DeepRob

Seminar 6
Visual Odometry and Localization
University of Michigan and University of Minnesota

This Week: Object Tracking

- Seminar 5: Recurrent Networks and Object Tracking
  1. DeepIM: Deep Iterative Matching for 6D Pose Estimation, Li et al., 2018
  2. PoseRBPF: A Rao-Blackwellized Particle Filter for 6D Object Pose Tracking, Deng et al., 2019
  3. 6-PACK: Category-level 6D Pose Tracker with Anchor-Based Keypoints, Wang et al., 2020
  4. XMem: Long-Term Video Object Segmentation with an Atkinson-Shiffrin Memory Model, Cheng and Schwing, 2022

- Seminar 6: Visual Odometry and Localization
  1. Backprop KF: Learning Discriminative Deterministic State Estimators, Haarnoja et al., 2016
  2. Differentiable Particle Filters: End-to-End Learning with Algorithmic Priors, Jonschkowski et al., 2018
  3. Multimodal Sensor Fusion with Differentiable Filters, Lee et al., 2020
  4. Differentiable SLAM-net: Learning Particle SLAM for Visual Navigation, Karkus et al., 2021

Today: Visual Odometry and Localization

- Seminar 5: Recurrent Networks and Object Tracking
  1. DeepIM: Deep Iterative Matching for 6D Pose Estimation, Li et al., 2018
  2. PoseRBPF: A Rao-Blackwellized Particle Filter for 6D Object Pose Tracking, Deng et al., 2019
  3. 6-PACK: Category-level 6D Pose Tracker with Anchor-Based Keypoints, Wang et al., 2020
  4. XMem: Long-Term Video Object Segmentation with an Atkinson-Shiffrin Memory Model, Cheng and Schwing, 2022

- Seminar 6: Visual Odometry and Localization
  1. Backprop KF: Learning Discriminative Deterministic State Estimators, Haarnoja et al., 2016
  2. Differentiable Particle Filters: End-to-End Learning with Algorithmic Priors, Jonschkowski et al., 2018
  3. Multimodal Sensor Fusion with Differentiable Filters, Lee et al., 2020
  4. Differentiable SLAM-net: Learning Particle SLAM for Visual Navigation, Karkus et al., 2021

BackpropKF

Learning Discriminative Deterministic State Estimators

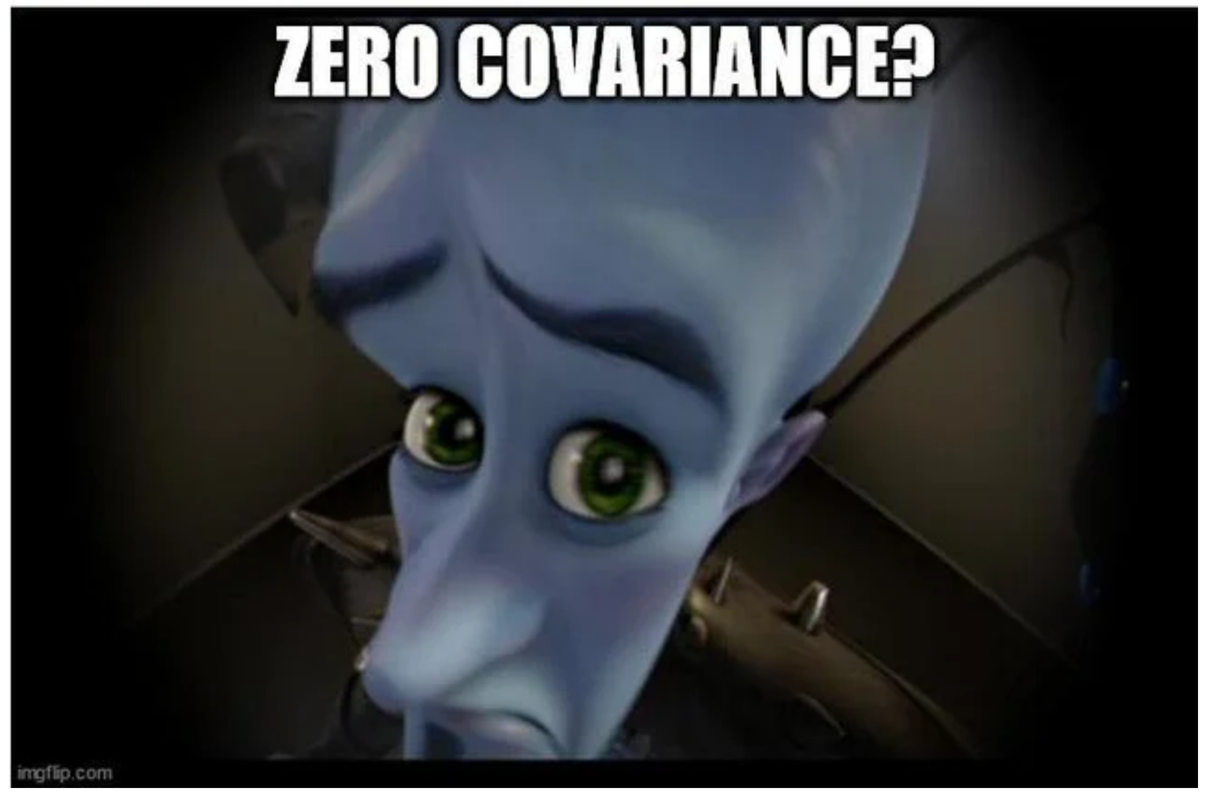
By: Tuomas Haarnoja, Anurag Ajay, Sergey Levine, Pieter Abbeel

Presented by: Sarvesh Mayilvahanan, Hersh Vakharia

Kalman Filters cut through the noise

https://cstwiki.wtb.tue.nl/wiki/Embedded_Motion_Control_2015_Group_4

POV: Your Kalman filter isn't processing measurements



Kalman Filters cut through the noise

https://cstwiki.wtb.tue.nl/wiki/Embedded_Motion_Control_2015_Group_4

The Authors

- Tuomas Haarnoja
  - Research Scientist at DeepMind
  - PhD from UC Berkeley
- Anurag Ajay
  - PhD student at MIT, BS from UC Berkeley
- Sergey Levine
  - Associate Professor at UC Berkeley
- Pieter Abbeel
  - Professor at UC Berkeley

Motivation

- Kalman Filters are unable to handle complex sensory input (images)

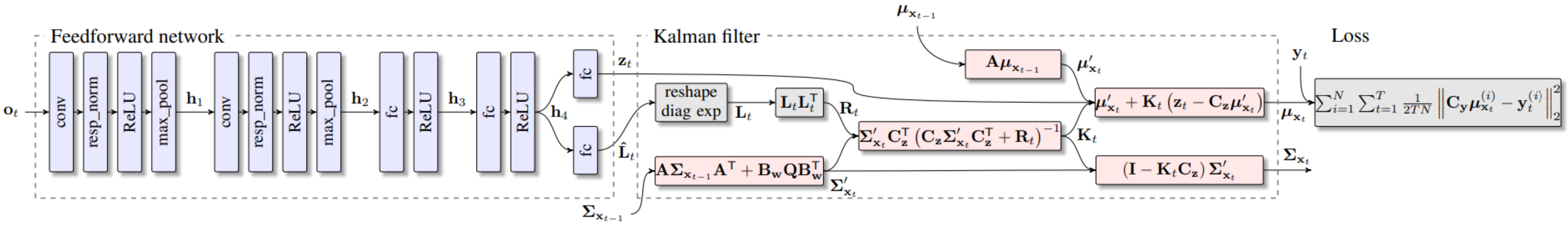
- Backprop KF allows Kalman Filters to use rich sensory input

