Robotic 3D Vision

Lecture 4: Probabilistic State Estimation – Full Posterior Optimization

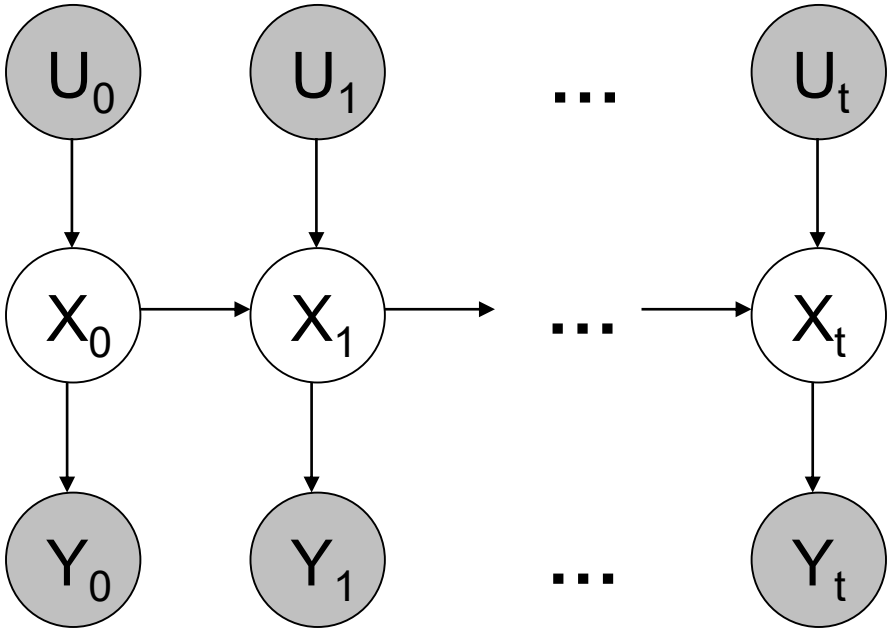
Prof. Dr. Jörg Stückler
Computer Vision Group, TU Munich
http://vision.in.tum.de

What We Will Cover Today

- Probabilistic modelling of state estimation problems
  - Examples of observation and state-transition models
  - Example: Monte-Carlo Localization
- Short intro to graphical models
  - Directed graphical models and factor graphs
- Full posterior optimization
  - Non-linear least squares
  - Optimization methods
  - Tools and frameworks

Recap: Probabilistic Model of Time-Sequential Processes

- Hidden state X gives rise to noisy observations Y
- At each time t,
  - the state changes stochastically from X_{t-1} to X_t
  - state change depends on action U_t
  - we get a new observation Y_t



Recap: Why Probabilistic State Estimation?

- Probabilistic modelling accounts for uncertainties
- State estimation: Inference in probabilistic model
- Cope with noisy state transitions and observations
- Maintain uncertainty in the state estimate
- Principled approaches to update the state estimate distribution based on probability theory

Recap: Observation and State-Transition Models

- We assume
  - Knowledge about probability distribution of observations

$$p\left(Y_t \middle| X_{0:t}, U_{0:t}, Y_{0:t-1}\right)$$

  - Knowledge about probabilistic dynamics of state transitions

$$p\left(X_t \middle| X_{0:t-1}, U_{0:t}\right)$$

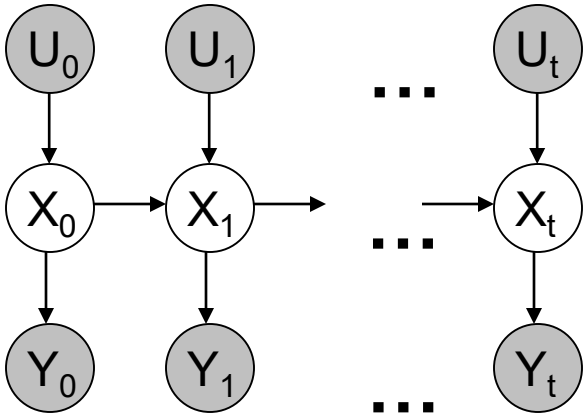
Recap: Markov Assumptions

- Only the immediate past matters for a state transition

$$p\left(X_t \middle| X_{0:t-1}, U_{0:t}\right) = \boxed{p\left(X_t \middle| X_{t-1}, U_t\right)}$$ **state transition model**

- Observations depend only on the current state

$$p\left(Y_t \middle| X_{0:t}, U_{0:t}, Y_{0:t-1}\right) = \boxed{p\left(Y_t \middle| X_t\right)}$$ **observation model**

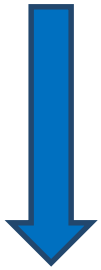


Recap: Predict-Correct Cycle

- Prediction:

$$p(X_t \mid y_{0:t-1}, u_{0:t}) = \int p(X_t \mid X_{t-1}, u_t)\, p(X_{t-1} \mid y_{0:t-1}, u_{0:t-1})\, dX_{t-1}$$

observation y_t



action u_t

- Correction:

$$p(X_t \mid y_0, \ldots, y_t) = \frac{p(y_t \mid X_t)\, p(X_t \mid y_{0:t-1}, u_{0:t})}{\int p(y_t \mid X_t)\, p(X_t \mid y_{0:t-1}, u_{0:t})\, dX_t}$$

Recap: Kalman Filter

- Kalman filters (KFs) instantiate recursive Bayesian filtering for a specific class of state transition and observation models

  - Linear state transition model with Gaussian noise:

$$\mathbf{x}_t = \mathbf{A}_t \mathbf{x}_{t-1} + \mathbf{B}_t \mathbf{u}_t + \boldsymbol{\epsilon} \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{d_t})$$

  - Linear observation model with Gaussian noise:

$$\mathbf{z}_t = \mathbf{C}_t \mathbf{x}_t + \boldsymbol{\delta} \qquad \boldsymbol{\delta} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{m_t})$$

  - Gaussian initial state estimate: $\mathbf{x}_0 \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$

Recap: Kalman Filter Prediction & Correction

- Efficient closed-form correction and prediction steps which involve manipulation of Gaussians
- The state estimate can be represented as a Gaussian distribution

$$\mathbf{x}_t \sim \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$$

- Prediction:

$$\boldsymbol{\mu}_t^- = \mathbf{A}_t \boldsymbol{\mu}_{t-1}^+ + \mathbf{B}_t \mathbf{u}_t$$
$$\boldsymbol{\Sigma}_t^- = \mathbf{A}_t \boldsymbol{\Sigma}_{t-1}^+ \mathbf{A}_t^\top + \boldsymbol{\Sigma}_{d_t}$$

- Correction:

$$\mathbf{K}_t = \boldsymbol{\Sigma}_t^- \mathbf{C}_t^\top \left(\mathbf{C}_t \boldsymbol{\Sigma}_t^- \mathbf{C}_t^\top + \boldsymbol{\Sigma}_{m_t} \right)^{-1}$$
$$\boldsymbol{\mu}_t^+ = \boldsymbol{\mu}_t^- + \mathbf{K}_t (\mathbf{y}_t - \mathbf{C}_t \boldsymbol{\mu}_t^-)$$
$$\boldsymbol{\Sigma}_t^+ = (\mathbf{I} - \mathbf{K}_t \mathbf{C}_t) \boldsymbol{\Sigma}_t^-$$

Kalman gain

Recap: Extended Kalman Filter (EKF)

- Non-linear state-transition model with Gaussian noise:

$$\mathbf{x}_t = g(\mathbf{x}_{t-1}, \mathbf{u}_t) + \boldsymbol{\epsilon}_t \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{d_t})$$

- Non-linear observation model with Gaussian noise:

$$\mathbf{y}_t = h(\mathbf{x}_t) + \boldsymbol{\delta}_t \qquad \boldsymbol{\delta} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{m_t})$$

- How to cope with non-linear system?

- Idea: linearize the models in each time step

$$\Rightarrow \quad \mathbf{x}_t \approx g(\mathbf{x}^0_{t-1}, \mathbf{u}_t) + \nabla g(\mathbf{x}, \mathbf{u}_t)|_{\mathbf{x}=\mathbf{x}^0_{t-1}} \left(\mathbf{x}_{t-1} - \mathbf{x}^0_{t-1}\right) + \boldsymbol{\epsilon}_t$$

$$\Rightarrow \quad \mathbf{y}_t \approx h(\mathbf{x}^0_t) + \nabla h(\mathbf{x})|_{\mathbf{x}=\mathbf{x}^0_t} \left(\mathbf{x}_t - \mathbf{x}^0_t\right) + \boldsymbol{\delta}_t$$

Recap: EKF Prediction & Correction

- Efficient approximate correction and prediction steps which involve manipulation of Gaussians and linearization
- The state estimate can be represented as a Gaussian distribution

$$\mathbf{x}_t \sim \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$$

- Prediction:

$$\boldsymbol{\mu}_t^- = g(\boldsymbol{\mu}_{t-1}^+, \mathbf{u}_t)$$

$$\boldsymbol{\Sigma}_t^- = \mathbf{G}_t \boldsymbol{\Sigma}_{t-1}^+ \mathbf{G}_t^\top + \boldsymbol{\Sigma}_{d_t} \qquad \mathbf{G}_t := \nabla g(\mathbf{x}, \mathbf{u}_t)|_{\mathbf{x}=\boldsymbol{\mu}_{t-1}^+}$$

- Correction:

$$\mathbf{K}_t = \boldsymbol{\Sigma}_t^- \mathbf{H}_t^\top \left(\mathbf{H}_t \boldsymbol{\Sigma}_t^- \mathbf{H}_t^\top + \boldsymbol{\Sigma}_{m_t}\right)^{-1}$$

$$\boldsymbol{\mu}_t^+ = \boldsymbol{\mu}_t^- + \mathbf{K}_t \left(\mathbf{y}_t - h(\boldsymbol{\mu}_t^-)\right) \qquad \mathbf{H}_t := \nabla h(\mathbf{x})|_{\mathbf{x}=\boldsymbol{\mu}_t^-}$$

$$\boldsymbol{\Sigma}_t^+ = \left(\mathbf{I} - \mathbf{K}_t \mathbf{H}_t\right) \boldsymbol{\Sigma}_t^-$$

Recap: Particle Filter (PF)

- Non-linear observation and state-transition distributions

$$p(\mathbf{y}_t \mid \mathbf{x}_t) \qquad p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t)$$

- State estimate (full posterior!) represented as a set of weighted samples

$$\left\{\mathbf{x}_{0:t}^i, w_t^i\right\}_{i=1}^N$$

$$p(\mathbf{x}_{0:t} \mid \mathbf{y}_{0:t}, \mathbf{u}_{1:t}) \approx \sum_{i=1}^N w_t^i \delta_{\mathbf{x}_{0:t}^i}(\mathbf{x}_{0:t})$$

- The weighted samples a.k.a. particles are propagated and updated over time to approximate the full posterior

Recap: Sampling Importance Resampling (SIR)

- Sequential update:
  - Particle update: $\mathbf{x}_t^i \sim q(\mathbf{x}_t \mid \mathbf{x}_{t-1}^i, \mathbf{y}_t, \mathbf{u}_t)$
  - Weight update: $w_t^i = w_{t-1}^i \frac{p(\mathbf{y}_t|\mathbf{x}_t^i)\, p(\mathbf{x}_t^i|\mathbf{x}_{t-1}^i,\mathbf{u}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1}^i,\mathbf{y}_t,\mathbf{u}_t)}$
  - Resampling: Draw new particles with replacement with probability proportional to weights

SIR Algorithm

- At each time step t:

$\eta = 0$

for $i = 1{:}N$

$\mathbf{x}_t^i \sim q(\mathbf{x}_t \mid \mathbf{x}_{t-1}^i, \mathbf{y}_t, \mathbf{u}_t)$

$w_t^i = w_{t-1}^i \frac{p(\mathbf{y}_t|\mathbf{x}_t^i)\, p(\mathbf{x}_t^i|\mathbf{x}_{t-1}^i, \mathbf{u}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1}^i, \mathbf{y}_t, \mathbf{u}_t)}$

$\eta = \eta + w_t^i$

end

for $i = 1{:}N$

$w_t^i = w_t^i / \eta$

end

if $N_{eff} <$ threshold

  for $i = 1 : N$

    randomly draw $\overline{\mathbf{x}}_t^i$ with replacement with probability $p(\overline{\mathbf{x}}_t^i = \mathbf{x}_t^i) \propto w_t^i$

  end

  for $i = 1 : N$

    $\mathbf{x}_t^i \leftarrow \overline{\mathbf{x}}_t^i$

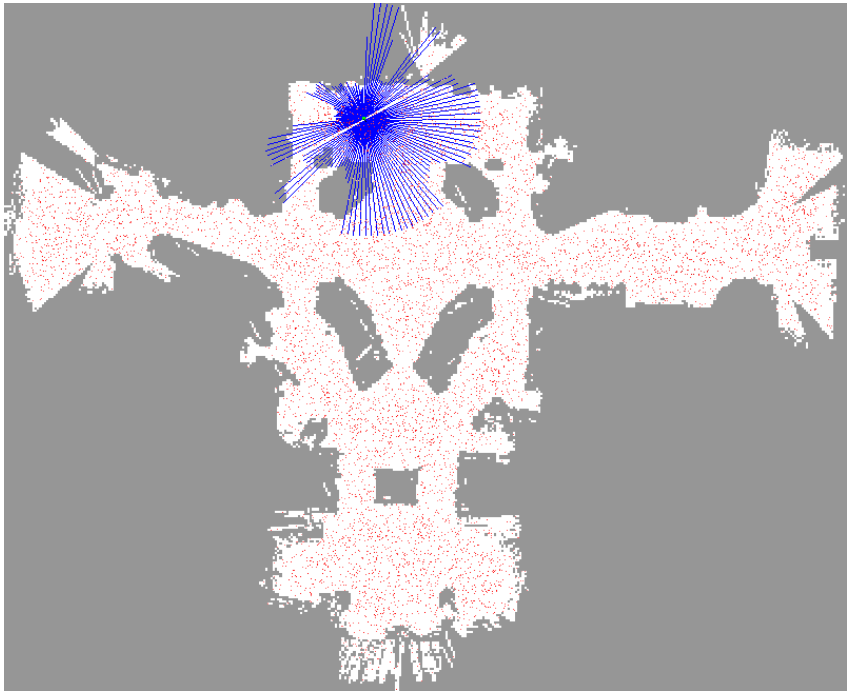
    $w_t^i \leftarrow 1$

  end

end

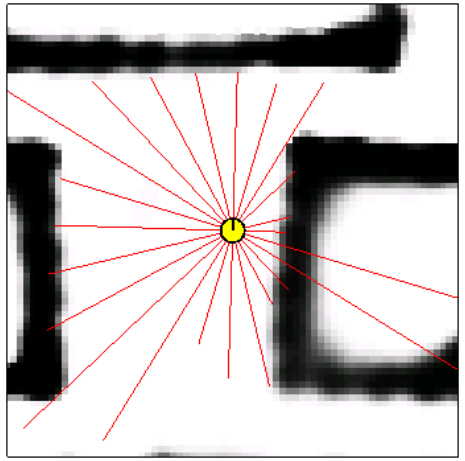
Example: Laser-based Monte Carlo Localization

- Where is the robot (position and orientation)?
- Indoor environment
- Robot moves on flat ground (2D plane)
- Laser sensor measures distance to obstacles in a 2D plane
- Known map

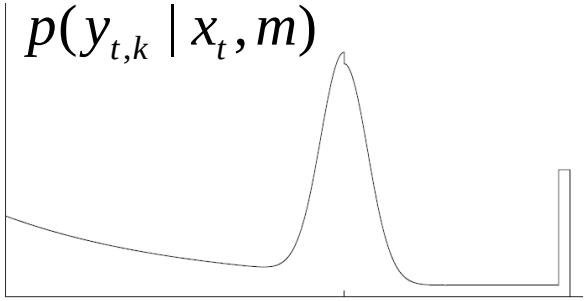


Example Observation Model

- Measurement can be due …
  - a known obstacle
  - an unexpected obstacle (people, furniture, …)
  - missing all obstacles (total reflection, glass, …).
- Noise is due to uncertainty …
  - in measuring distance to known obstacle.
  - in position of known obstacles.
  - in position of additional obstacles.
  - whether obstacle is missed.

