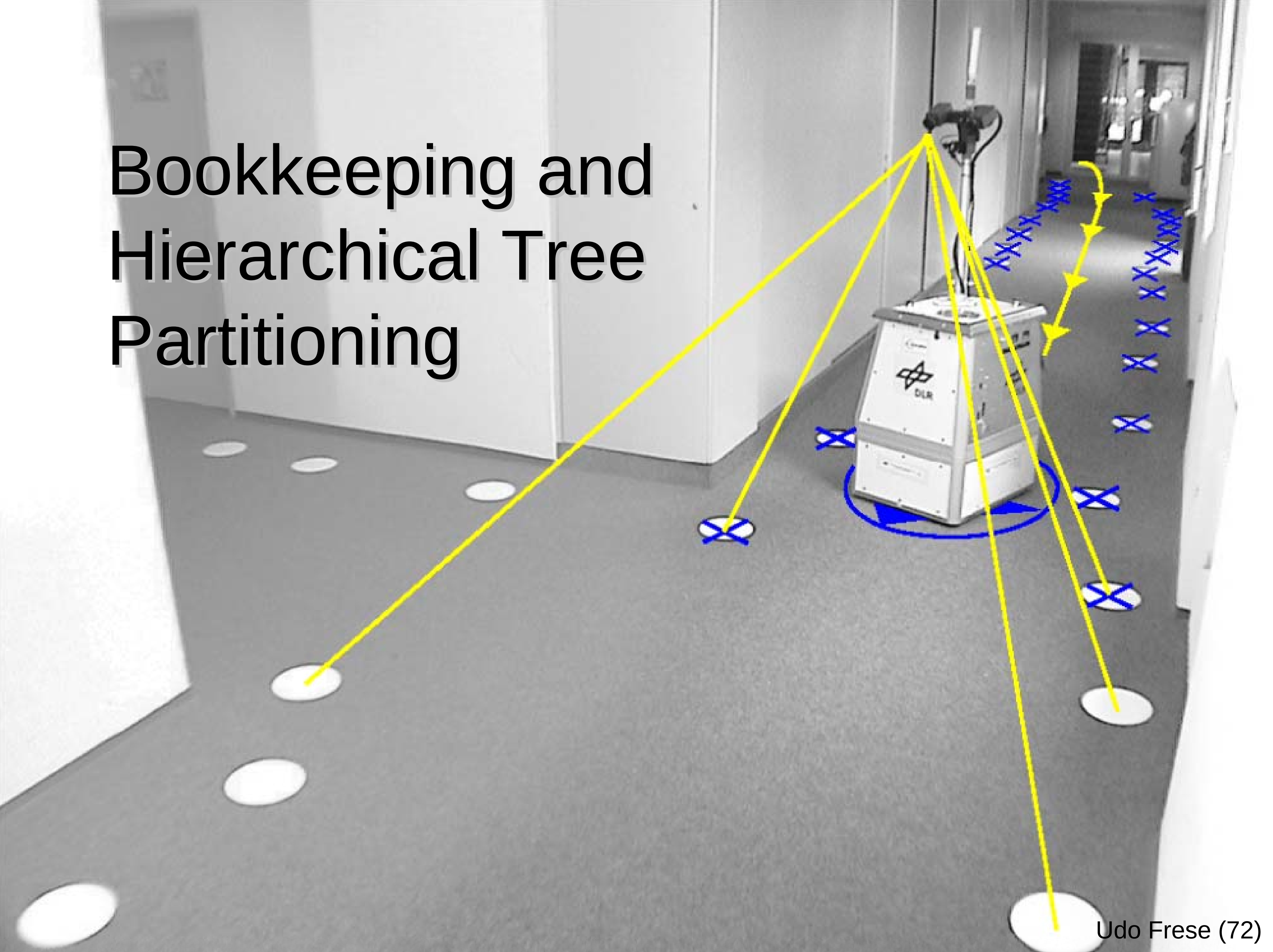
D:
Sparsify,
1: sacrifice
pose
equality
constraint