- Computational graph structure allows for backprop through the Kalman Filter

- Allows for more robust state estimation

Contributions

1. Formulation of Kalman Filter as Computational Graph
2. Representation of Uncertainty in Latent Observations
3. Novel Response Normalization

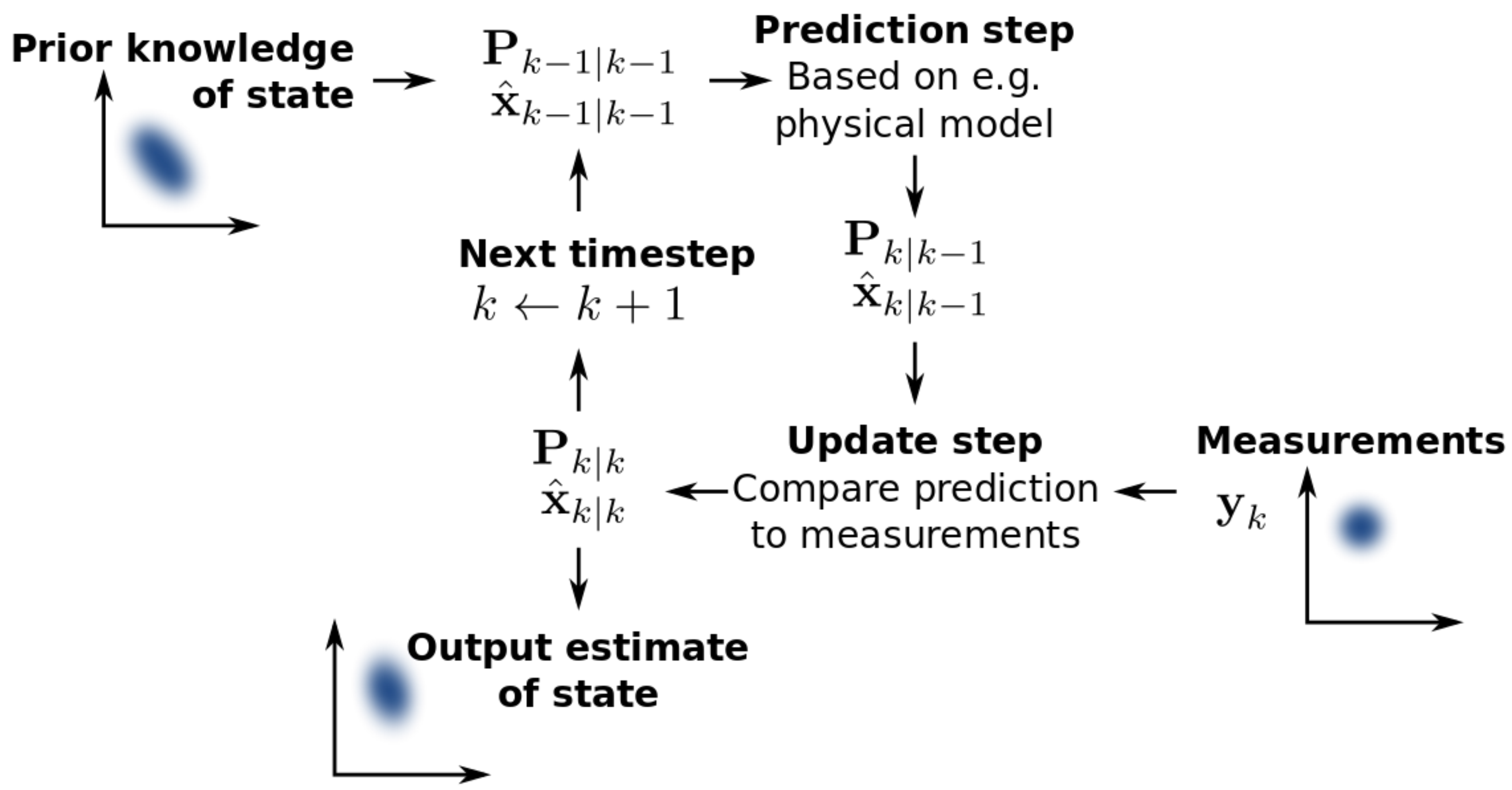


Background

- Generative models (KF) cannot use rich sensory input
- Discriminative models (RNNs) do not use knowledge of dynamics
- Generative and Discriminative models have been applied separately but haven’t been successfully combined for state estimation

What are Kalman Filters?

- **Prior**: current understanding of state
- **Prediction**: Propagate prior using dynamics
- **Update**: Correct prediction using observations



https://en.wikipedia.org/wiki/Kalman_filter

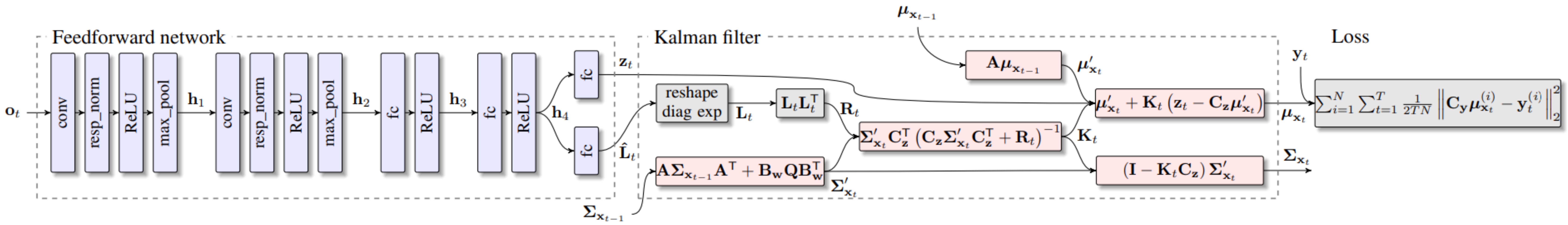
Approach

- 3 main components
  - Feedforward Network
  - Kalman Filter
  - Loss Function

Feedforward Network

- Takes in high-dimensional observation $\mathbf{o}_t$
- Outputs condensed representation $\mathbf{z}_t$ in latent space
  - Also provides uncertainty in observation $\hat{\mathbf{L}}_t$