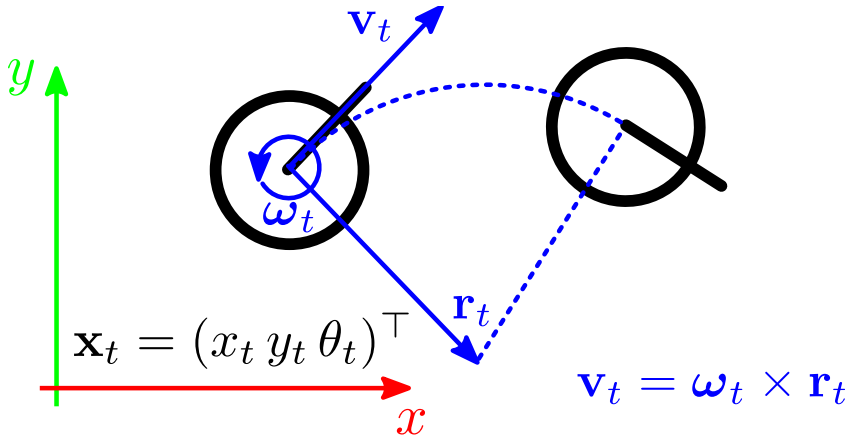


$$p(y_t \mid x_t, m) = \prod_{k=1}^{K} p(y_{t,k} \mid x_t, m)$$



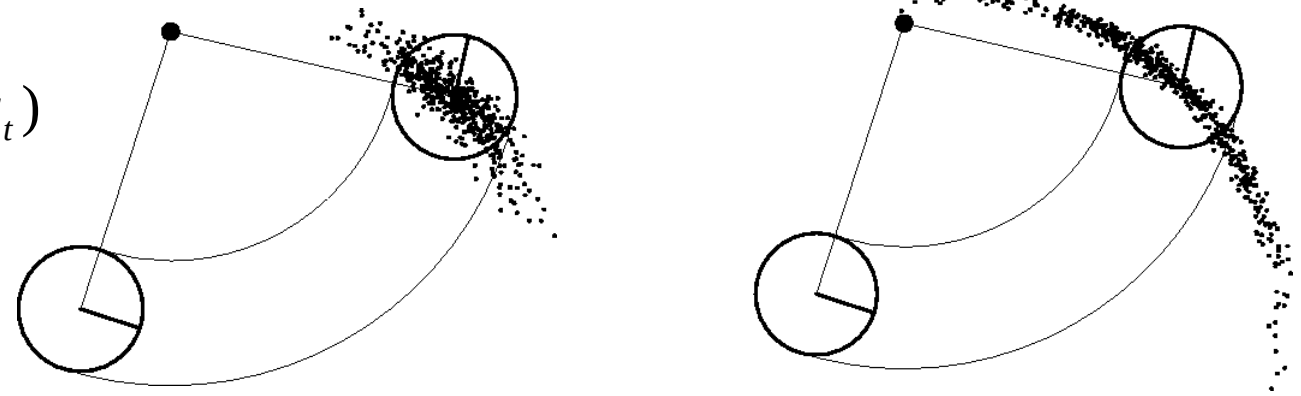
Example State-Transition Model

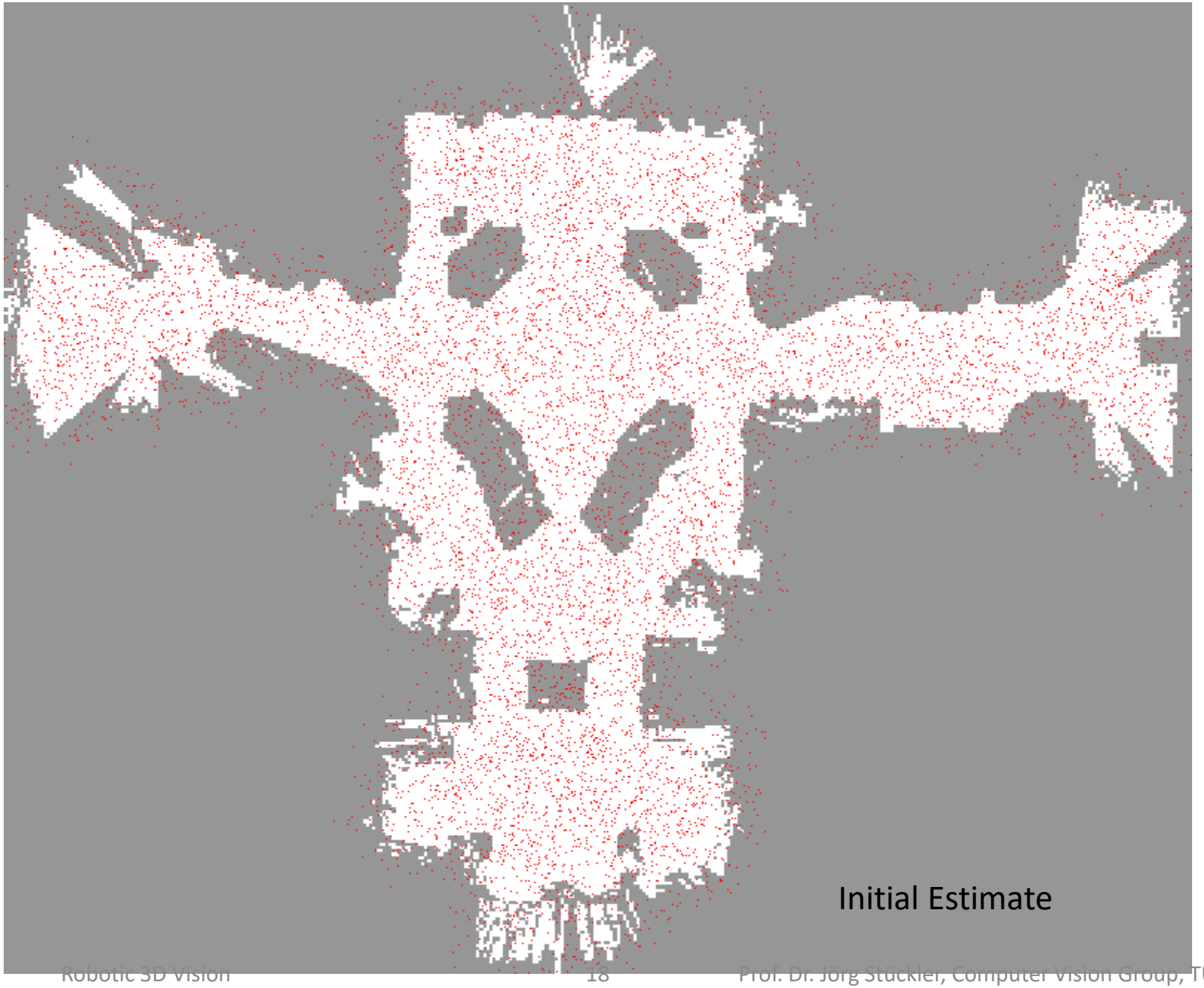
- Velocity-based motion model of a robot in the 2D plane
- Robot actions:
  - Linear velocity $\mathbf{v}_t$
  - Rotational velocity $\boldsymbol{\omega}_t$

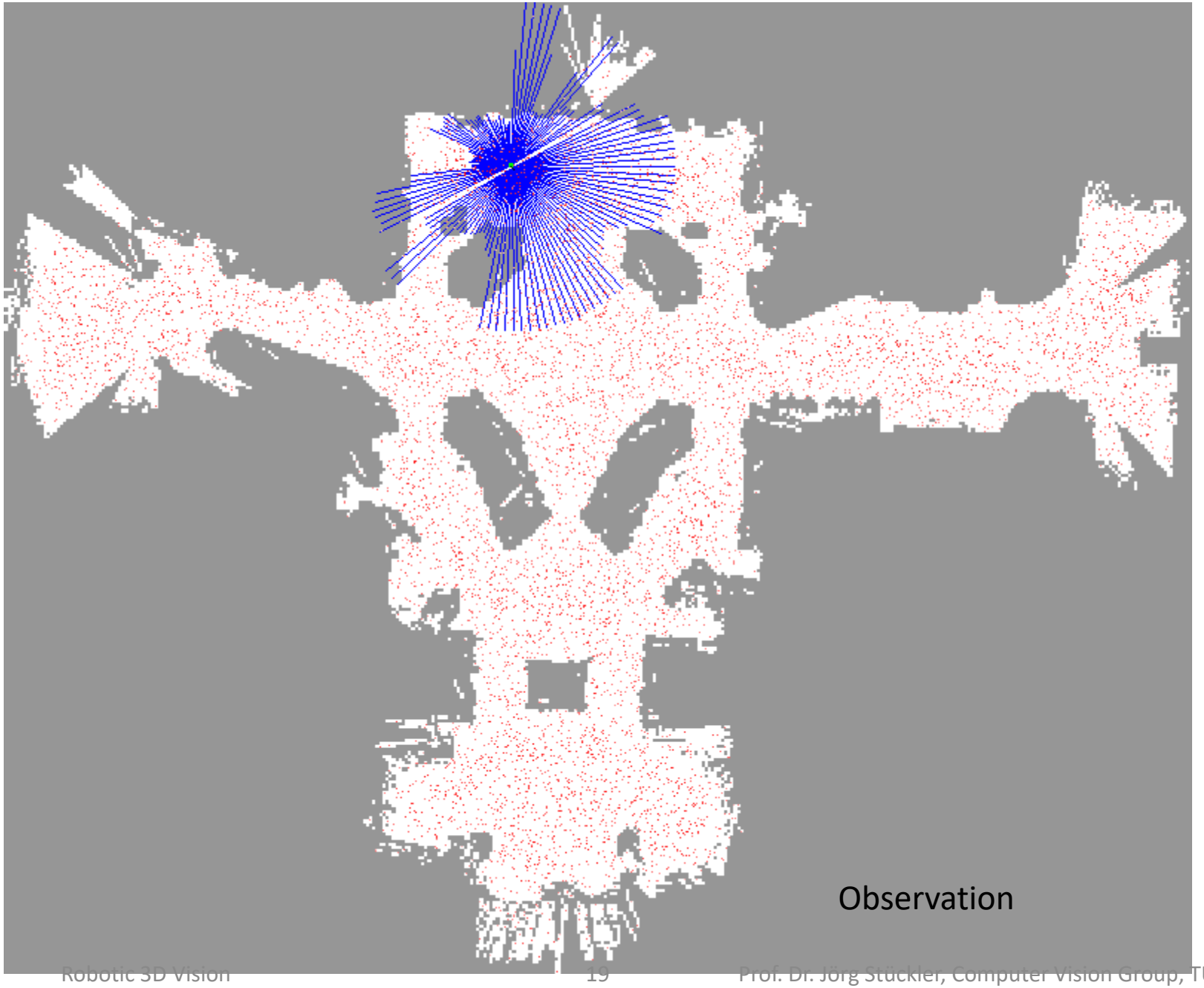


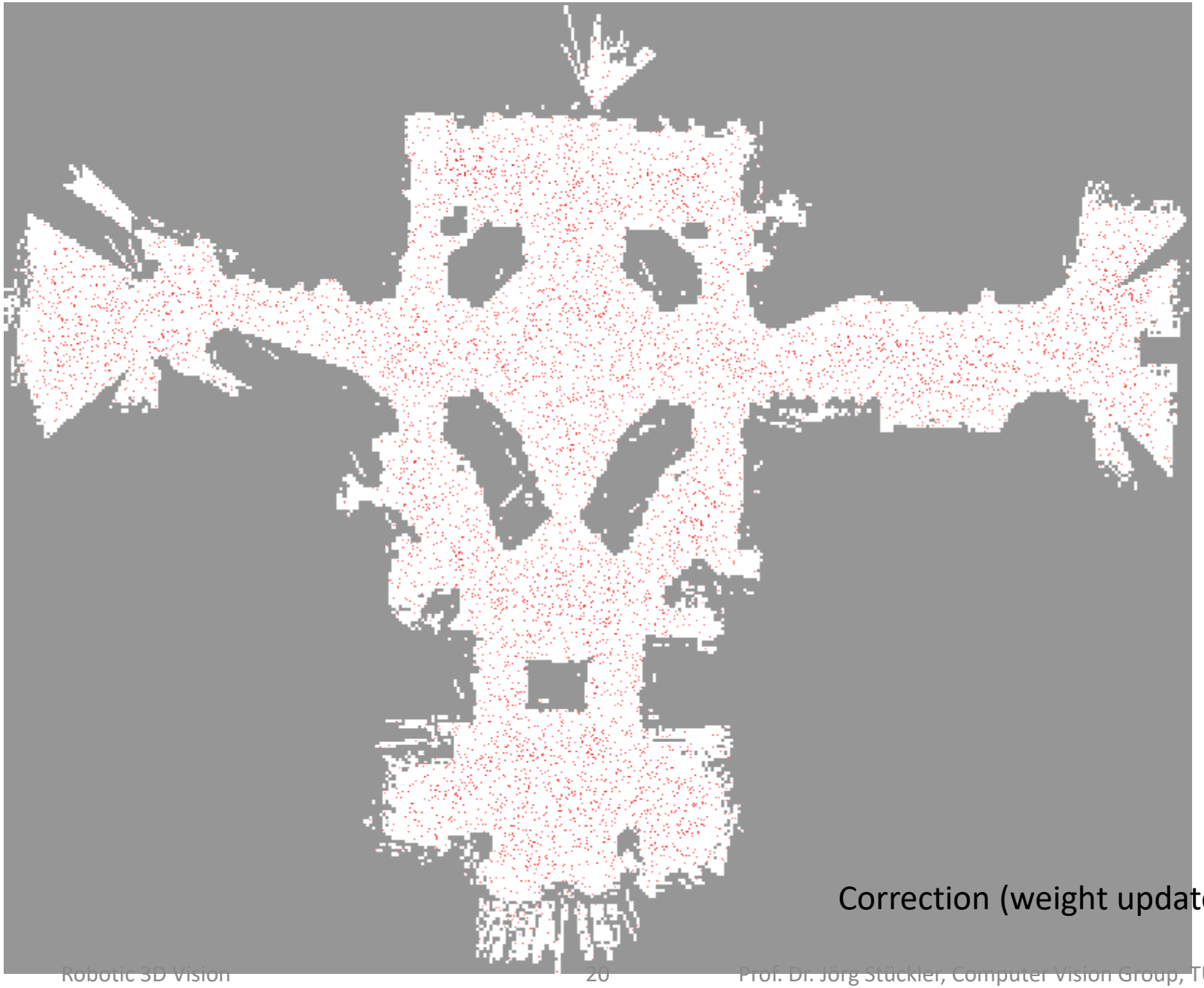
- Actions are executed with uncertainty (f.e. Gaussian noise on velocities)