Closing a Million-Landmarks Loop

D:
Sparsify,
1: sacrifice pose equality constraint
2: marginalize out all poses





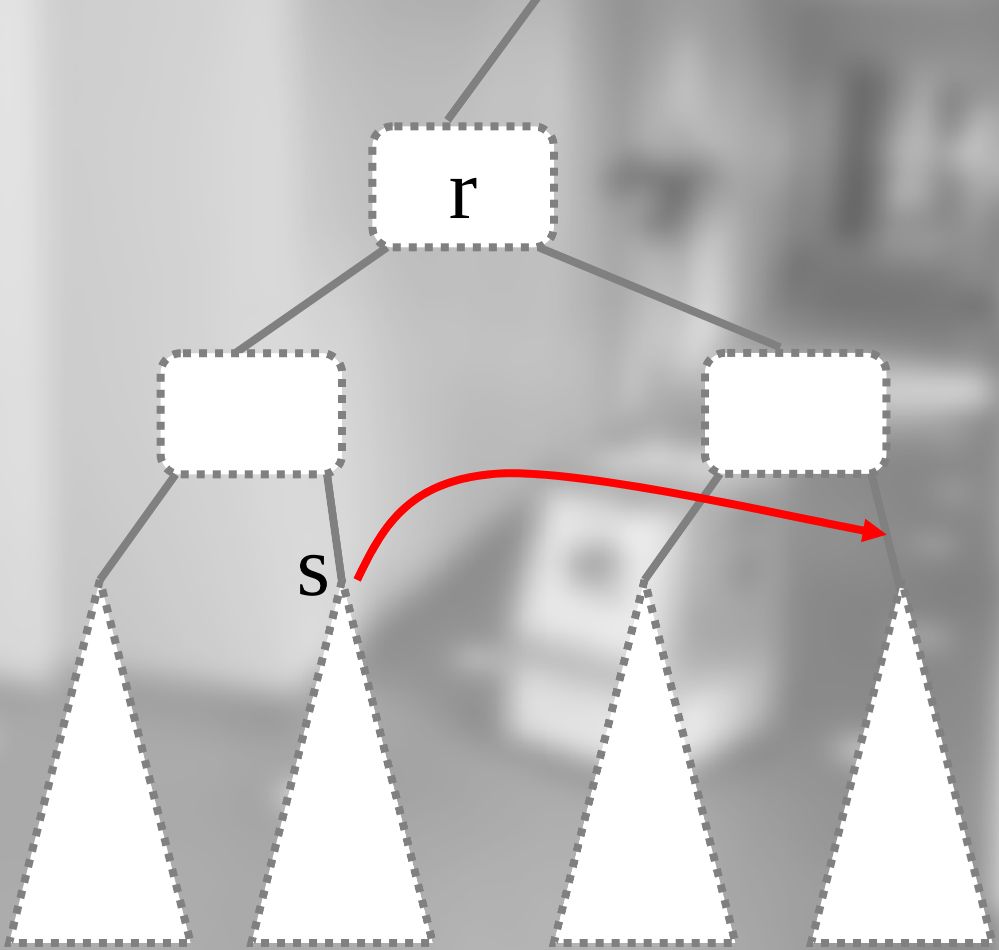
Bookkeeping and Hierarchical Tree Partitioning

Bookkeeping and HTP

- Which nodes to recompute?
- Rearrange the tree to improve computation time.
- NP-hard
- Multilevel Khernighan and Lin heuristics established in the field of graph partitioning
- Do some Khernighan and Lin runs after each update
- Optimize worst-case update time

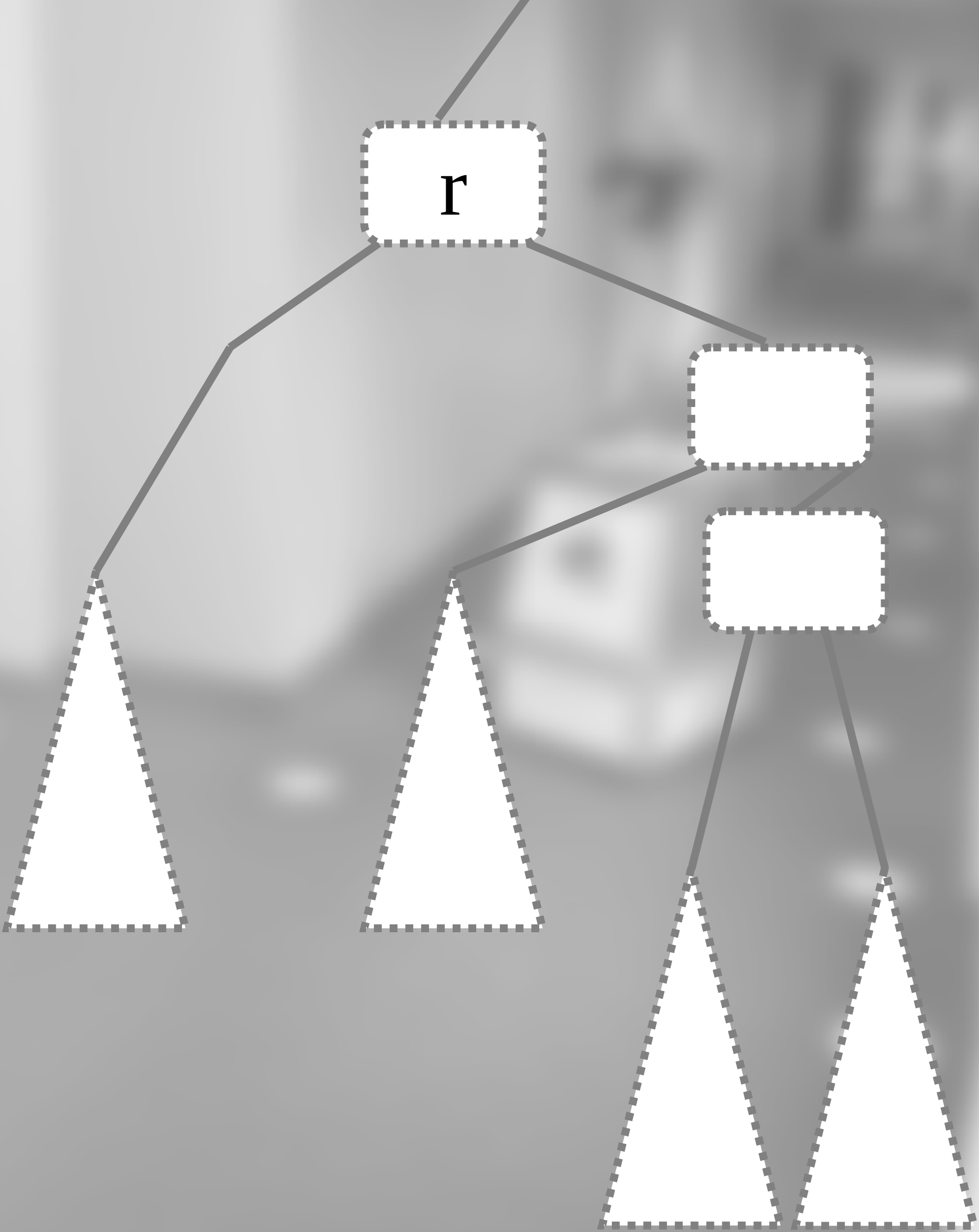
Bookkeeping and HTP

- Choose a node r from a queue
- Consider moving a single s subtree from one side of r to the other