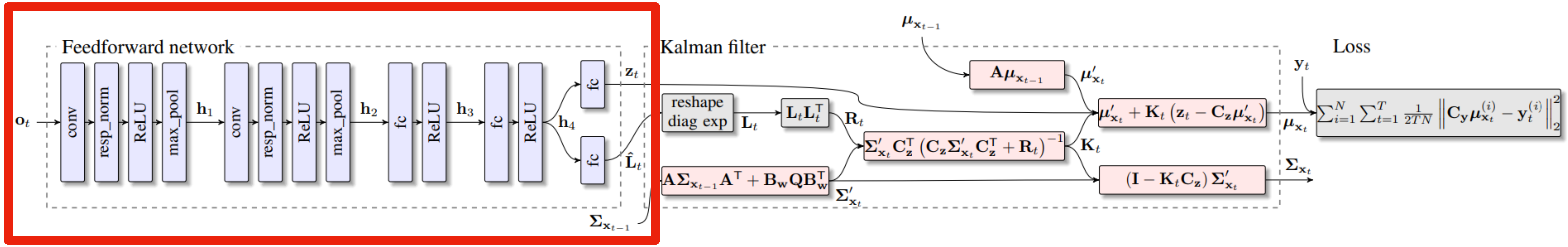
- Novel normalization layer: learnable mean, stdv



Feedforward Network

- Takes in high-dimensional observation $\mathbf{o}_t$
- Outputs condensed representation $\mathbf{z}_t$ in latent space
  - Also provides uncertainty in observation $\hat{\mathbf{L}}_t$

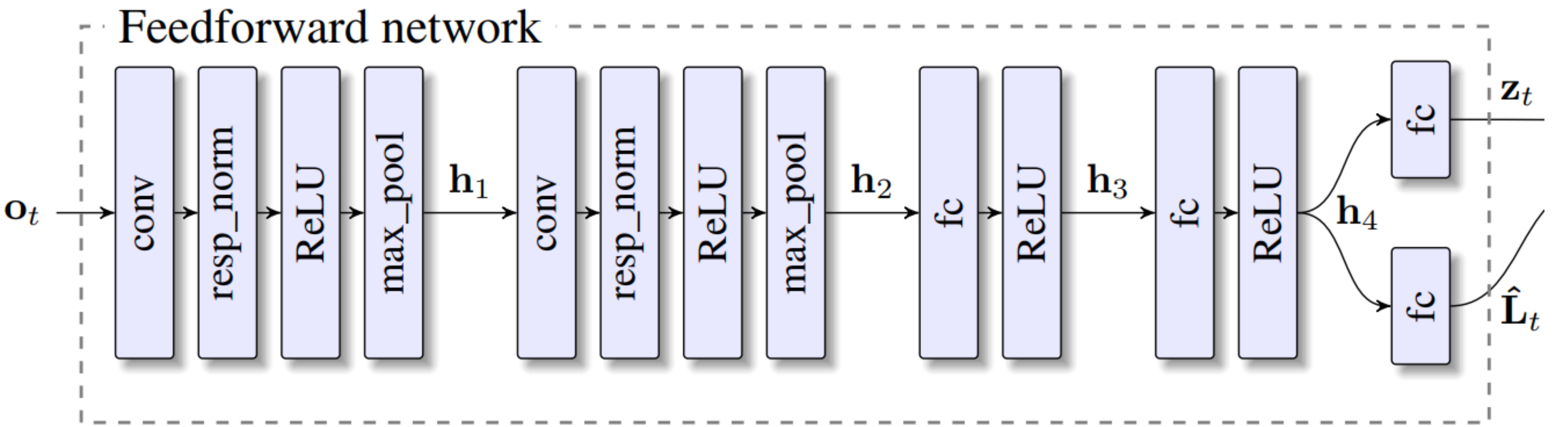
- Novel normalization layer: learnable mean, stdv



Feedforward Network

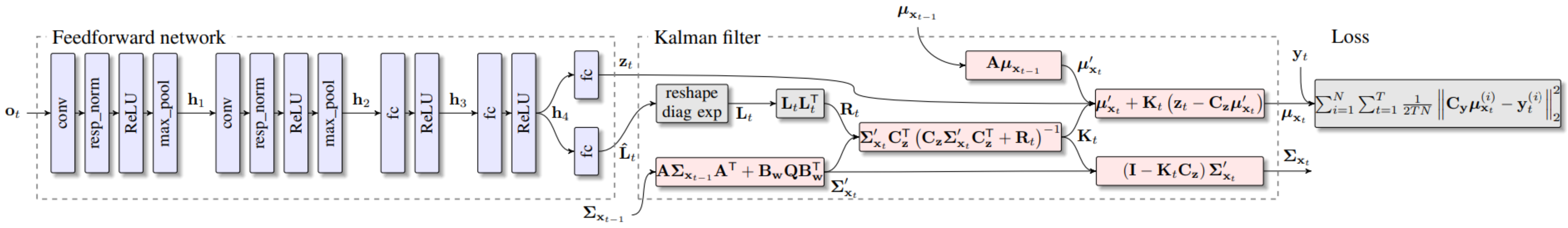
- Takes in high-dimensional observation $\mathbf{o}_t$
- Outputs condensed representation $\mathbf{z}_t$ in latent space
  - Also provides uncertainty in observation $\hat{\mathbf{L}}_t$

- Novel normalization layer: learnable mean, stdv



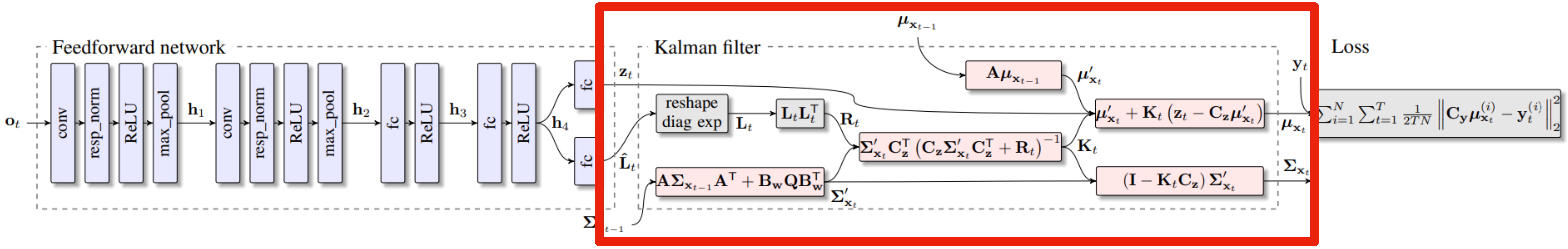
Kalman Filter

- Prediction step: $\boldsymbol{\mu}'_{\mathbf{x}_t}$ $\boldsymbol{\Sigma}'_{\mathbf{x}_t}$
  - Combine latent obs. uncertainty $\hat{\mathbf{L}}_t$ and prev. state uncertainty $\boldsymbol{\Sigma}_{\mathbf{x}_{t-1}}$
- Update step: $\boldsymbol{\mu}_{\mathbf{x}_t}$ $\boldsymbol{\Sigma}_{\mathbf{x}_t}$
  - Use latent obs. $\mathbf{z}_t$ and Kalman gain $\mathbf{K}_t$