$p(x_t \mid x_{t-1}, u_t)$

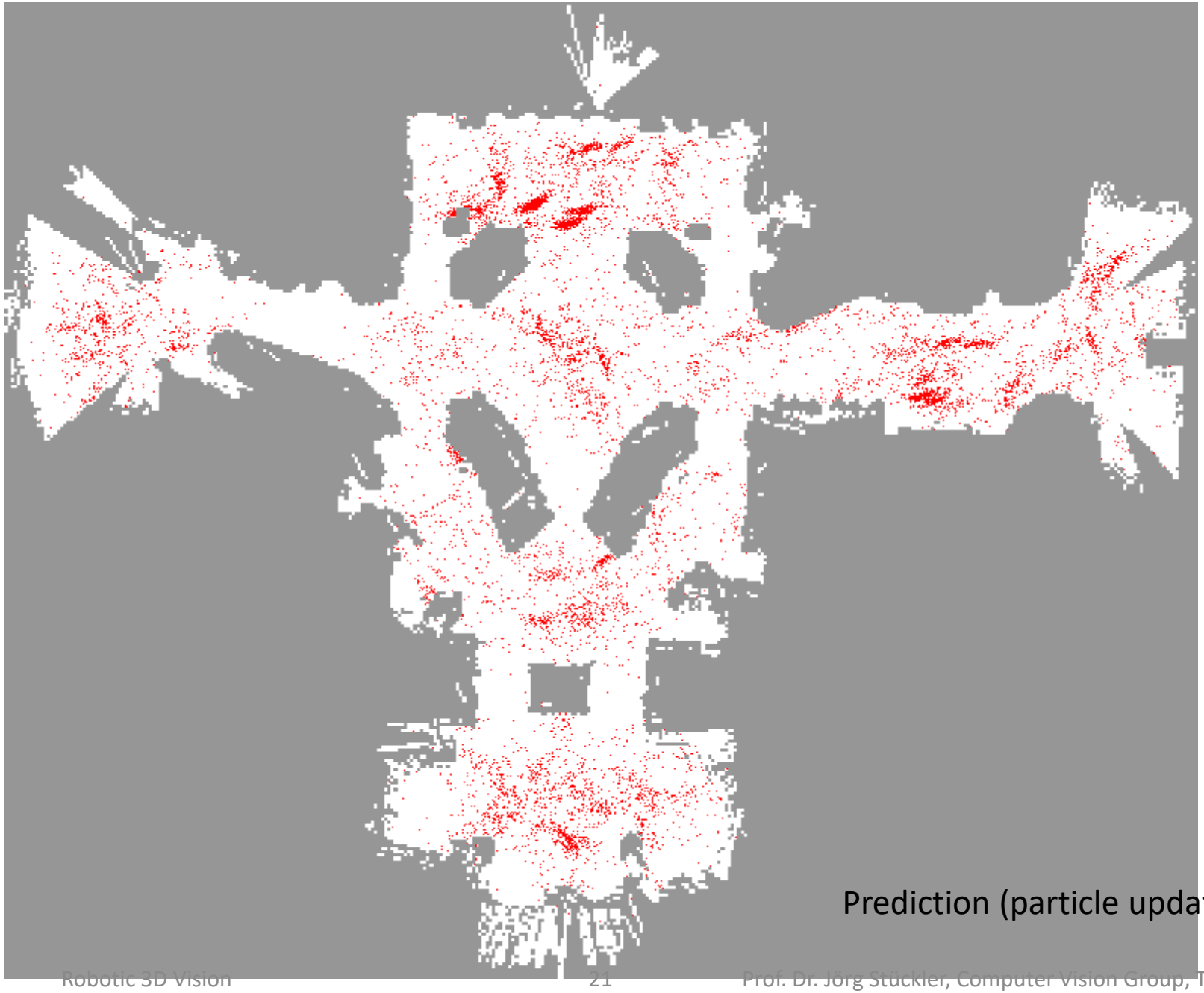



Initial Estimate

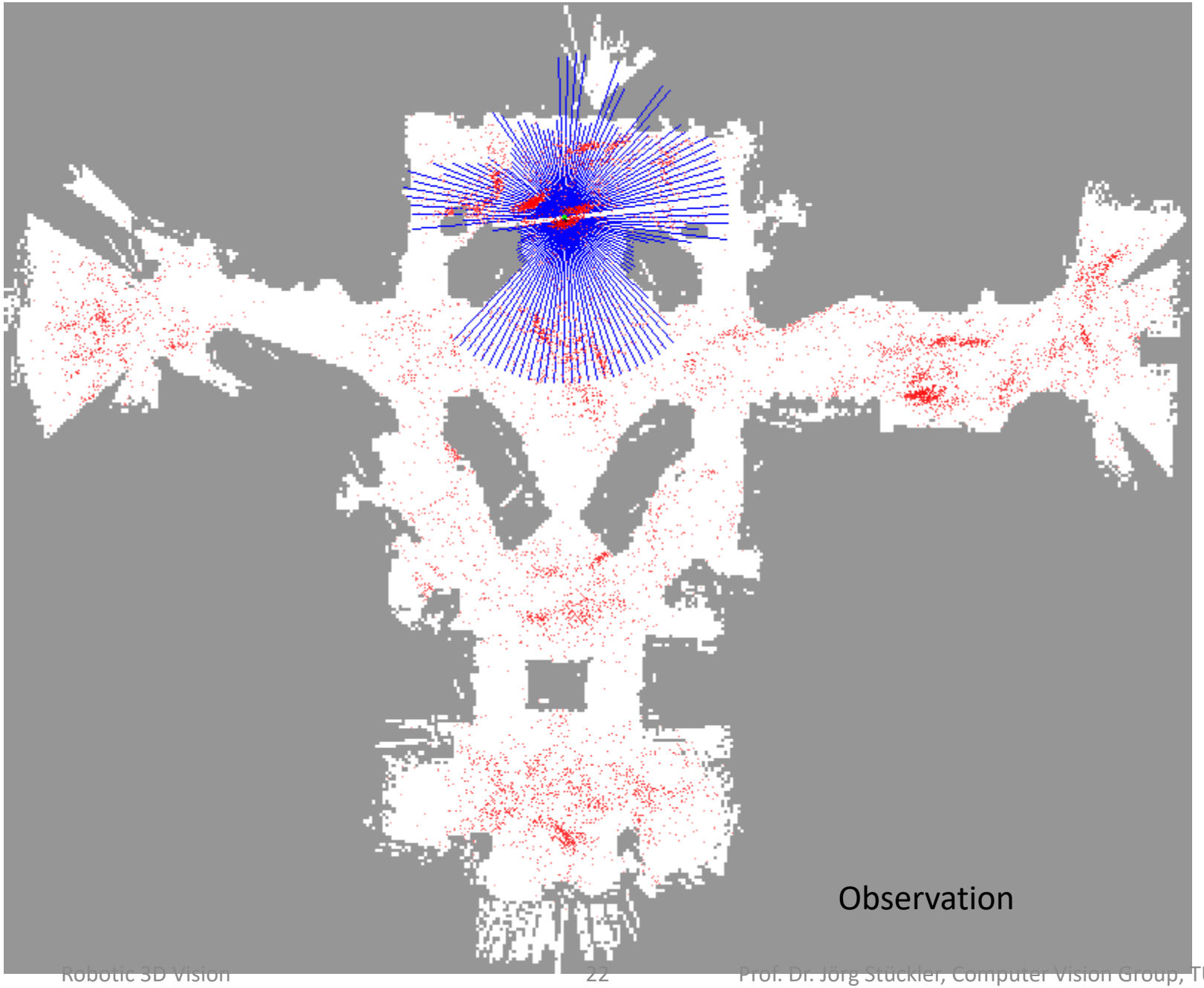

Observation

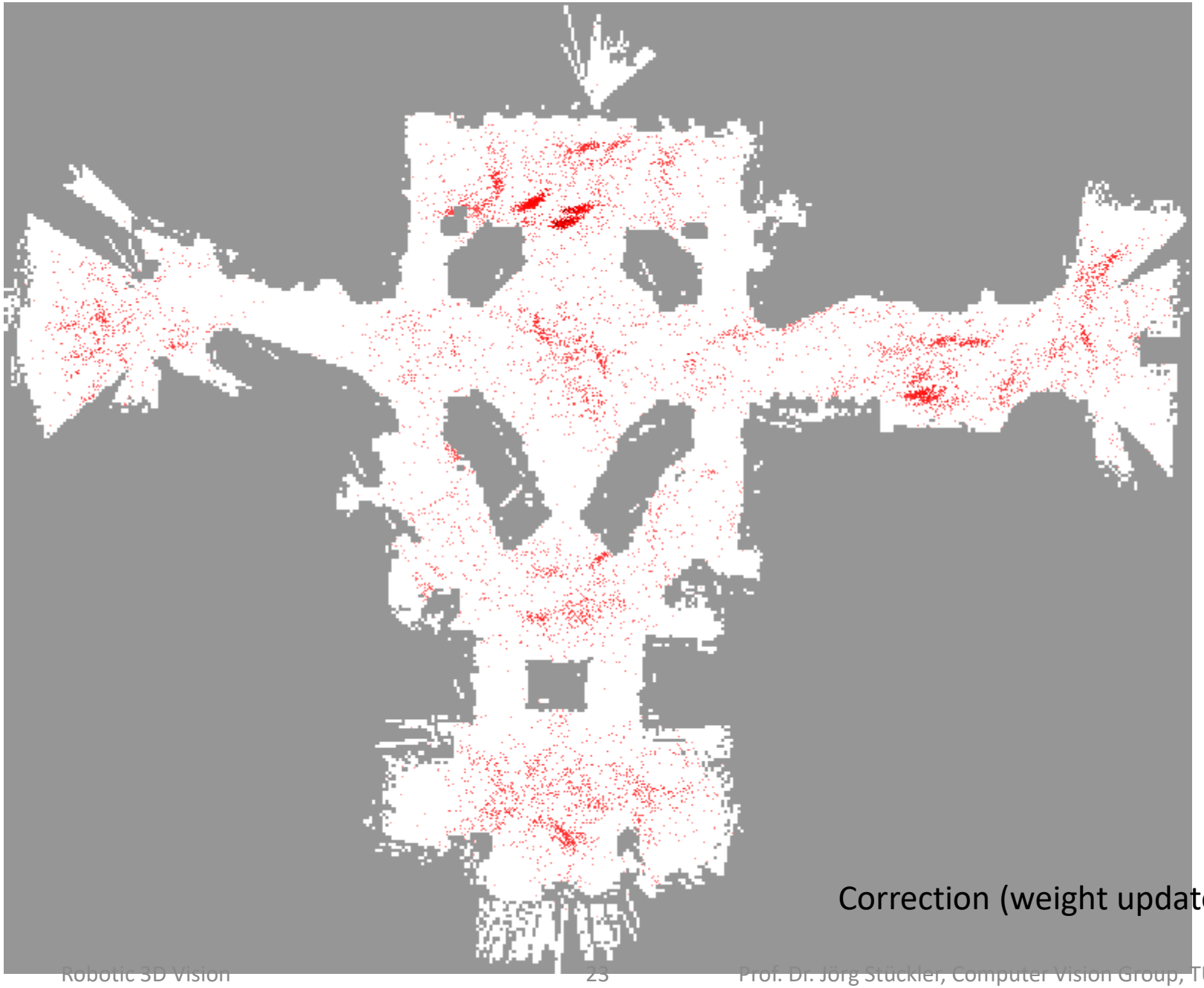


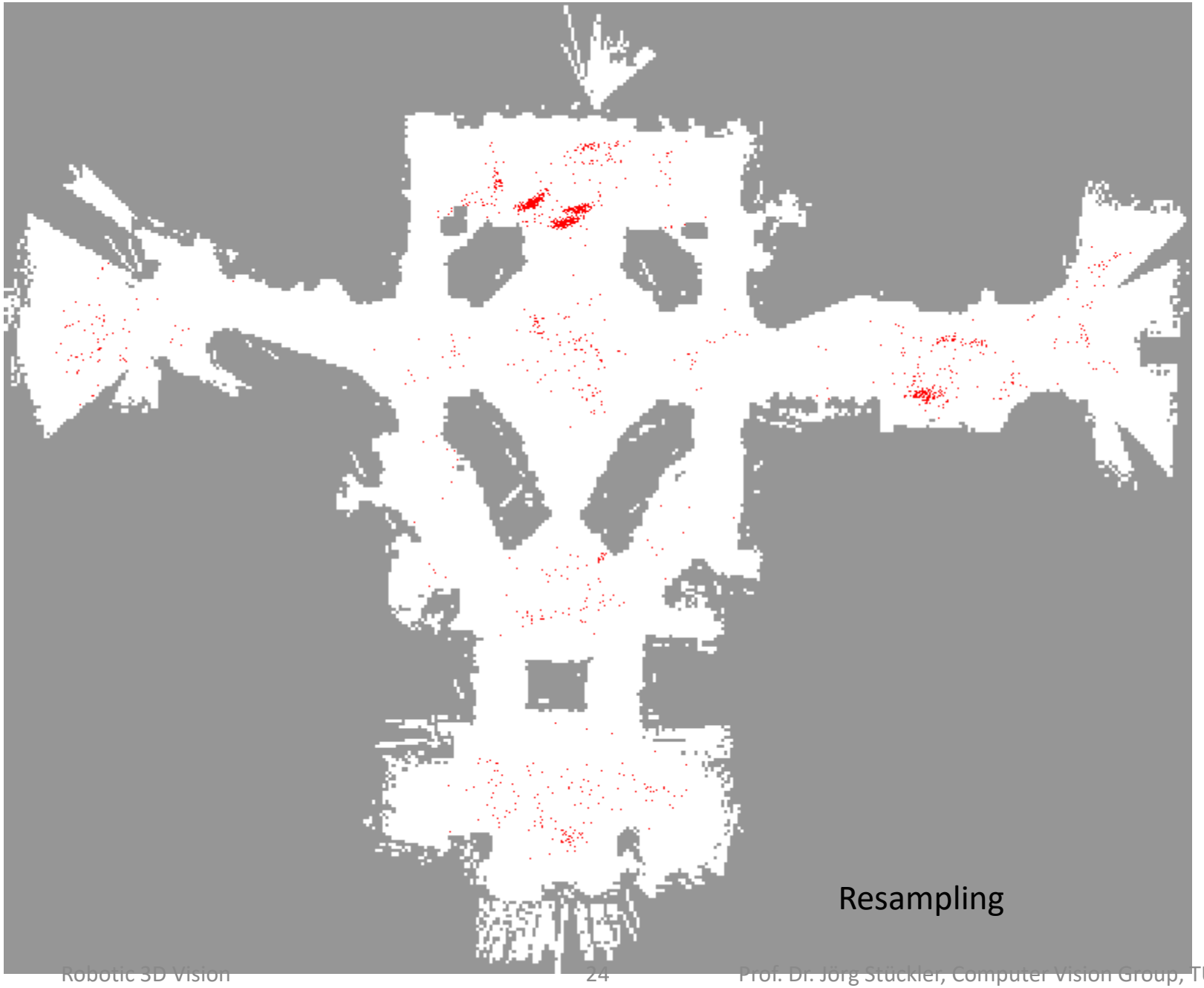