Bookkeeping and HTP

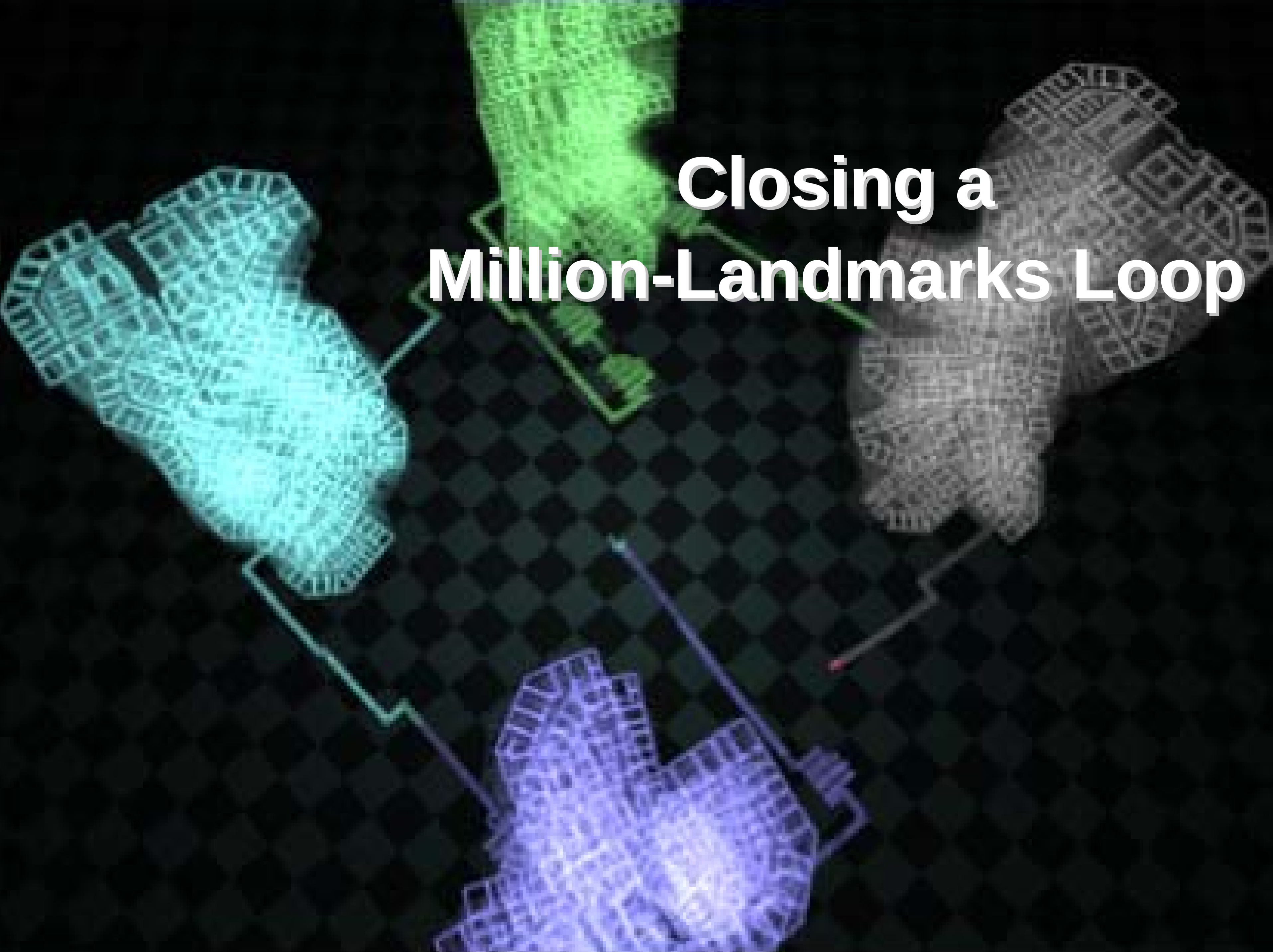
- Choose a node r from a queue
- Consider moving a single s subtree from one side of r to the other