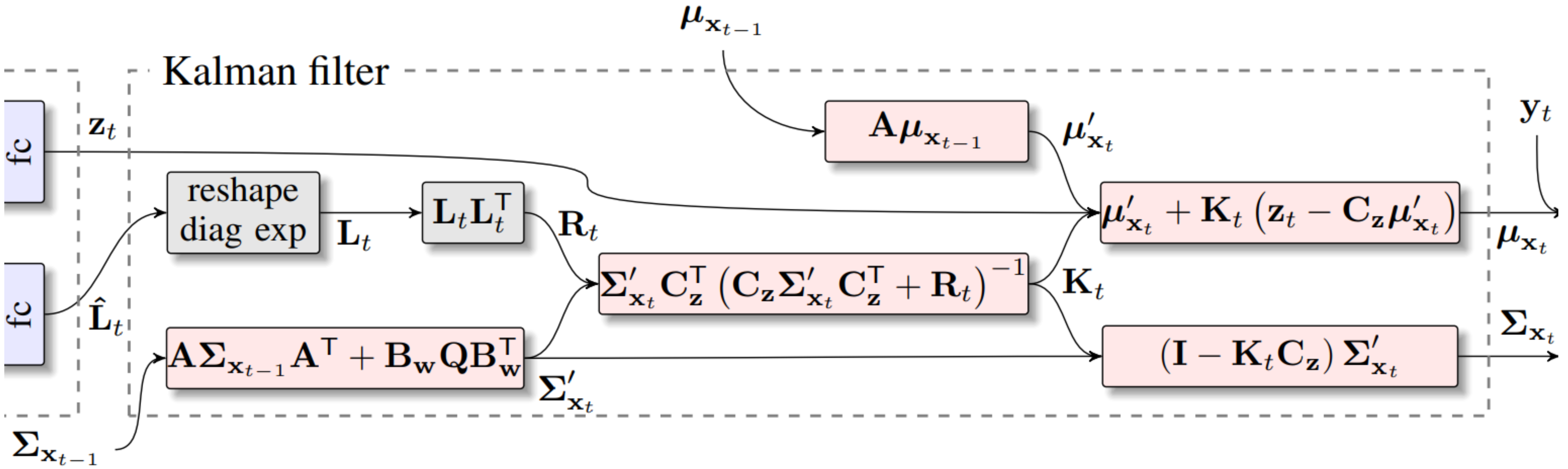
Kalman Filter

- Prediction step: $\boldsymbol{\mu}'_{\mathbf{x}_t}$ $\boldsymbol{\Sigma}'_{\mathbf{x}_t}$
  - Combine latent obs. uncertainty $\hat{\mathbf{L}}_t$ and prev. state uncertainty $\boldsymbol{\Sigma}_{\mathbf{x}_{t-1}}$
- Update step: $\boldsymbol{\mu}_{\mathbf{x}_t}$ $\boldsymbol{\Sigma}_{\mathbf{x}_t}$
  - Use latent obs. $\mathbf{z}_t$ and Kalman gain $\mathbf{K}_t$



Kalman Filter

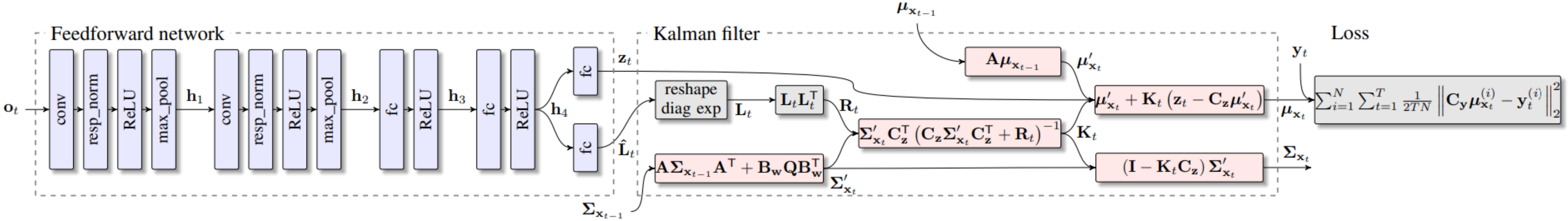
- Prediction step: $\boldsymbol{\mu}'_{\mathbf{x}_t}$ $\boldsymbol{\Sigma}'_{\mathbf{x}_t}$
  - Combine latent obs. uncertainty $\hat{\mathbf{L}}_t$ and prev. state uncertainty $\boldsymbol{\Sigma}_{\mathbf{x}_{t-1}}$
- Update step: $\boldsymbol{\mu}_{\mathbf{x}_t}$ $\boldsymbol{\Sigma}_{\mathbf{x}_t}$
  - Use latent obs. $\mathbf{z}_t$ and Kalman gain $\mathbf{K}_t$



Loss Function

- Compare noiseless obs $\mathbf{y}_t$ with predicted mea $\boldsymbol{\mu}_{\mathbf{x}_t}$ using observation r $\mathbf{C_y}$ del
  - Weighted L2 norm

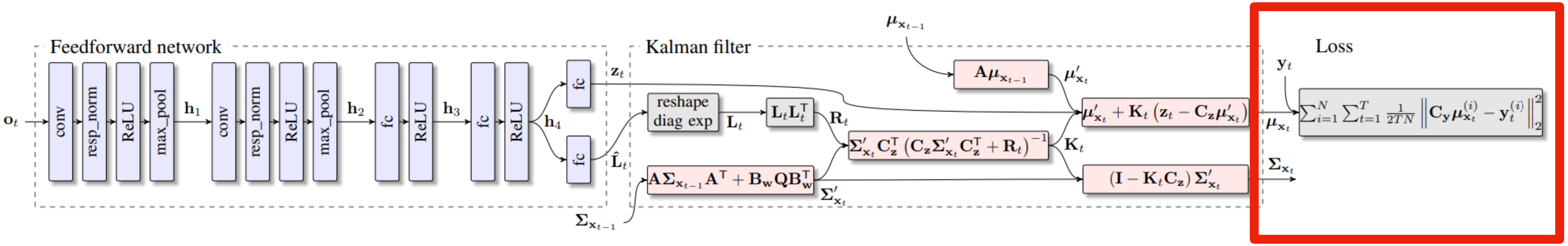
- Loss is backpropagated to update NN weights



Loss Function

- Compare noiseless obs $\mathbf{y}_t$ with predicted mea $\boldsymbol{\mu}_{\mathbf{x}_t}$ using observation r$\mathbf{C}_\mathbf{y}$del
  - Weighted L2 norm

- Loss is backpropagated to update NN weights



Loss Function

- Compare noiseless obs $\mathbf{y}_t$ with predicted mea $\boldsymbol{\mu}_{\mathbf{x}_t}$ using observation r $\mathbf{C}_\mathbf{y}$ del
  - Weighted L2 norm