Correction (weight update)

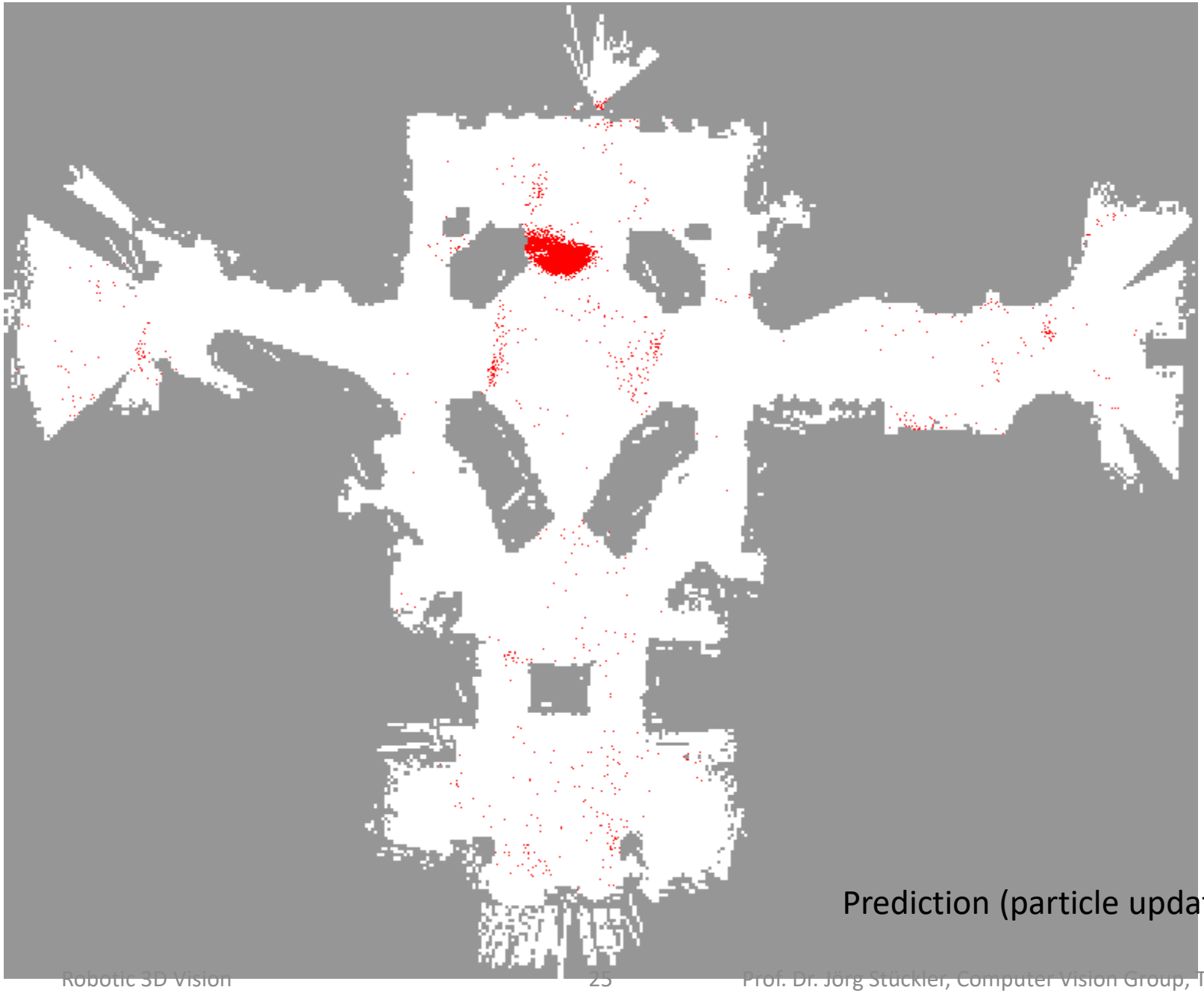


Prediction (particle update)

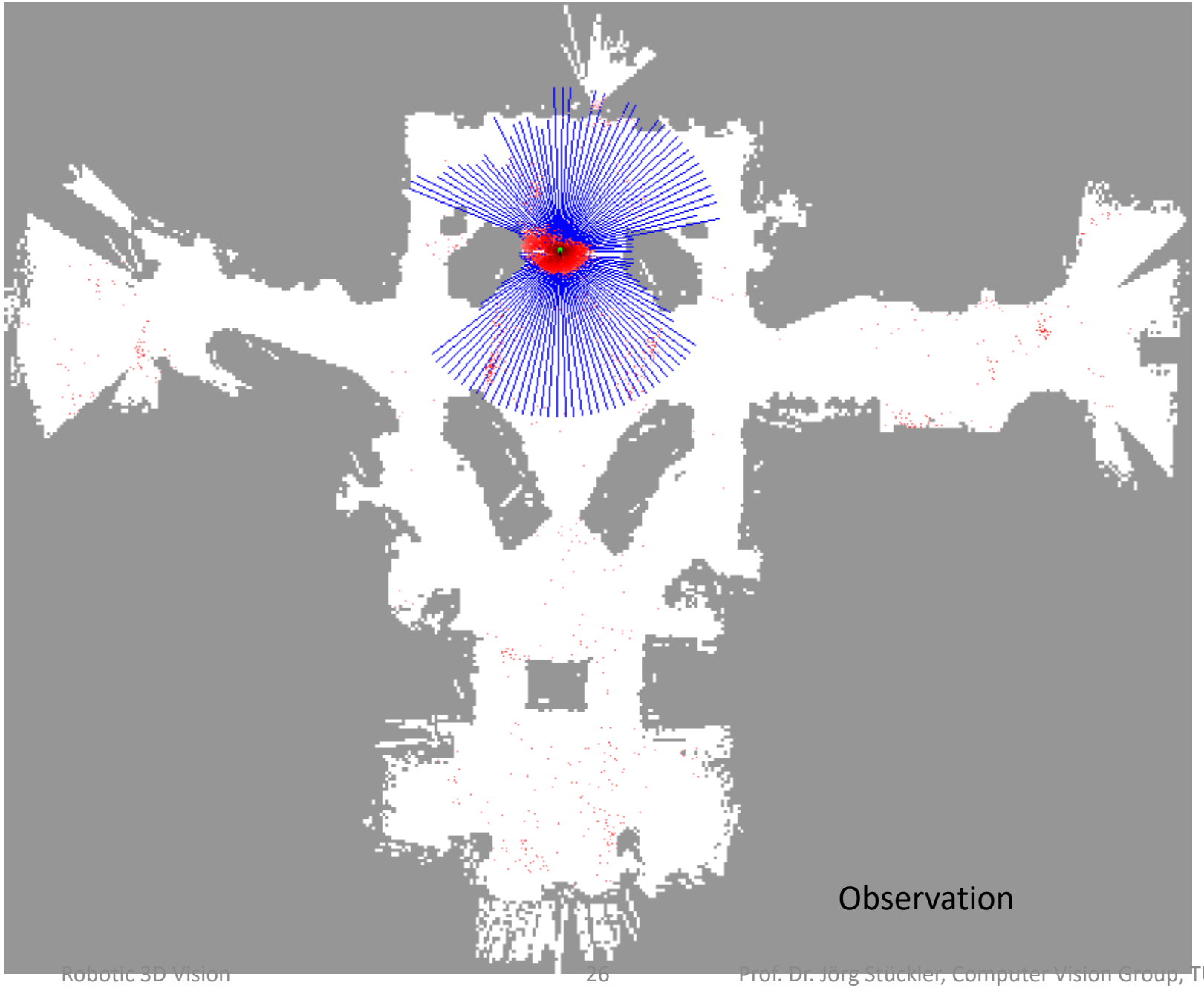

Observation

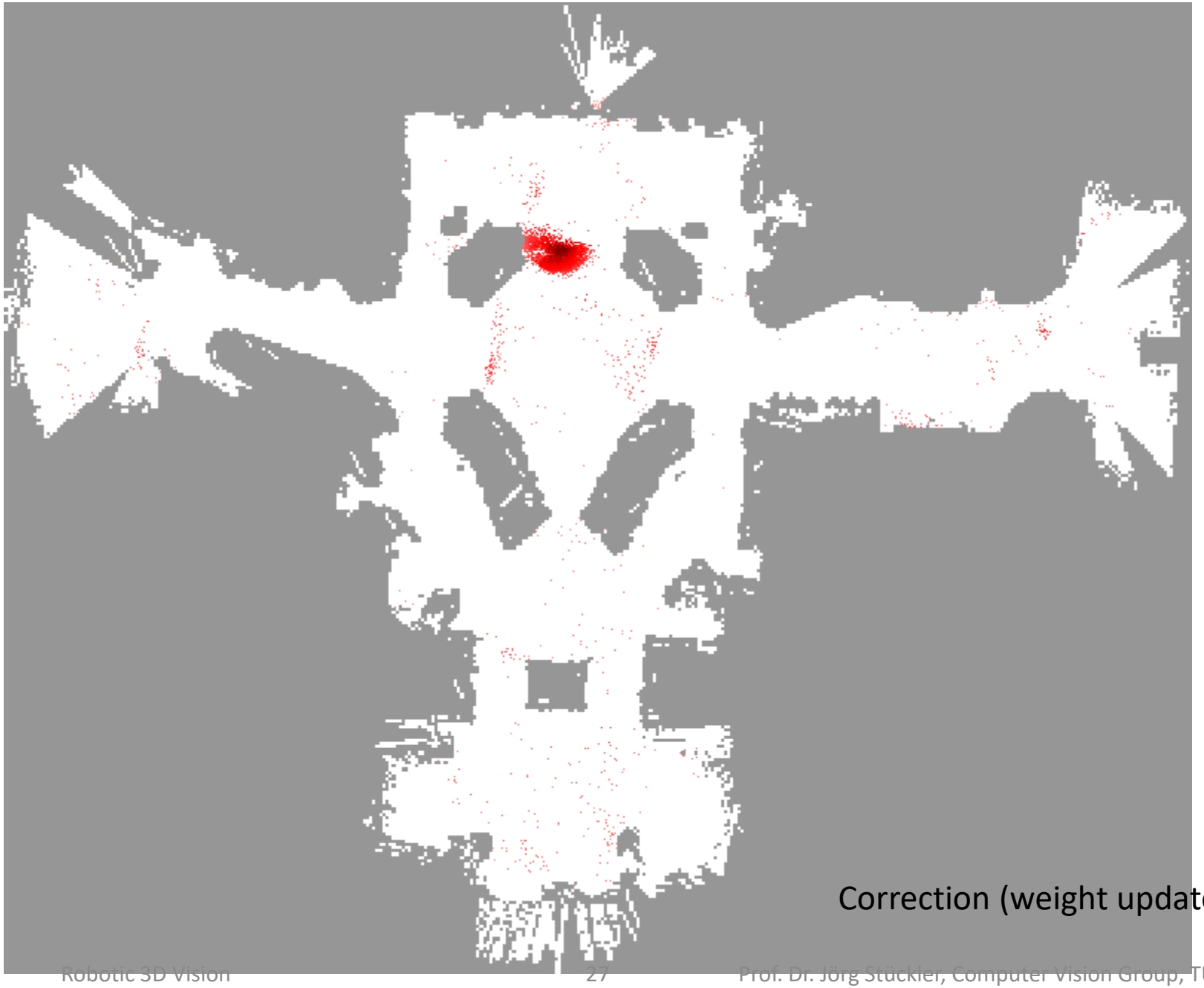

Correction (weight update)