Bookkeeping and HTP

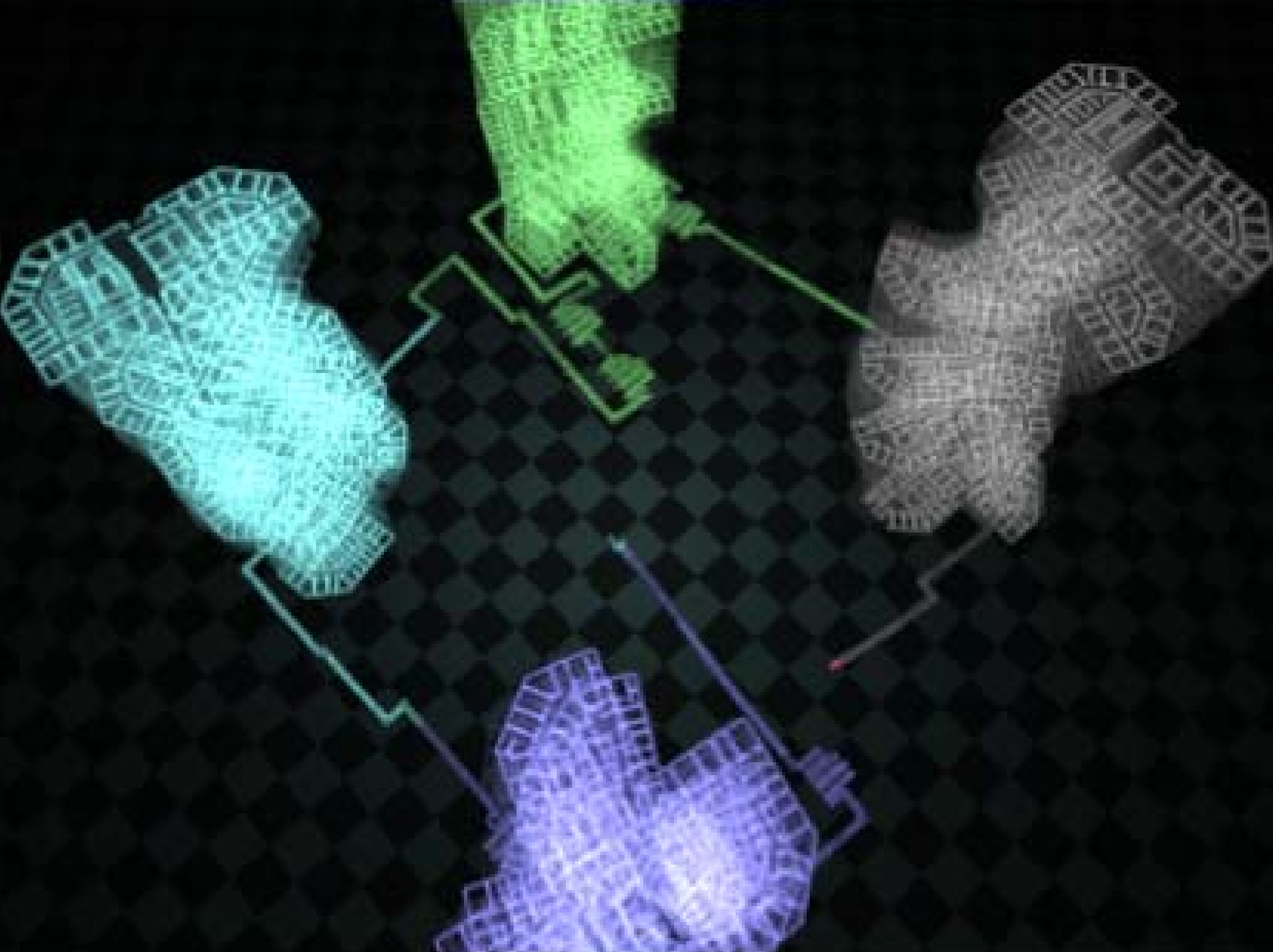
- Try to move every subtree that shares a feature (KL) on the left of s to move to the right of s and vice versa ($O(k \log^2 n)$)
- Choose the best
- Try it for some steps even if it makes things worst (KL)
- Consider integration, marginalization when moving
- Consider sparsification as a last resort