- Loss is backpropagated to update NN weights

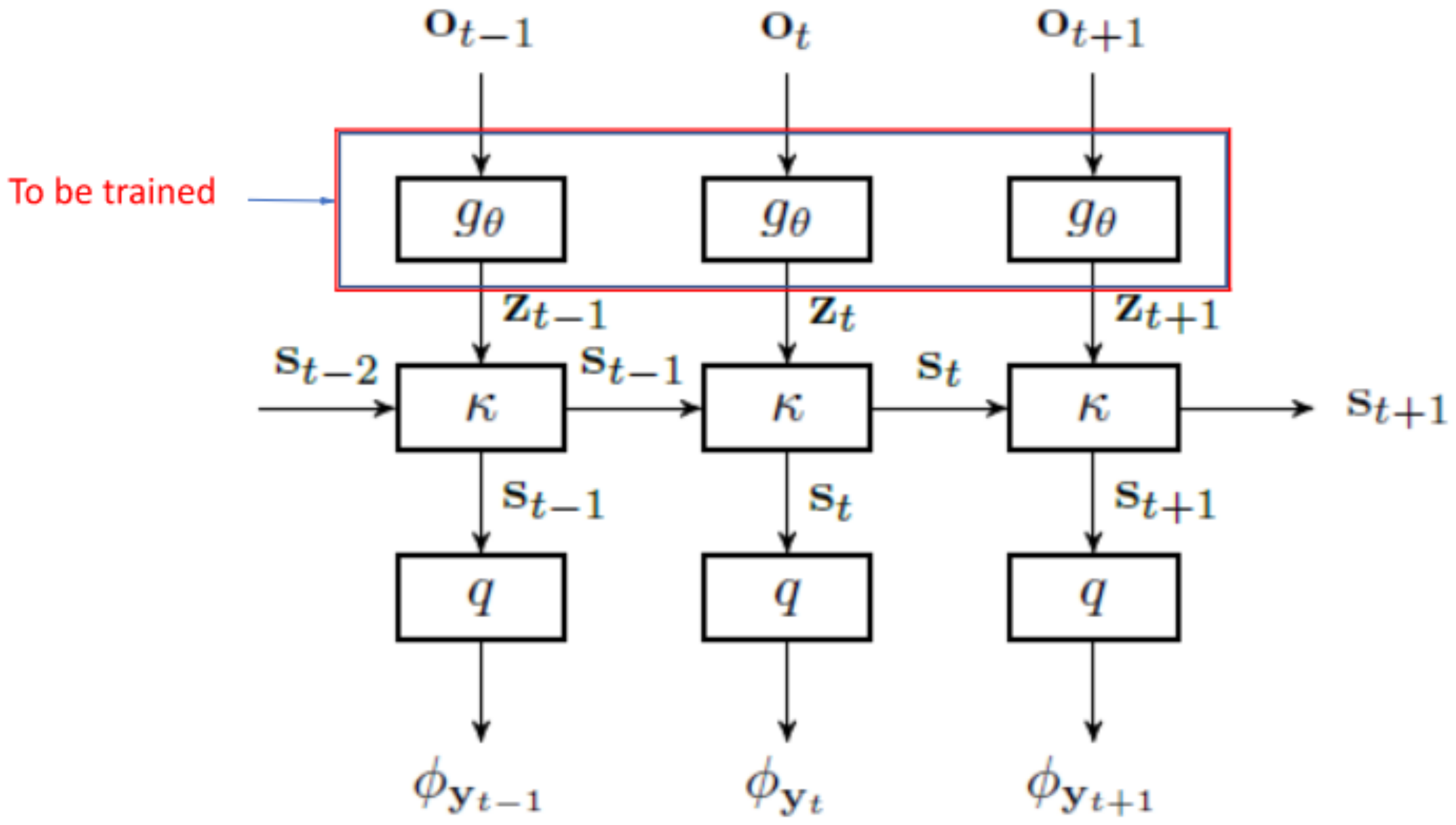
$\mathbf{y}_t$

$\boldsymbol{\mu}_{\mathbf{x}_t}$

$$\boxed{\sum_{i=1}^{N} \sum_{t=1}^{T} \frac{1}{2TN} \left\| \mathbf{C}_\mathbf{y} \boldsymbol{\mu}_{\mathbf{x}_t}^{(i)} - \mathbf{y}_t^{(i)} \right\|_2^2}$$