Resampling

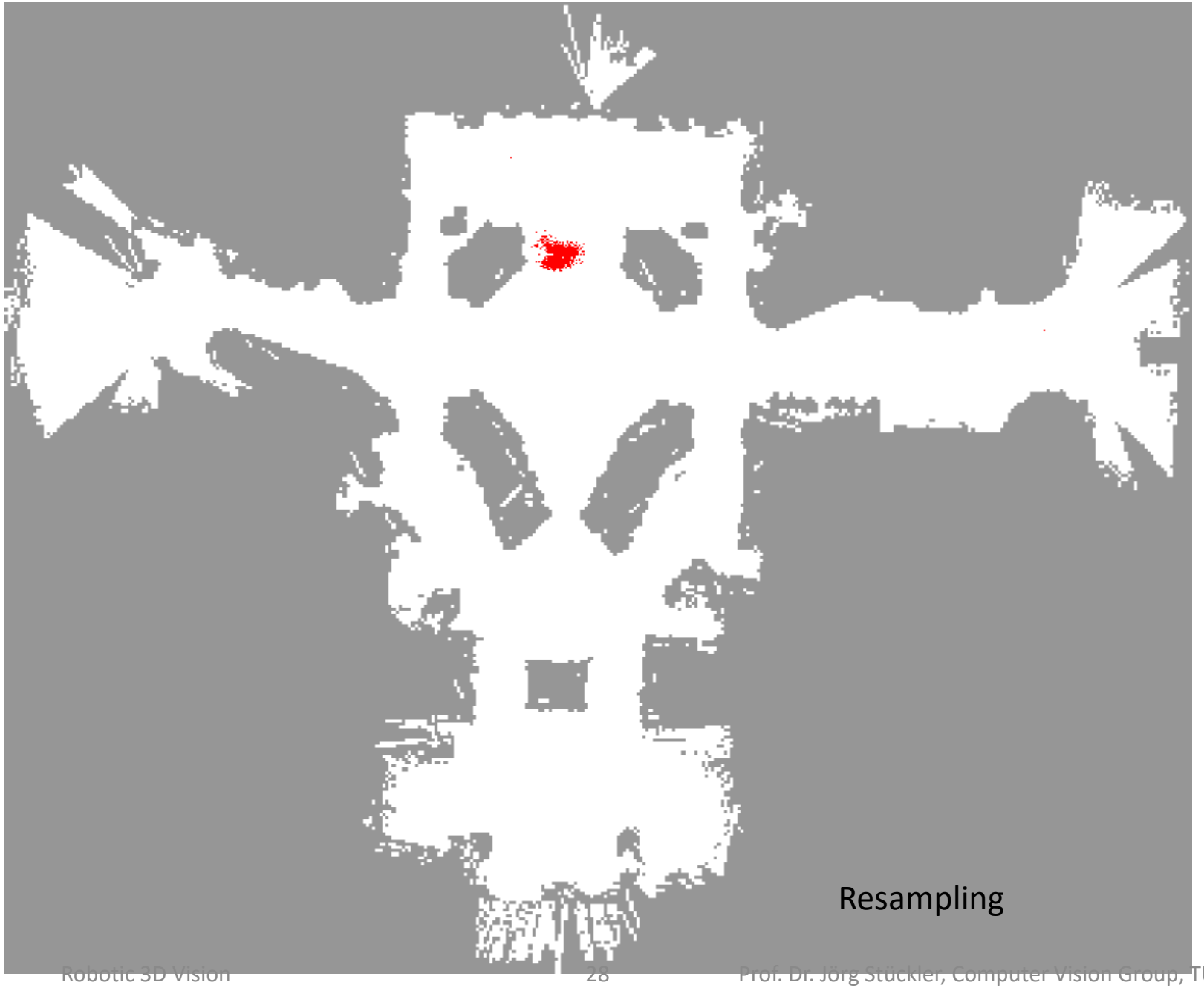


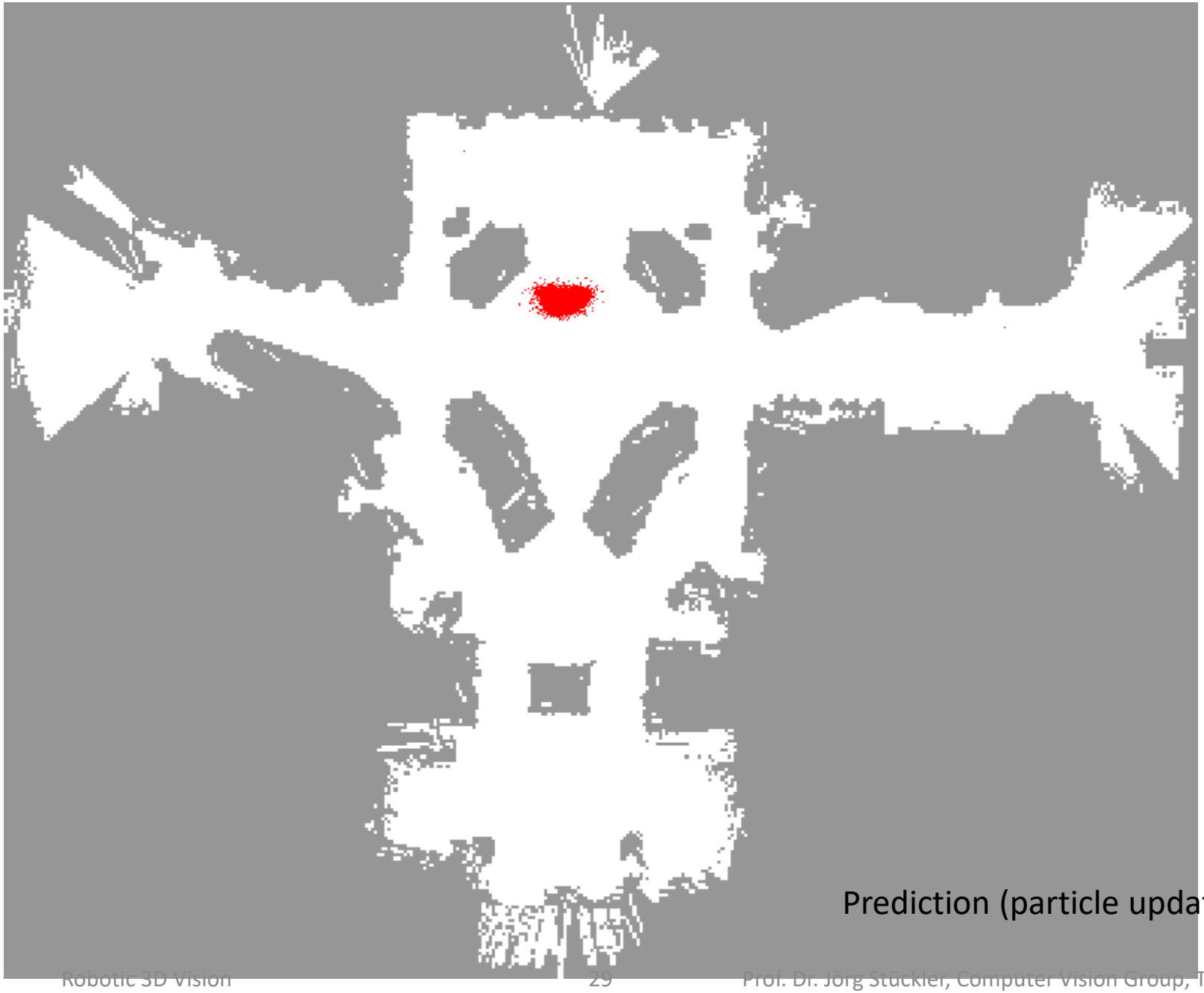
Prediction (particle update)

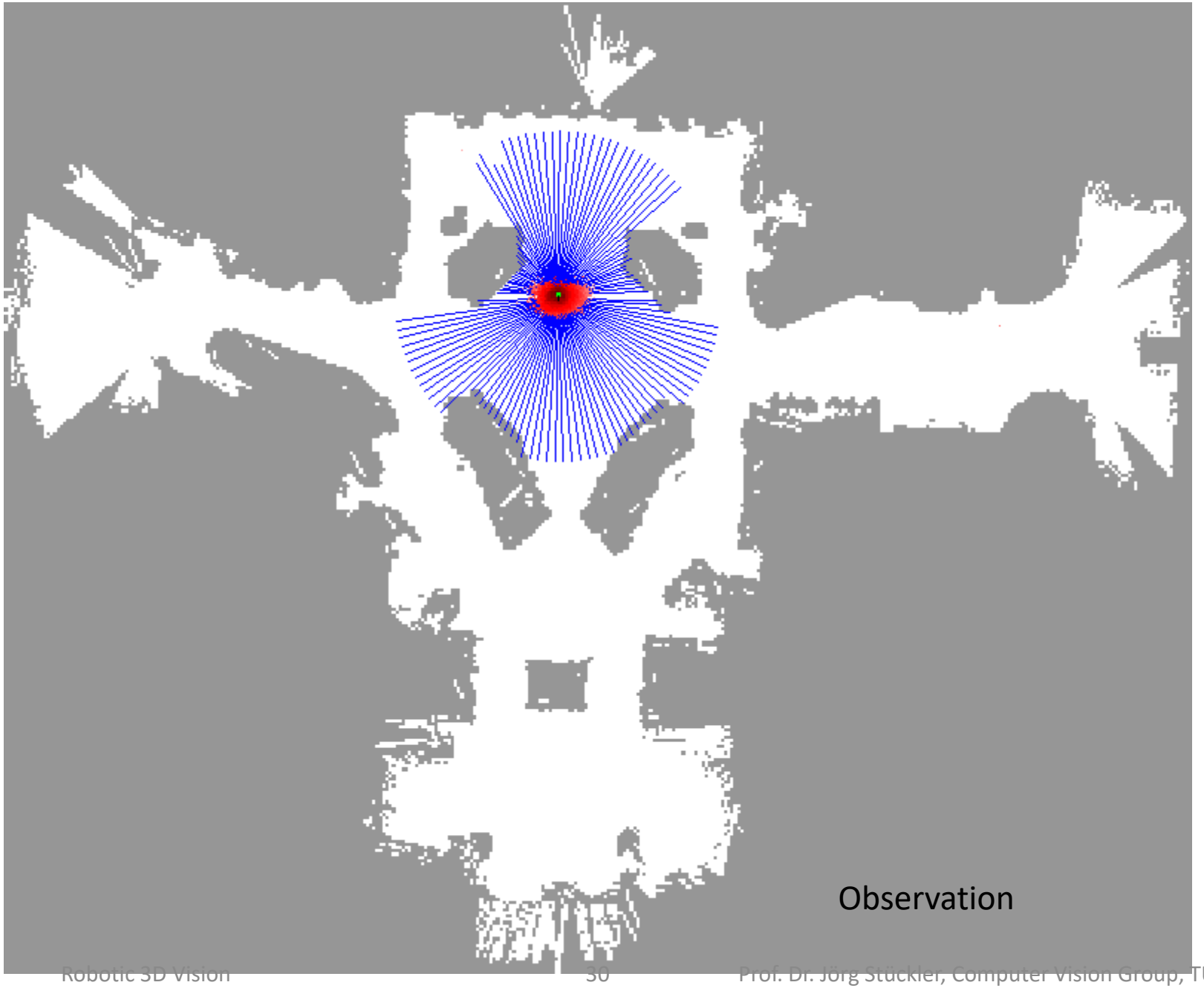

Observation

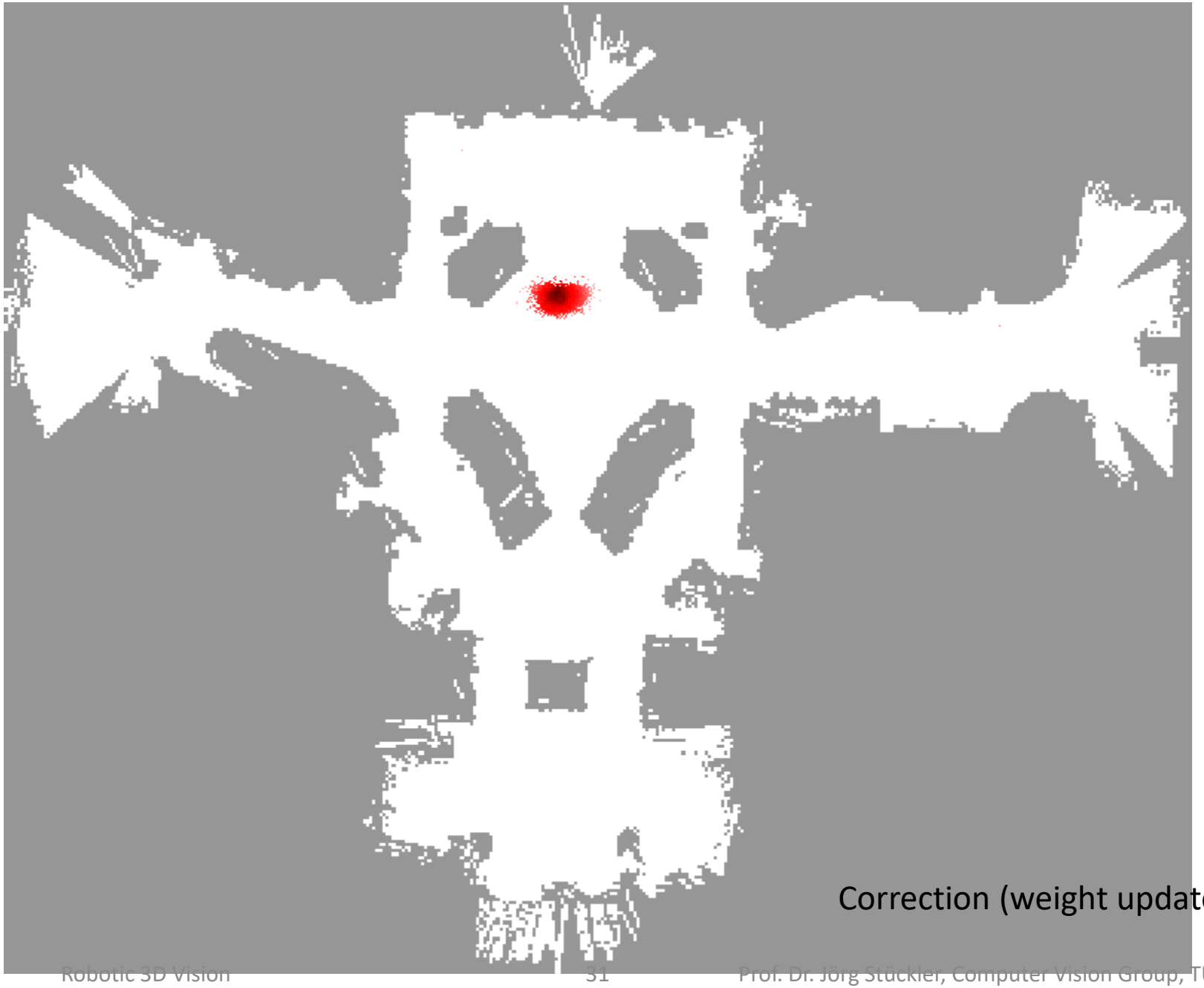


Correction (weight update)