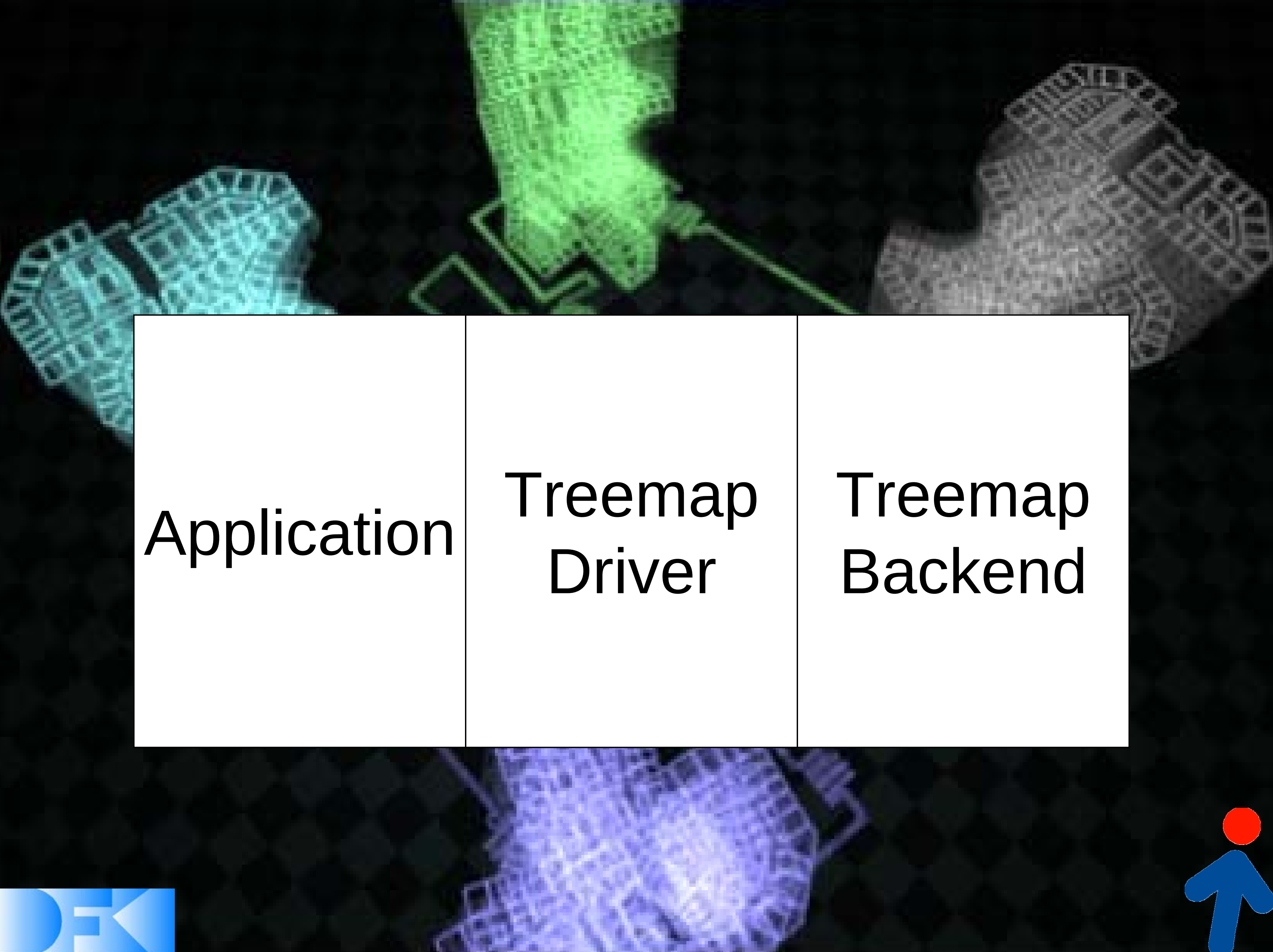
Closing a Million-Landmarks Loop

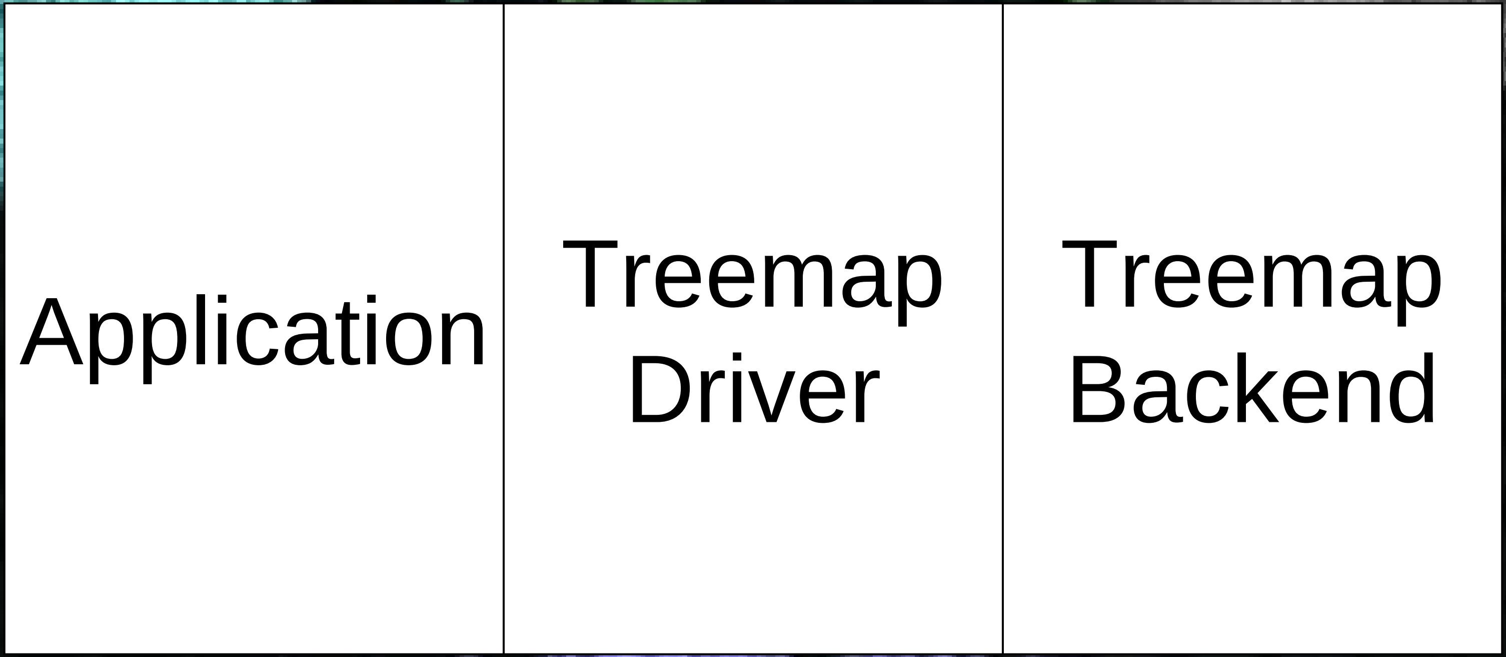


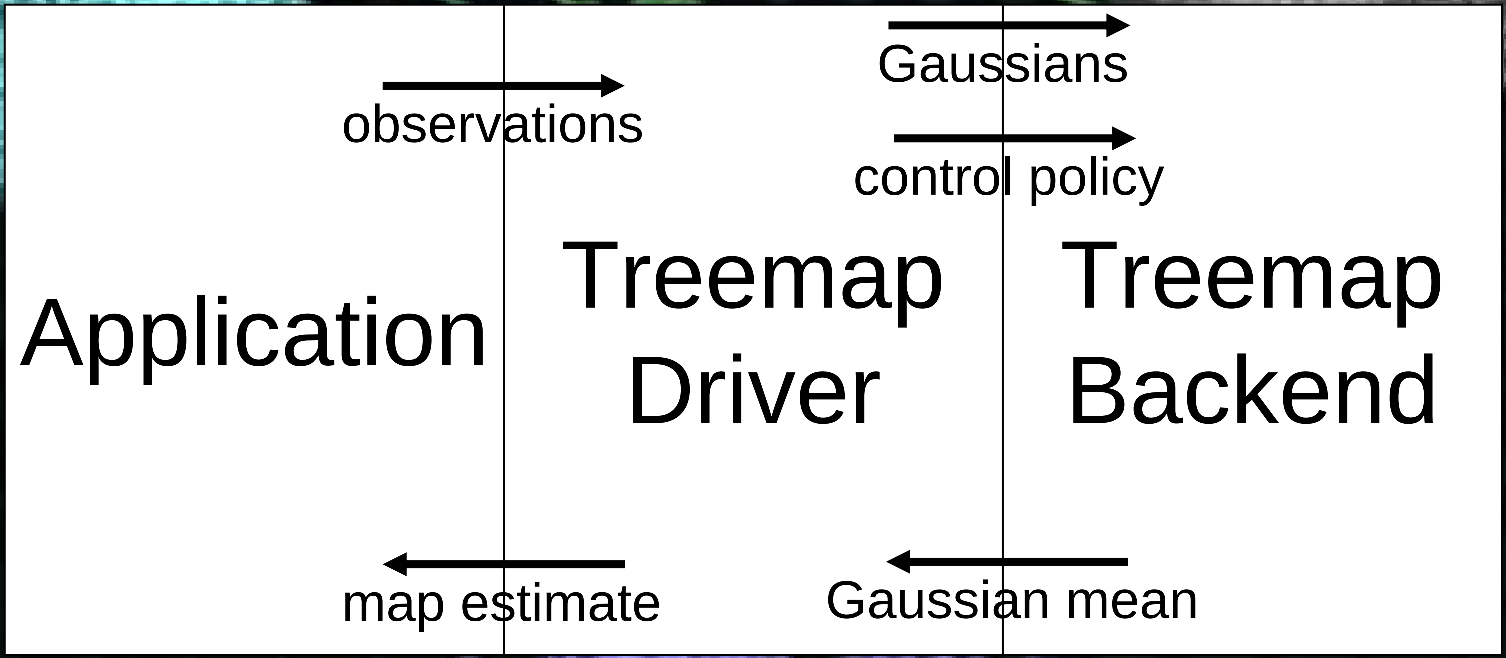
S. Julier and J. Uhlmann,
“Building a million beacon map”
Proceedings of SPIE: Sensor Fusion and Decentralized Control in Robotic Systems IV, vol. 4571, 2001.