$\boldsymbol{\Sigma}_{\mathbf{x}_t}$

BKF Forward pass



O – high dimensional raw observation

g – discriminative model to predict low dimension observation **z**

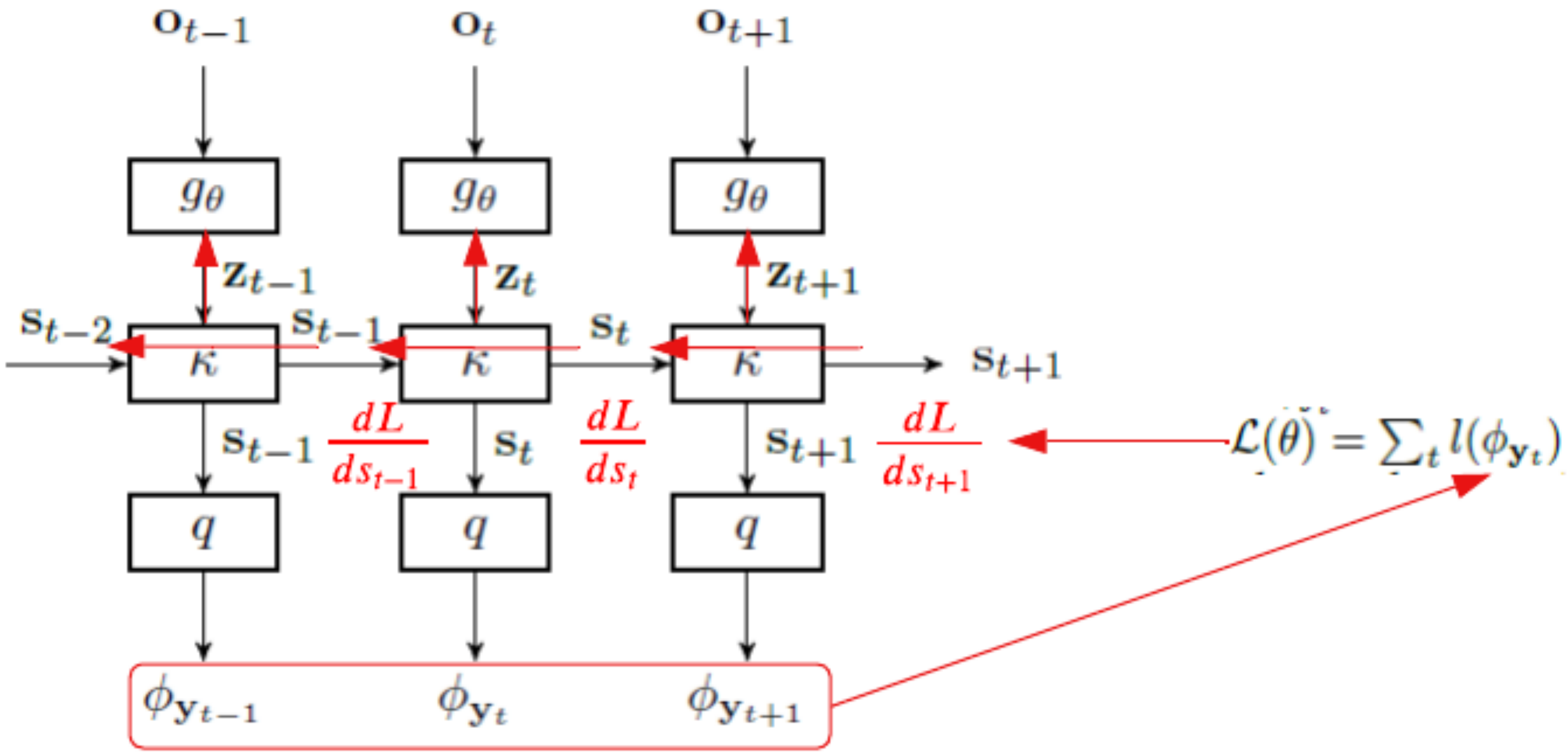
k – Kalman filter update

q – deterministic output function, such as

$q(\mathbf{s}_t) = (\mathbf{C}_\mathbf{y}\mu_{\mathbf{x}_t}, \mathbf{C}_\mathbf{y}\Sigma_{\mathbf{x}_t}\mathbf{C}_\mathbf{y}^\top)$

phi – output state

Backpropagation



O – high dimensional raw observation

g – discriminative model to predict low dimension observation **z**

k – Kalman filter update

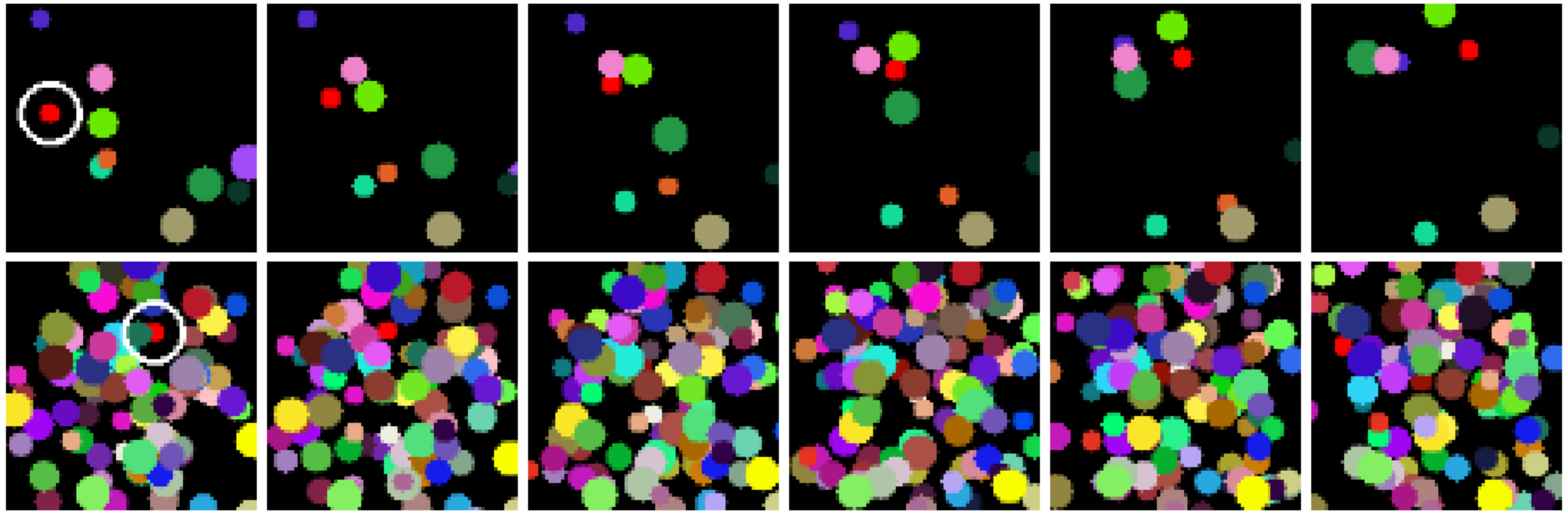
q – deterministic output function, such as

$q(\mathbf{s}_t) = (\mathbf{C}_\mathbf{y}\boldsymbol{\mu}_{\mathbf{x}_t}, \mathbf{C}_\mathbf{y}\boldsymbol{\Sigma}_{\mathbf{x}_t}\mathbf{C}_\mathbf{y}^\mathsf{T})$

phi – output state

$$\nabla_\theta \mathcal{L}(\theta) = \sum_{t=1}^{T} \frac{d\mathbf{z}_t}{d\theta} \frac{d\mathbf{s}_t}{d\mathbf{z}_t} \frac{d\mathcal{L}}{d\mathbf{s}_t}$$

Results: Long Term Tracking w/ Occlusion



State Estimation Model	# Parameters	RMS test error $\pm\sigma$
feedforward model	7394	0.2322 ± 0.1316
piecewise KF	7397	0.1160 ± 0.0330
LSTM model (64 units)	33506	0.1407 ± 0.1154
LSTM model (128 units)	92450	0.1423 ± 0.1352
BKF (ours)	**7493**	$\mathbf{0.0537} \pm \mathbf{0.1235}$

- Compared models
  - **Feedforward Model:** pure CNN
  - **Piecewise KF:** similar to BKF w/o uncertainty propagation, separate backpropagation
  - **LSTM:** replaces KF w/ RNN

Results: KITTI

	Test 100			Test 100/200/400/800		
# training trajectories	3	6	10	3	6	10
Translational Error [m/m]						
piecewise KF	0.3257	0.2452	0.2265	0.3277	0.2313	0.2197
LSTM model (128 units)	0.5022	0.3456	0.2769	0.5491	0.4732	0.4352
LSTM model (256 units)	0.5199	0.3172	0.2630	0.5439	0.4506	0.4228
BKF (ours)	**0.3089**	**0.2346**	**0.2062**	**0.2982**	**0.2031**	**0.1804**
Rotational Error [deg/m]						
piecewise KF	0.1408	0.1028	0.0978	0.1069	0.0768	0.0754
LSTM model (128 units)	0.5484	0.3681	0.3767	0.4123	0. 3573	0.3530
LSTM model (256 units)	0.4960	0.3391	0.2933	0.3845	0.3566	0.3221
BKF (ours)	**0.1207**	**0.0901**	**0.0801**	**0.0888**	**0.0587**	**0.0556**

- 11-fold cross validation
  - Randomly sampled subsequences (100/200/400/800) timesteps
  - Repeat experiment for subset of 10 folds

Conclusions

- Combination of Discriminative and Generative models outperforms models separately
  - Uses domain knowledge, compressed representation of complex observations
- End-to-end training of entire model improves performance over separate training (piecewise KF)

Directions for Future Work

- Future directions
  - Can be applied to other probabilistic/deterministic filters (UKF, Particle Filters, etc.)
  - Extend BKF using complex non-linear dynamics, larger latent space

Thank you

Differentiable SLAM-net

Learning Particle SLAM for Visual Navigation

By: Peter Karkus, Shaojun Cai, David Hsu

Presented by: Rahul Kashyap Swayampakula, Dhyey Manish Rajani, Surya Pratap Singh

The Authors

- Peter Karkus
  - Previously: PhD Candidate at National University of Singapore
- Shaojun Cai
  - 2nd year PhD Candidate at National University of Singapore
- David Hsu
  - Provost's Chair Professor, Department of Computer Science, National University of Singapore

Background

- Simultaneous localization and mapping (SLAM) remains challenging for numerous downstream applications, such as indoor visual robot navigation, because of its inability to handle rapid turns, featureless walls, and poor camera quality.