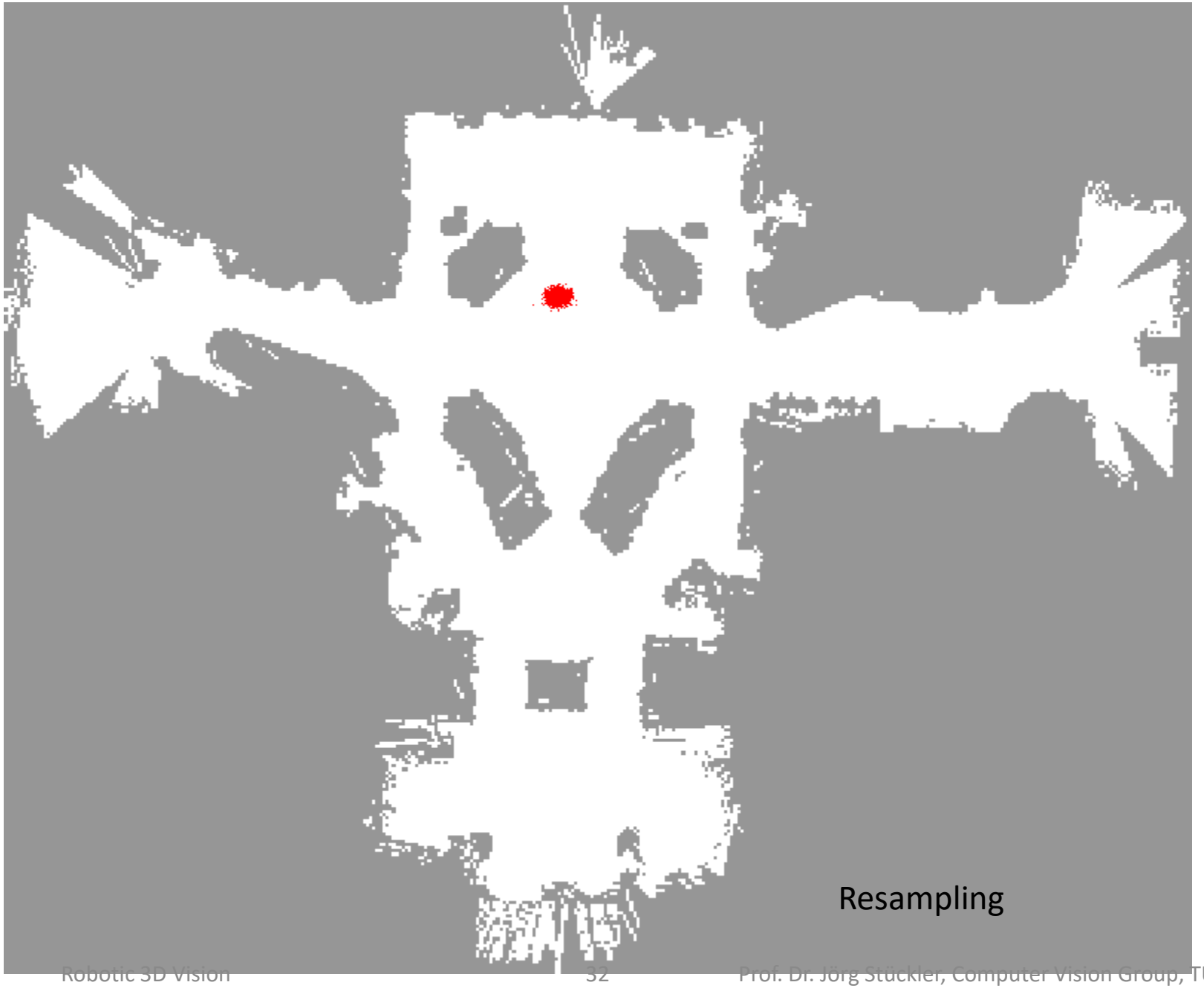

Resampling

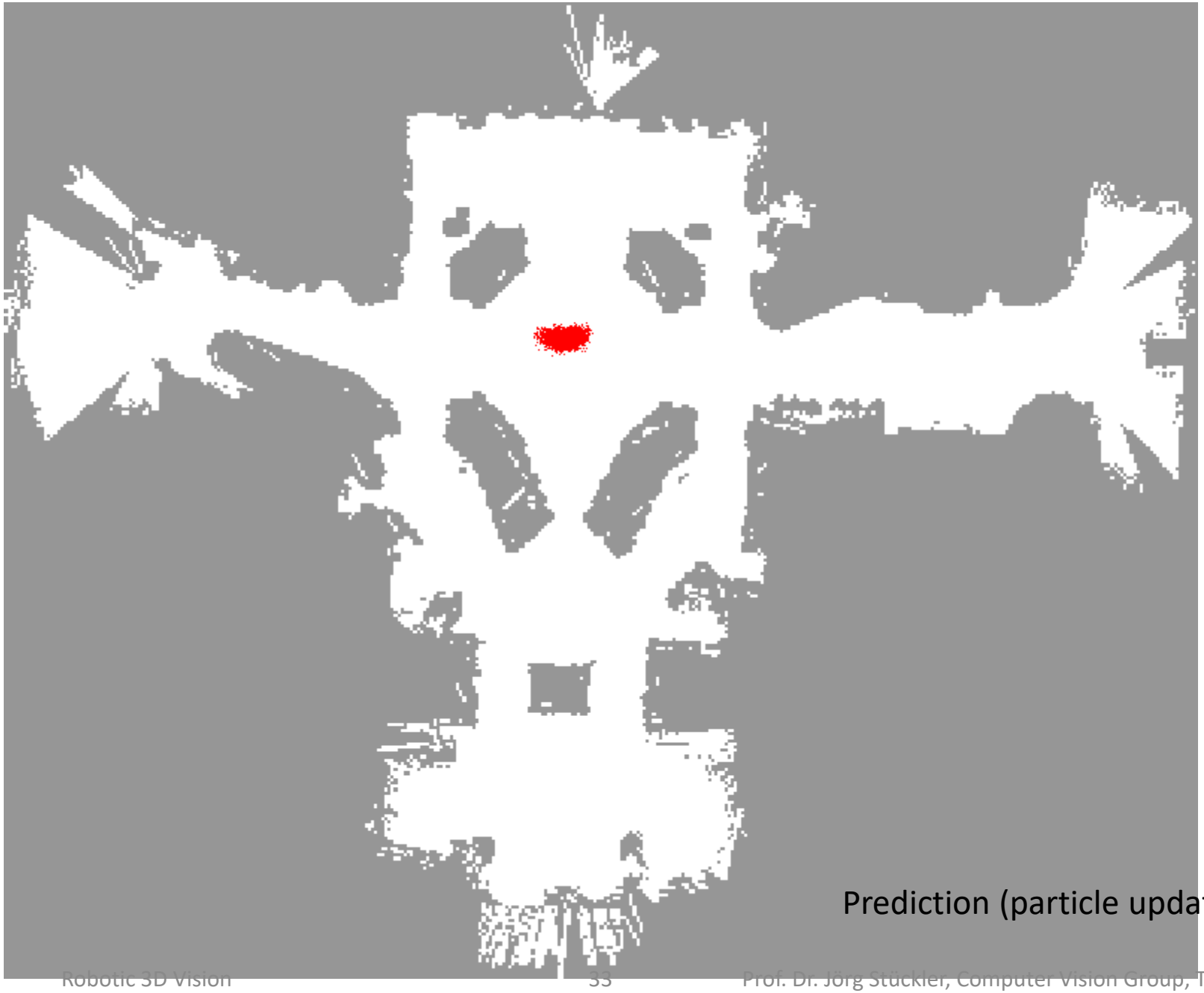

Prediction (particle update)


Observation

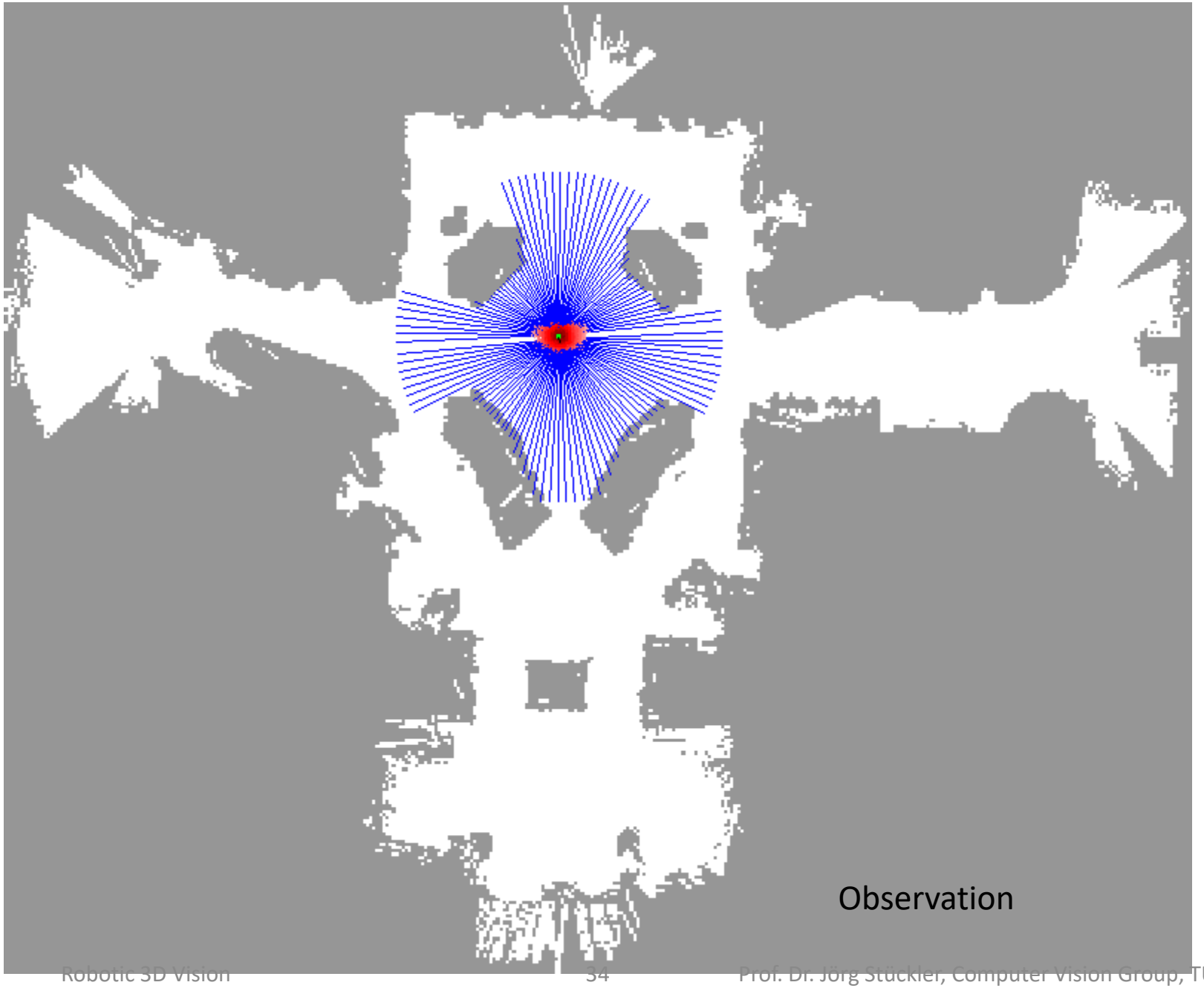


Correction (weight update)


Resampling

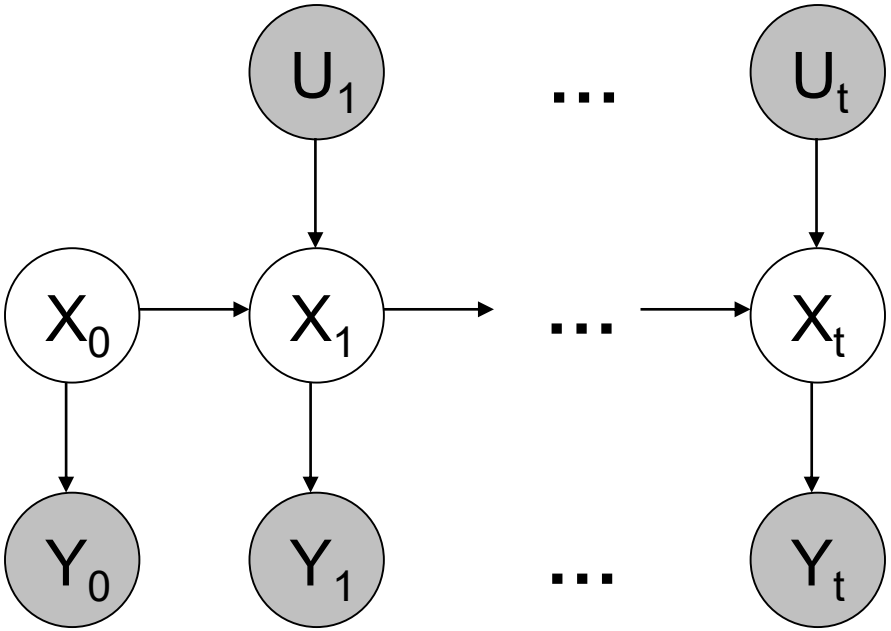


Prediction (particle update)


Observation

Short Recap of Directed Graphical Models

- What is the probabilistic semantics of a directed graphical model?
- Example of a directed graphical model:

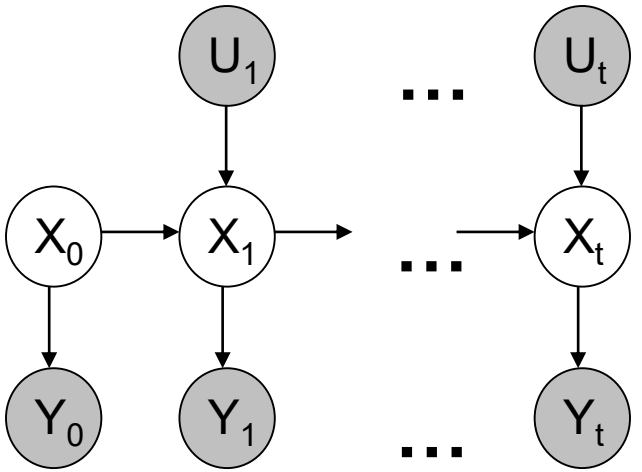


Directed Graphical Models

- Graph describes factorization of joint probability over random variables in terms of conditional probabilities
  - Nodes are random variables
  - Directed edges encode stochastic dependency relations

$$p(X_{0:t}, U_{1:t}, Y_{0:t}) = p(X_0)\left(\prod_{\tau=0}^{t} p(Y_\tau | X_\tau)\right)\left(\prod_{\tau=1}^{t} p(X_\tau | X_{\tau-1}, U_\tau) p(U_\tau)\right)$$

- Bayesian factorization into conditional probabilities of variables conditioned on their parents
- Graph needs to be acyclic
- Also called Bayesian network