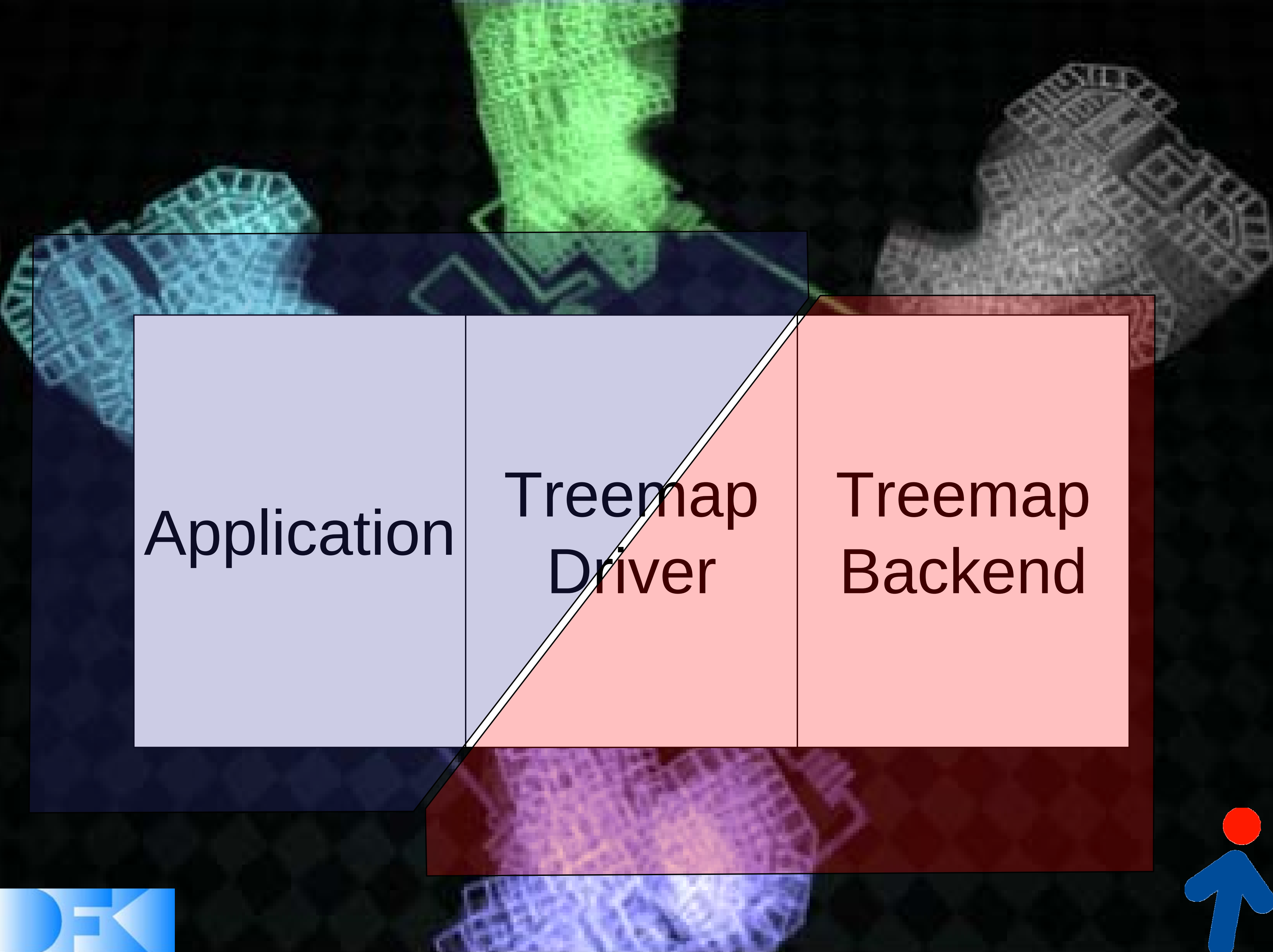
Our homage
our response






Application
Treemap Driver
Treemap Backend


observations
Gaussians
control policy
Application
Treemap Driver
Treemap Backend
map estimate
Gaussian mean