- Also, traditional SLAM algorithms require handcrafted features and manually-tuned parameters, limiting their applicability in complex environments.

Related Work

1. Learning based SLAM

Uses learning for (i) compact representation; (ii) CNN-based depth predictor; (iii) feature metric representation and Bundle Adjustment (BA).

2. Classic SLAM

Filtering-based approaches(maintain & sequentially update a probability distribution.)
Optimization-based approaches (apply BA on keyframes and local maps; popular for both visual and lidar-based SLAM.)

3. Differentiable Algorithms

Only differential approximation for state estimation , visual mapping , planning and control tasks .

4. Visual Navigation Algorithms

Existing approaches either assume a known location or a known map or rely solely on relative visual odometry.

Contributions

- The navigation architecture with SLAM-net improves the state-of-the-art for the Habitat Challenge 2020 PointNav task by a large margin (from 37% to 64% success)
- The authors conduct a comprehensive ablation study to analyze the contribution of each component of the proposed approach.
- The paper evaluates the proposed approach on various visual navigation tasks; method outperforms various baselines like FastSLAM, ORB-SLAM on RGB and RGB-D datasets.

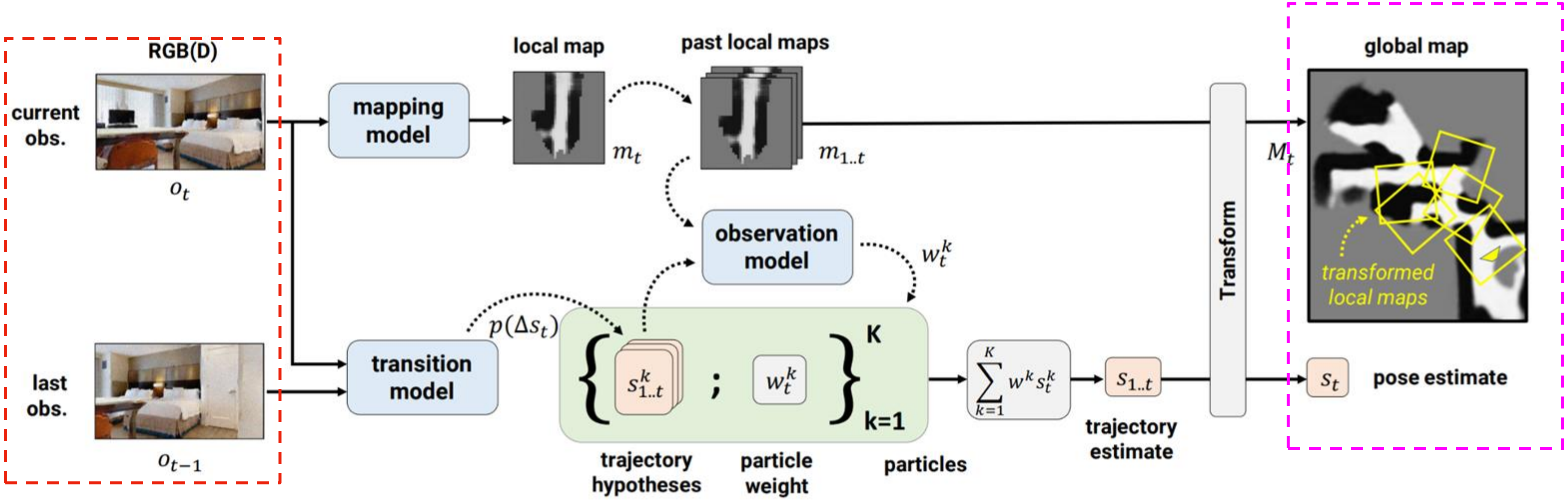
What are the benefits if we solve this problem?

Solving the problem addressed can provide benefits like:

1. Improved accuracy
2. Reduced complexity
3. Increased versatility
4. Faster development
5. Better feature extraction and association.
6. Better relocalization and loop closure.
7. Better SLAM integration into navigation pipeline

Approach

- Differentiable SLAM-net



Input: RGB-D (or RGB) images

- Pose at time t, $s_t = (x_t, y_t, \theta_t)$
- Trajectory till time t, $s_{1..t} = (x_{1..t}, y_{1..t}, \theta_{1..t})$

Output: 2D pose + global map

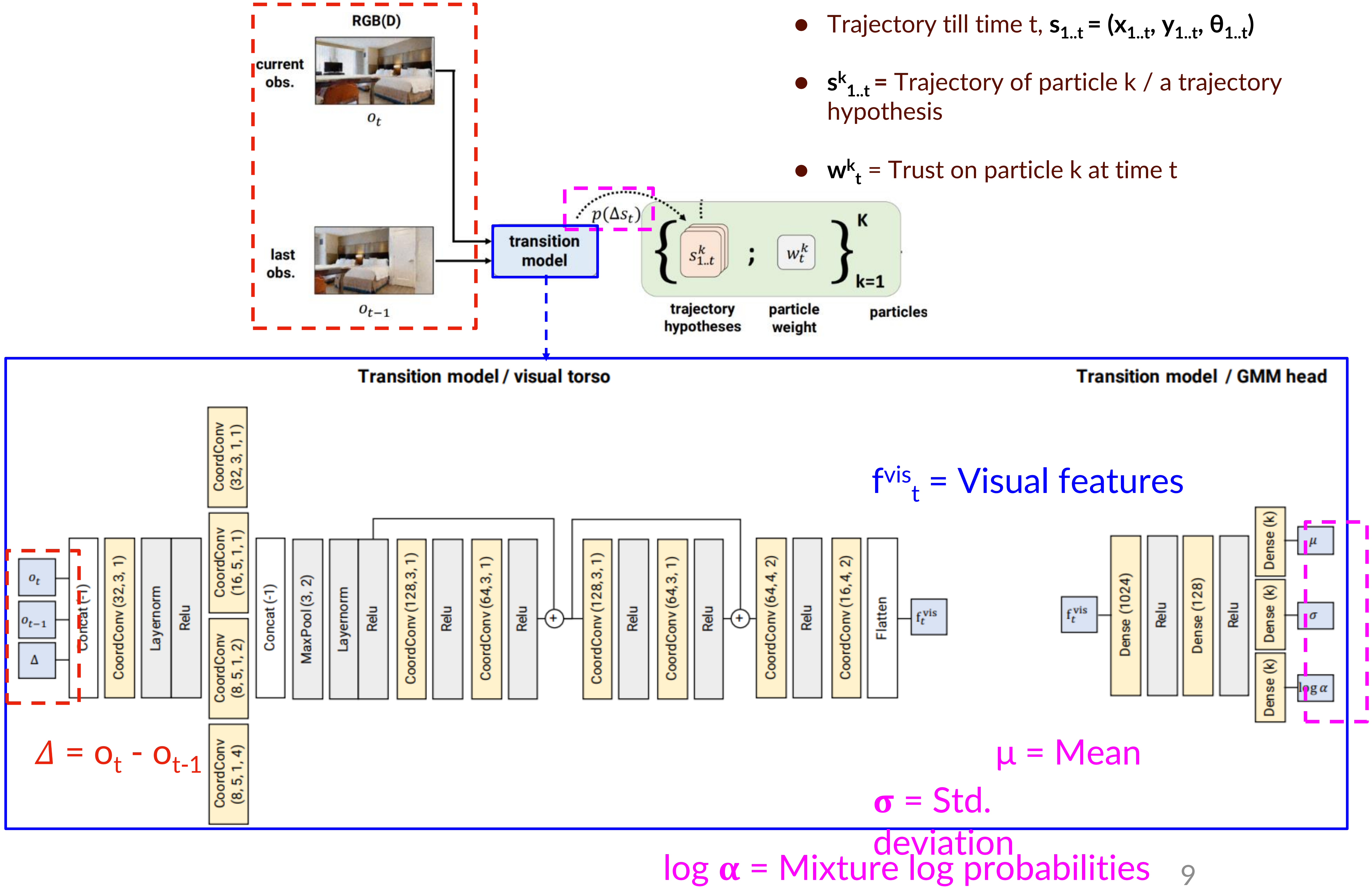
- $s^k_{1..t}$ = Trajectory of particle k / a trajectory hypothesis
- w^k_t = Trust on particle k at time t