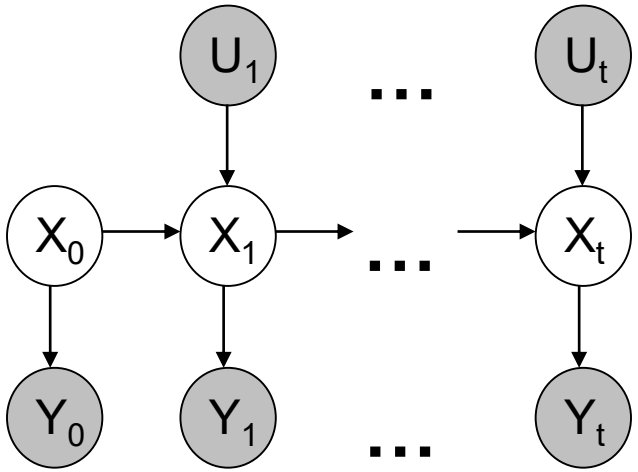
Semantics of Directed Graphical Models

- A directed graphical model expresses stochastic dependency relations between random variables
- Node sets X and Y are conditionally independent given set Z if Z d-separates X and Y in the graph:
  - Every undirected path between nodes in X and Y is blocked by nodes in Z
  - Path is blocked if there is a node W such that either
    - W has no converging arrows along path and W is in Z
    - W has converging arrows along path and neither W nor its descendants are in Z

$$X \perp\!\!\!\perp Y \mid Z$$

iff

$$p(X, Y \mid Z) = p(X \mid Z)\, p(Y \mid Z)$$



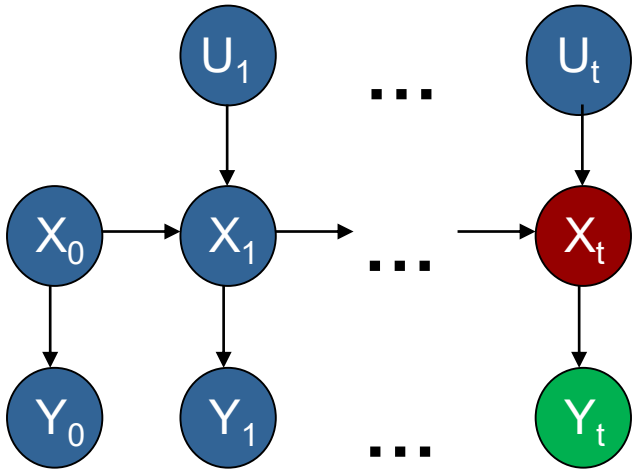
Semantics of Directed Graphical Models

- A directed graphical model expresses stochastic dependency relations between random variables
- Node sets X and Y are conditionally independent given set Z if Z d-separates X and Y in the graph:
  - Every undirected path between nodes in X and Y is blocked by nodes in Z
  - Path is blocked if there is a node W such that either
    - W has no converging arrows along path and W is in Z
    - W has converging arrows along path and neither W nor its descendants are in Z

$$X \perp\!\!\!\perp Y \mid Z \quad ?$$

iff

$$p(\underline{X}, \underline{Y} \mid \underline{Z}) = p(X \mid Z)\, p(Y \mid Z)$$



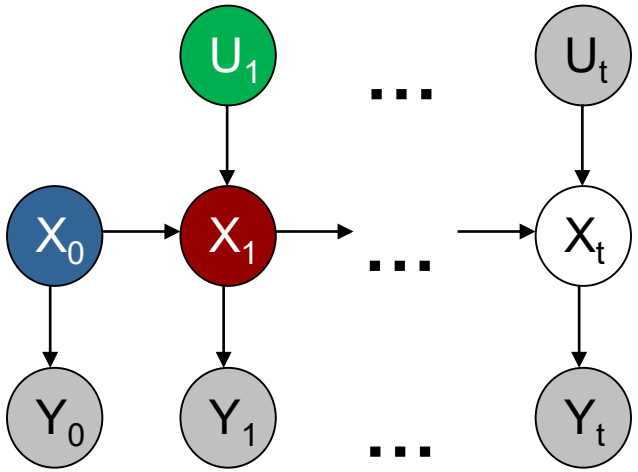
Semantics of Directed Graphical Models

- A directed graphical model expresses stochastic dependency relations between random variables
- Node sets X and Y are conditionally independent given set Z if Z d-separates X and Y in the graph:
  - Every undirected path between nodes in X and Y is blocked by nodes in Z
  - Path is blocked if there is a node W such that either
    - W has no converging arrows along path and W is in Z
    - W has converging arrows along path and neither W nor its descendants are in Z

$$X \perp\!\!\!\perp Y \mid Z \quad ?$$

iff

$$p(\underline{X}, \underline{Y} \mid \underline{Z}) = p(X \mid Z)\, p(Y \mid Z)$$



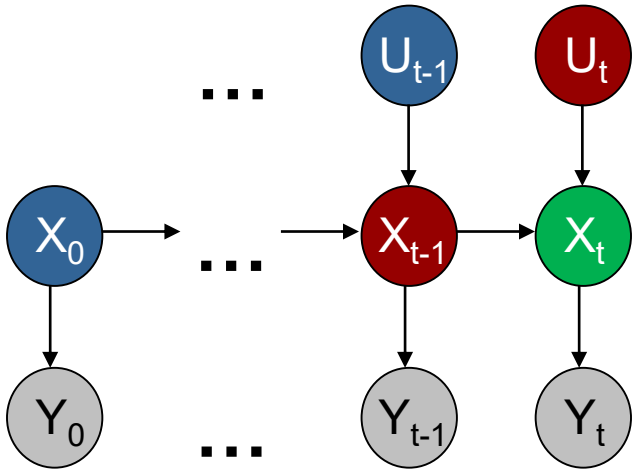
Semantics of Directed Graphical Models

- A directed graphical model expresses stochastic dependency relations between random variables
- Node sets X and Y are conditionally independent given set Z if Z d-separates X and Y in the graph:
  - Every undirected path between nodes in X and Y is blocked by nodes in Z
  - Path is blocked if there is a node W such that either
    - W has no converging arrows along path and W is in Z
    - W has converging arrows along path and neither W nor its descendants are in Z

$$X \perp\!\!\!\perp Y \mid Z \quad ?$$

iff

$$p(\underline{X}, \underline{Y} \mid \underline{Z}) = p(X \mid Z)\, p(Y \mid Z)$$



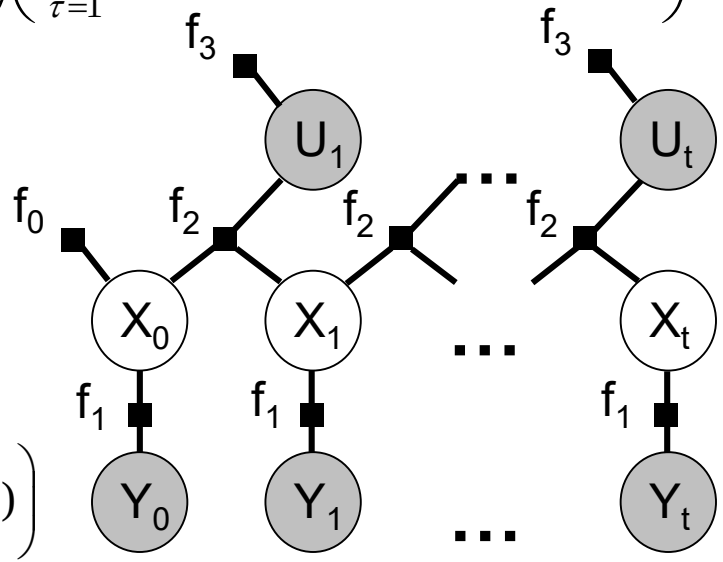
Factor Graphs

- Factor graphs also describe factorization of joint probability over random variables
  - Nodes are either random variables (discs) or factors (squares)
  - Undirected edges connect variables with factors

$$p(X_{0:t}, U_{1:t}, Y_{0:t}) = \frac{1}{Z} f_0(X_0) \left(\prod_{\tau=0}^{t} f_1(X_\tau, Y_\tau) \right) \left(\prod_{\tau=1}^{t} f_2(X_\tau, X_{\tau-1}, U_\tau) f_3(U_\tau) \right)$$

- Factors f_i are non-negative functions over variables
- Normalizer Z ensures that probability function integrates to 1

$$Z := \iiint_{X_{0:t}, Y_{0:t}, U_{1:t}} f_0(X_0) \left(\prod_{\tau=0}^{t} f_1(X_\tau, Y_\tau) \right) \left(\prod_{\tau=1}^{t} f_2(X_\tau, X_{\tau-1}, U_\tau) f_3(U_\tau) \right)$$



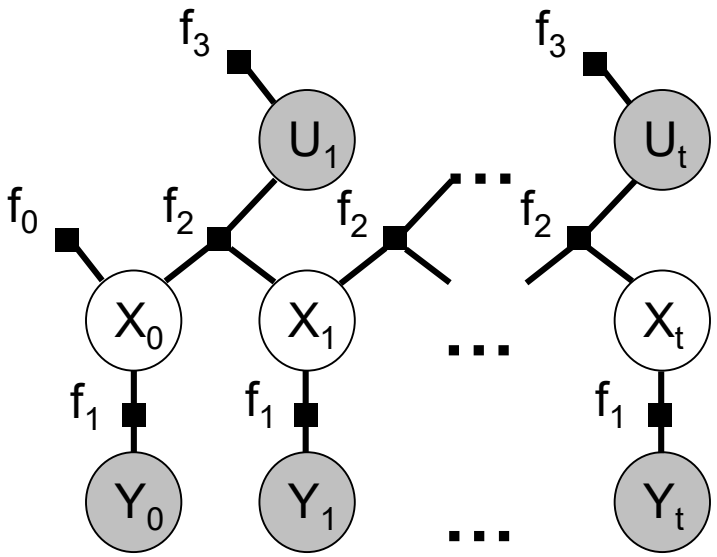
Semantics of Factor Graphs

- Factor graphs also express conditional independence relations
  - Two nodes are neighbors if they appear in a common factor
  - Path: sequence of neighboring nodes between two variables
  - Node set X is conditionally independent from set Y given set Z if all paths between nodes in X and Y are blocked by some node in Z
  - Given the set ne(X) of neighbors of X, the variable X is conditionally independent from the remaining variables

$$X \perp\!\!\!\perp Y \mid Z$$

iff

$$p(X, Y \mid Z) = p(X \mid Z)\, p(Y \mid Z)$$



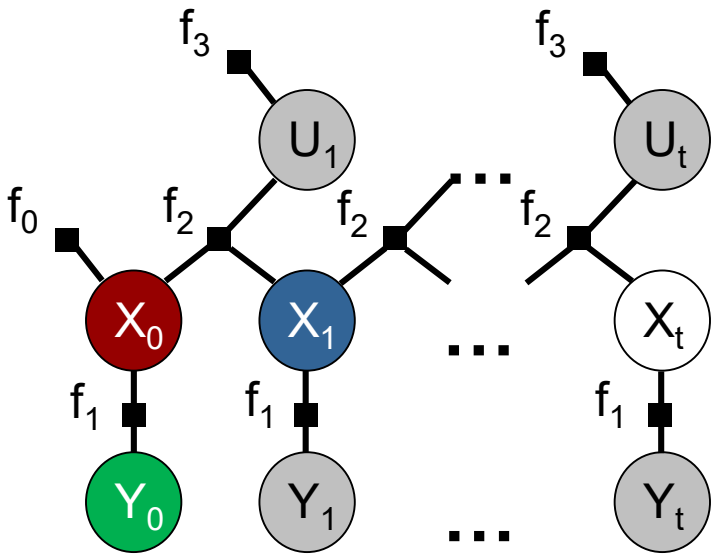
Semantics of Factor Graphs

- Factor graphs also express conditional independence relations
  - Two nodes are neighbors if they appear in a common factor
  - Path: sequence of neighboring nodes between two variables
  - Node set X is conditionally independent from set Y given set Z if all paths between nodes in X and Y are blocked by some node in Z
  - Given the set ne(X) of neighbors of X, the variable X is conditionally independent from the remaining variables

$$X \perp\!\!\!\perp Y \mid Z \quad ?$$

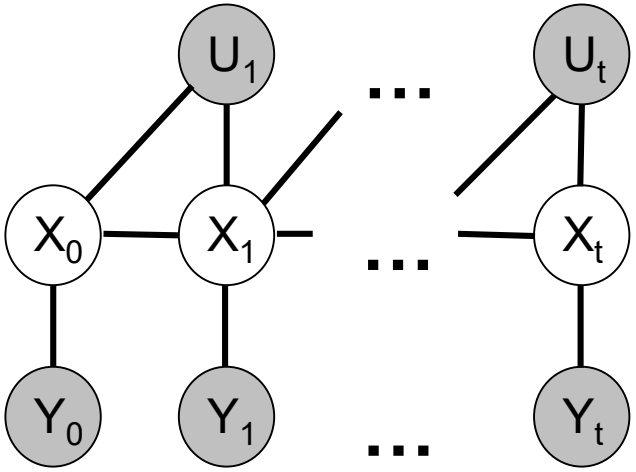
iff

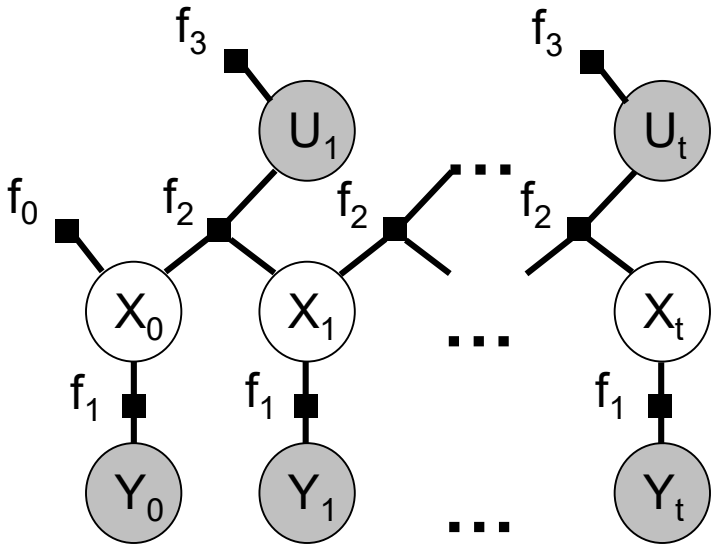
$$p(\underline{X}, \underline{Y} \mid \underline{Z}) = p(X \mid Z)\, p(Y \mid Z)$$