Application
Treemap Driver
Treemap Backend

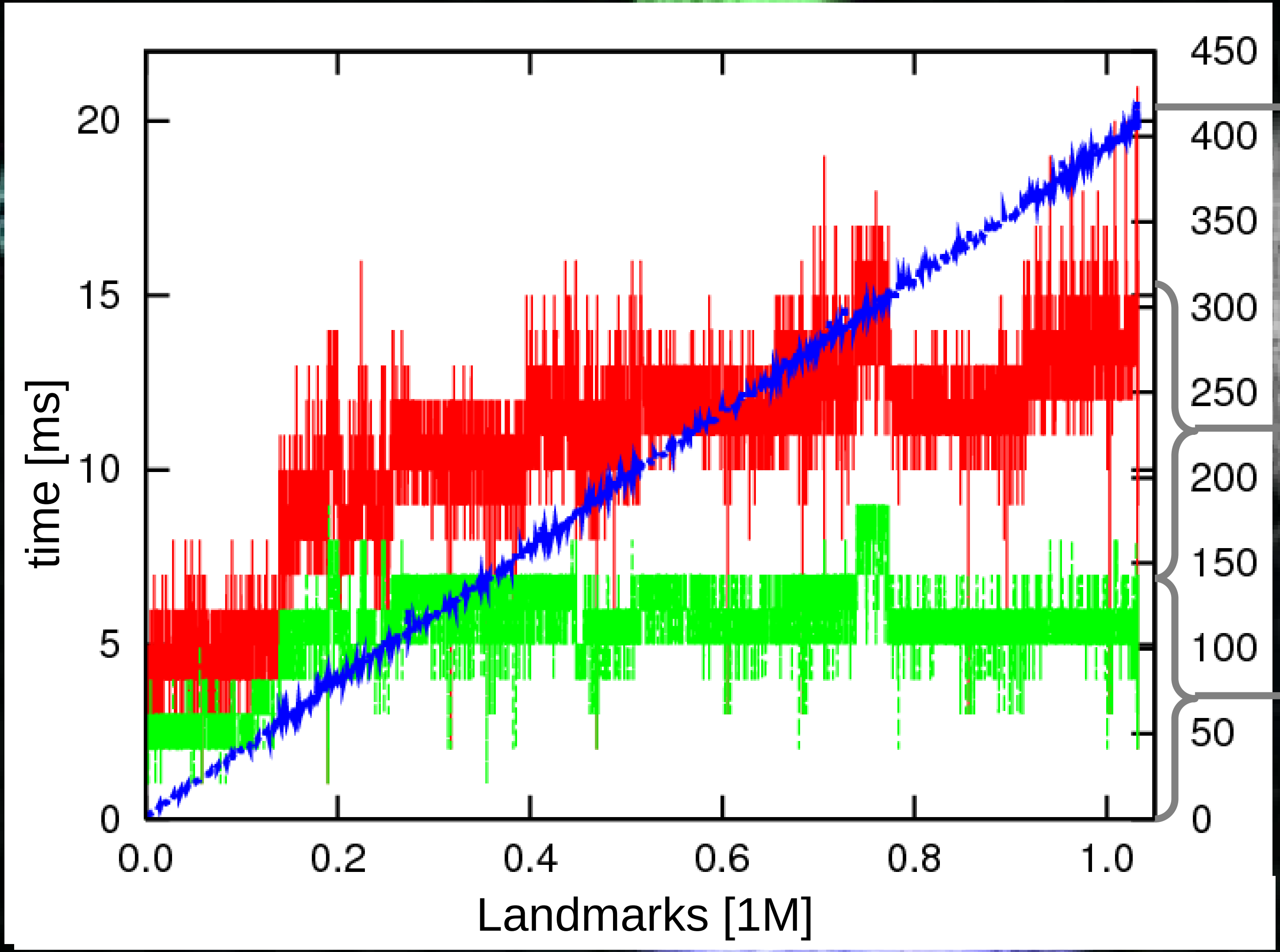
Video: Closing a Million-Landmarks Loop

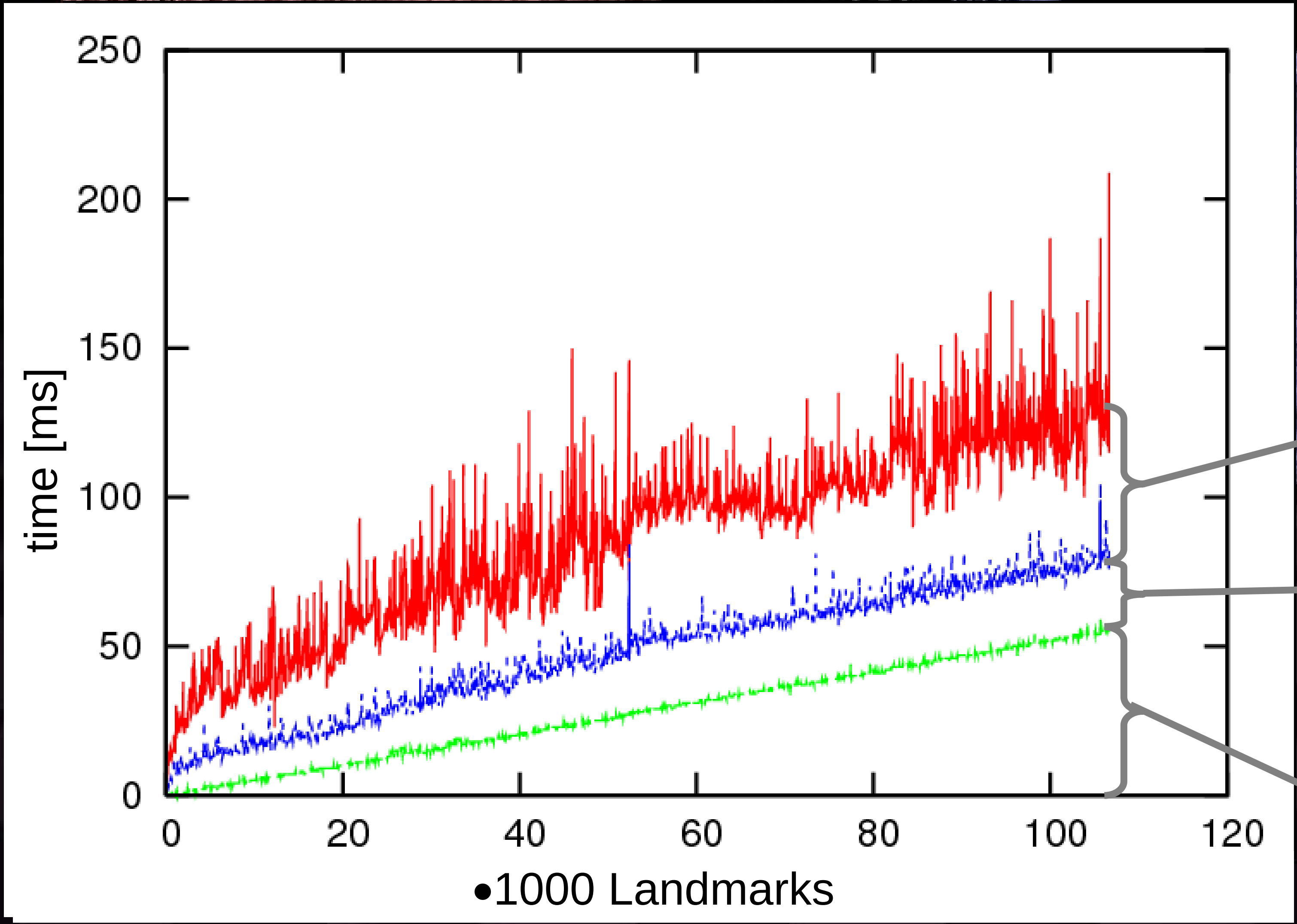
(http://www.informatik.uni-bremen.de/~ufrese/slammillionlandmarks/fresemillionlandmarks.avi)

Video: Using Treemap for a Generic Least Square Backend for 6-DOF SLAM

(http://www.informatik.uni-bremen.de/~ufrese/slammillionlandmarks/avi)

The Experiments


time [ms]
0
5
10
15
20
0.0
0.2
0.4
0.6
0.8
1.0
Landmarks [1M]
0
50
100
150
200
250
300
350
400
450
global downward estimation
upward update
book-keeping



upward update

book-keeping

downward estimation

Treemap

- closes a loop over 1032271 features in 21ms (local) or 442ms (global)
- $O(k^3 \log n + k^2 \log^2 n + kn)$
- generic backend & specific driver
- open source soon
- driver has to implement
  - measurement function, initial estimate, Jacobian
  - approximation policy
  - 2-D, 3-DOF: 690 lines of C++ code
  - 3-D, 6-DOF: 410 lines of C++ code