Factor Graphs and Undirected Graphical Models

- Another class of graphical models are undirected graphical models
  - Nodes are random variables (discs)
  - Undirected edges connect variables
- Undirected graphical models can be represented by factor graphs
  - Each factor node in a factor graph connects the cliques in an undirected graphical model





Factor Graphs vs. Directed Graphical Models

- In general, neither factor graphs nor directed graphical models can represent conditional independence relations of arbitrary probability distributions
- Both types of graphs have different expressiveness
- Examples on blackboard

Factor Graphs vs. Undirected Graphical Models

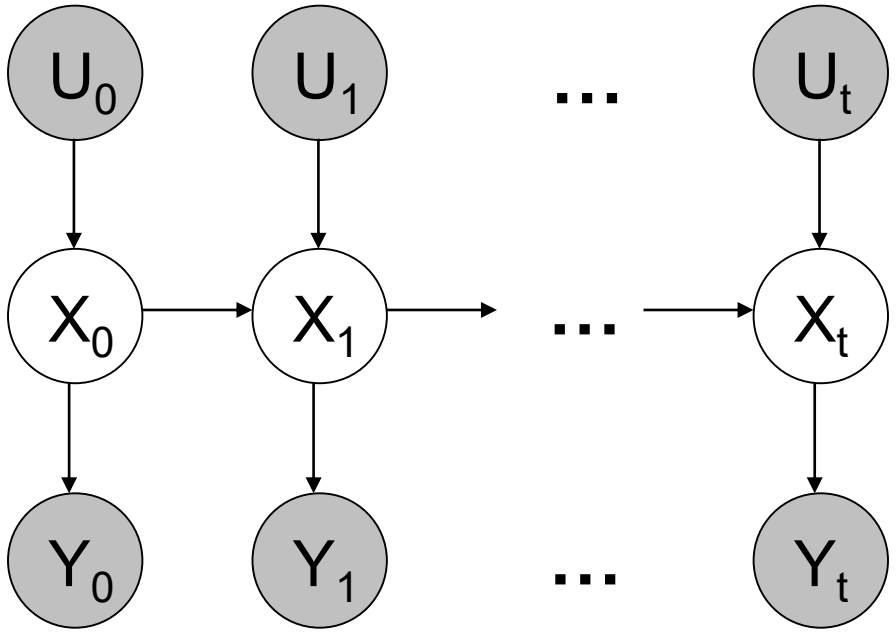
- Factor graphs and undirected graphical models have same expressiveness in terms of conditional independencies
- However, factor graphs can represent different granularities of factorizations for the same conditional dependencies
- Example on blackboard

Full State Posterior

- In filtering, we only consider most recent observation and action
- While Kalman filters are optimal for linear models and Gaussian noise, any approximation (EKF, PF, etc.) will introduce errors
- Can we perform inference for the full state $X_{0:t}$ given all observations $Y_{0:t}$ and actions $U_{0:t}$ so far?

$$p\left(X_{0:t} \middle| U_{0:t}, Y_{0:t}\right)$$

full state posterior



Full State Posterior Factorization

- The full state posterior factorizes into a product of observation likelihoods, state-transition likelihoods and the initial state distribution

$$
\begin{aligned}
p\left(X_{0:t} \middle| U_{1:t}, Y_{0:t}\right) &= \frac{p\left(Y_t \middle| X_{0:t}, U_{1:t}, Y_{0:t-1}\right) p\left(X_{0:t} \middle| U_{1:t}, Y_{0:t-1}\right)}{p\left(Y_t \middle| U_{1:t}, Y_{0:t}\right)} \\
&= \frac{p\left(Y_t \middle| X_t\right) p\left(X_t \middle| X_{0:t-1}, U_{1:t}, Y_{0:t-1}\right) p\left(X_{0:t-1} \middle| U_{1:t}, Y_{0:t-1}\right)}{p\left(Y_t \middle| U_{1:t}, Y_{0:t}\right)} \\
&= \eta_t \, p\left(Y_t \middle| X_t\right) p\left(X_t \middle| X_{t-1}, U_t\right) p\left(X_{0:t-1} \middle| U_{1:t-1}, Y_{0:t-1}\right) \\
&= p\left(X_0\right) \left(\prod_{\tau=0}^{t} \eta_\tau \, p\left(Y_\tau \middle| X_\tau\right) \right) \left(\prod_{\tau=1}^{t} p\left(X_\tau \middle| X_{\tau-1}, U_\tau\right) \right)
\end{aligned}
$$

Full State Posterior – Non-Linear Gaussian Case

$$p\left(X_{0:t} \middle| U_{1:t}, Y_{0:t}\right) = p\left(X_0\right)\left(\prod_{\tau=0}^{t} \eta_\tau \, p\left(Y_\tau \middle| X_\tau\right)\right)\left(\prod_{\tau=1}^{t} p\left(X_\tau \middle| X_{\tau-1}, U_\tau\right)\right)$$

- Non-linear state-transition model with Gaussian noise:

$$\mathbf{x}_t = g(\mathbf{x}_{t-1}, \mathbf{u}_t) + \boldsymbol{\epsilon}_t$$

$$\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{d_t})$$

$$p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t) = \mathcal{N}(\mathbf{x}_t; g(\mathbf{x}_{t-1}, \mathbf{u}_t), \boldsymbol{\Sigma}_{d_t})$$

- Non-linear observation model with Gaussian noise:

$$\mathbf{y}_t = h(\mathbf{x}_t) + \boldsymbol{\delta}_t$$

$$\boldsymbol{\delta} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{m_t})$$

$$p(\mathbf{y}_t \mid \mathbf{x}_t) = \mathcal{N}(\mathbf{y}_t; h(\mathbf{x}_t), \boldsymbol{\Sigma}_{m_t})$$

- Gaussian initial state estimate:

$$\mathbf{x}_0 \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0) \qquad p(\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_0; \boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$$

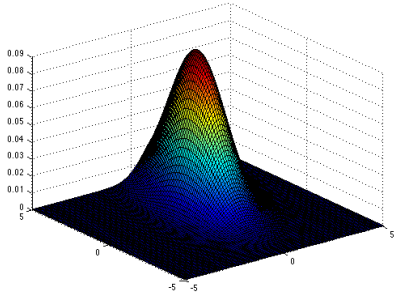
Full State Posterior – Non-Linear Gaussian Case

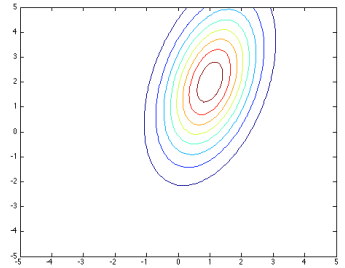
- We obtain the following factorization into normal distributions

$$p(\mathbf{x}_{0:t} \mid \mathbf{y}_{0:t}, \mathbf{u}_{1:t}) =$$
$$\mathcal{N}(\mathbf{x}_0; \boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0) \prod_{\tau=0}^{t} \eta_\tau \mathcal{N}(\mathbf{y}_\tau; h(\mathbf{x}_\tau), \boldsymbol{\Sigma}_{m_\tau}) \prod_{\tau=1}^{t} \mathcal{N}(\mathbf{x}_\tau; g(\mathbf{x}_{\tau-1}, \mathbf{u}_\tau), \boldsymbol{\Sigma}_{d_\tau})$$

- Recap: multivariate normal distribution

$$\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \left((2\pi)^d \det(\boldsymbol{\Sigma})\right)^{-1} \exp\left(-\tfrac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$
$$d = \dim(\mathbf{x})$$





- How can we find the maximum a-posteriori estimate for $\mathbf{x}_{0:t}$?

Negative Log-Posterior

- Since the logarithm is a monotonic increasing function, we can minimize the negative log-posterior probability instead

$$E(\mathbf{x}_{0:t}) = \text{const.} + \tfrac{1}{2}(\mathbf{x}_0 - \boldsymbol{\mu}_0)^\top \boldsymbol{\Sigma}_0^{-1}(\mathbf{x}_0 - \boldsymbol{\mu}_0)$$
$$+ \tfrac{1}{2}\sum_{\tau=0}^{t}(\mathbf{y}_\tau - h(\mathbf{x}_\tau))^\top \boldsymbol{\Sigma}_{m_\tau}^{-1}(\mathbf{y}_\tau - h(\mathbf{x}_\tau))$$
$$+ \tfrac{1}{2}\sum_{\tau=1}^{t}(\mathbf{x}_\tau - g(\mathbf{x}_{\tau-1}, \mathbf{u}_\tau))^\top \boldsymbol{\Sigma}_{d_\tau}^{-1}(\mathbf{x}_\tau - g(\mathbf{x}_{\tau-1}, \mathbf{u}_\tau))$$

  - The exponential functions vanish
  - The normalization factors are independent of the state variables and can be subsumed in a constant
  - Constants do not contribute to the minimization problem
  - Quadratic function in non-linear residuals on $\mathbf{x}_{0:t}$
- Non-linear least squares problem!

Non-Linear Least Squares

- We can rewrite the negative log-posterior as a non-linear least squares problem:

$$\arg\min_{\mathbf{x}} E(\mathbf{x}) = \frac{1}{2}\mathbf{r}(\mathbf{x})^\top \mathbf{W}\mathbf{r}(\mathbf{x})$$

  - Stack residuals in residual vector $\mathbf{r}(\mathbf{x})$
  - Inverse covariances in block-diagonal weight matrix $\mathbf{W}$

- Optimization approaches:
  - Gradient descent
  - Gauss-Newton
  - Levenberg-Marquardt
  - etc.

Gradient Descent

- Idea 1: Perform gradient descent to minimize E(x)

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \eta \nabla E(\mathbf{x}_k)$$

$$\nabla E(\mathbf{x}) = \nabla \mathbf{r}(\mathbf{x})^\top \mathbf{W} \mathbf{r}(\mathbf{x})$$

- Pros:
  - Stable convergence for sufficiently small step size
- Cons:
  - Slow convergence (linear convergence rate)
  - Solution quality depends on initial guess

Gauss-Newton Method

- Idea 2: Approximate Newton's method to minimize E(x)
  - Approximate E(x) through linearization of residuals

$$\widetilde{E}(\mathbf{x}) = \frac{1}{2}\widetilde{\mathbf{r}}(\mathbf{x})^\top \mathbf{W}\widetilde{\mathbf{r}}(\mathbf{x})$$

$$= \frac{1}{2}\left(\mathbf{r}(\mathbf{x}_k) + \mathbf{J}_k\left(\mathbf{x}-\mathbf{x}_k\right)\right)^\top \mathbf{W}\left(\mathbf{r}(\mathbf{x}_k) + \mathbf{J}_k\left(\mathbf{x}-\mathbf{x}_k\right)\right) \qquad \mathbf{J}_k := \nabla_{\mathbf{x}}\mathbf{r}(\mathbf{x})\big|_{\mathbf{x}=\mathbf{x}_k}$$

$$= \frac{1}{2}\mathbf{r}(\mathbf{x}_k)^\top \mathbf{W}\mathbf{r}(\mathbf{x}_k) + \underbrace{\mathbf{r}(\mathbf{x}_k)^\top \mathbf{W}\mathbf{J}_k}_{=:\mathbf{b}_k^\top}\left(\mathbf{x}-\mathbf{x}_k\right) + \frac{1}{2}\left(\mathbf{x}-\mathbf{x}_k\right)^\top \underbrace{\mathbf{J}_k^\top \mathbf{W}\mathbf{J}_k}_{=:\mathbf{H}_k}\left(\mathbf{x}-\mathbf{x}_k\right)$$

  - Find root of $\nabla_{\mathbf{x}}\widetilde{E}(\mathbf{x}) = \mathbf{b}_k^\top + \left(\mathbf{x}-\mathbf{x}_k\right)^\top \mathbf{H}_k$ using Newton's method, i.e.

$$\nabla_{\mathbf{x}}\widetilde{E}(\mathbf{x}) = \mathbf{0} \text{ iff } \mathbf{x} = \mathbf{x}_k - \mathbf{H}_k^{-1}\mathbf{b}_k$$

- Pros:
  - Faster convergence (approx. quadratic convergence rate)
- Cons:
  - Divergence if too far from local optimum (H not positive definite)
  - Solution quality depends on initial guess

Levenberg-Marquardt Method

- Idea 3: Gradually switch between gradient descent and Gauss-Newton
  - Augment Hessian approximation of Gauss-Newton (damping)

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \left(\mathbf{H}_k + \lambda \mathbf{I}\right)^{-1} \mathbf{b}_k$$

  - Adaptive weighting: $\mathbf{x}_{k+1} = \mathbf{x}_k - \left(\mathbf{H}_k + \lambda \operatorname{diag}(\mathbf{H}_k)\right)^{-1} \mathbf{b}_k$
  - Start with $\lambda = 0.1$
  - Accept step and decrease lambda $\lambda \leftarrow \lambda/2$ if error function decreases, otherwise discard step and increase lambda $\lambda \leftarrow 2\lambda$ (akin line search)

- Pros:
  - Fast convergence close to local optimum (quadratic convergence rate close to optimum)
  - More stable but slow convergence far from local optimum
- Cons:
  - Solution quality depends on initial guess

Iteratively Reweighted Least Squares

- Adaptive weights allow for approximately optimizing other norms on the residual function

$$E(\mathbf{x}) = \frac{1}{2} \sum_{i=1}^{N} w_i \left\| r_i(\mathbf{x}) \right\|_2^2$$

- Example: for any l_p norm, we need to set $w_{i,k} = \left| r_i(\mathbf{x}_k) \right|^{p-2}$

Why Filter?

- Runtime depends polynomially on time (inversion of the Hessian)
- Estimating the full state posterior can become prohibitively slow
- Typical current approaches bound the runtime by approximations:
  - Selection and optimization of keyframes to limit the optimization window size (subsampling)
  - Marginalization of old states to keep number of optimized frames constant
  - In fact, marginalization of all old states but the current one corresponds to Kalman Filtering in the Gaussian noise case
- We will see approximations to full posterior optimization later in more detail in concrete examples

Tools and Frameworks

- C++
  - ceres (http://ceres-solver.org/)
  - g2o (https://github.com/RainerKuemmerle/g2o)
- Matlab
  - Optimization toolbox (e.g. lsqnonlin)
- Python
  - lmfit
  - scipy.optimize.curve_fit

Lessons Learned Today

- Directed graphical models and factor graphs describe stochastic conditional independence relations of joint probability distributions
- Full state posterior factorizes into conditional observation and state-transition likelihoods per time step
- Negative log-posterior leads to non-linear least squares in the case of Gaussian noise
- Levenberg-Marquardt method for robust pseudo second-order optimization of non-linear least squares problems
- Runtime of full posterior optimization grows polynomially with time
- Approximations will be needed

Thanks for your attention!