Transition model

- **Input:** Current observation, $\mathbf{o}_t$, last observation, $\mathbf{o}_{t-1}$, & action, $\mathbf{a}_{t-1}$, if available
- **Output:** Probability distribution over 2D relative pose, $\mathbf{p}(\Delta \mathbf{s}_t)$
- **Serves 1 purpose:**
  - Generate trajectory hypotheses
- Transition model can be broken into a "Visual torso" and 3 GMM heads
- Visual torso works as a feature extractor
- GMM head generates the mean, std. deviation, and log-likelihood

$\mathbf{o}_t$: 160 x 90 x 4 image

- Pose at time t, $s_t = (x_t, y_t, \theta_t)$
- Trajectory till time t, $\mathbf{s}_{1..t} = (\mathbf{x}_{1..t}, \mathbf{y}_{1..t}, \theta_{1..t})$
- $s^k_{1..t}$ = Trajectory of particle k / a trajectory hypothesis
- w^k_t = Trust on particle k at time t

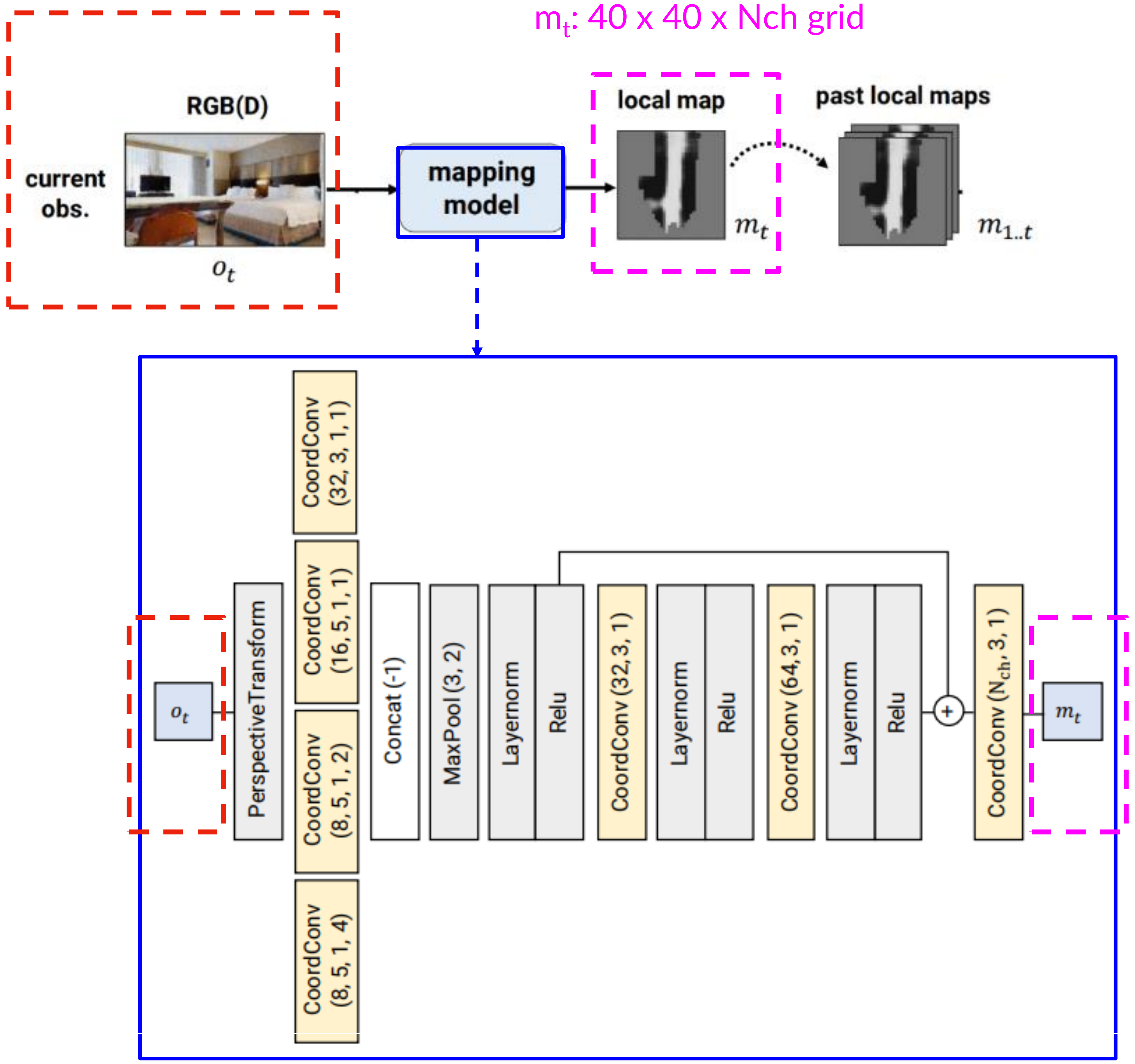


Mapping model

- **Input:** Current observation, $\mathbf{o_t}$
- **Output:** Local map, $\mathbf{m_t}$
- **Serves 2 purposes:**
  - Aid in loop closing for 2D pose estimation
  - Generate global map
- Perspective transform:
  - Converts observation to top-down view (160 x 160)
- Each cell in local map is 12cm x 12cm

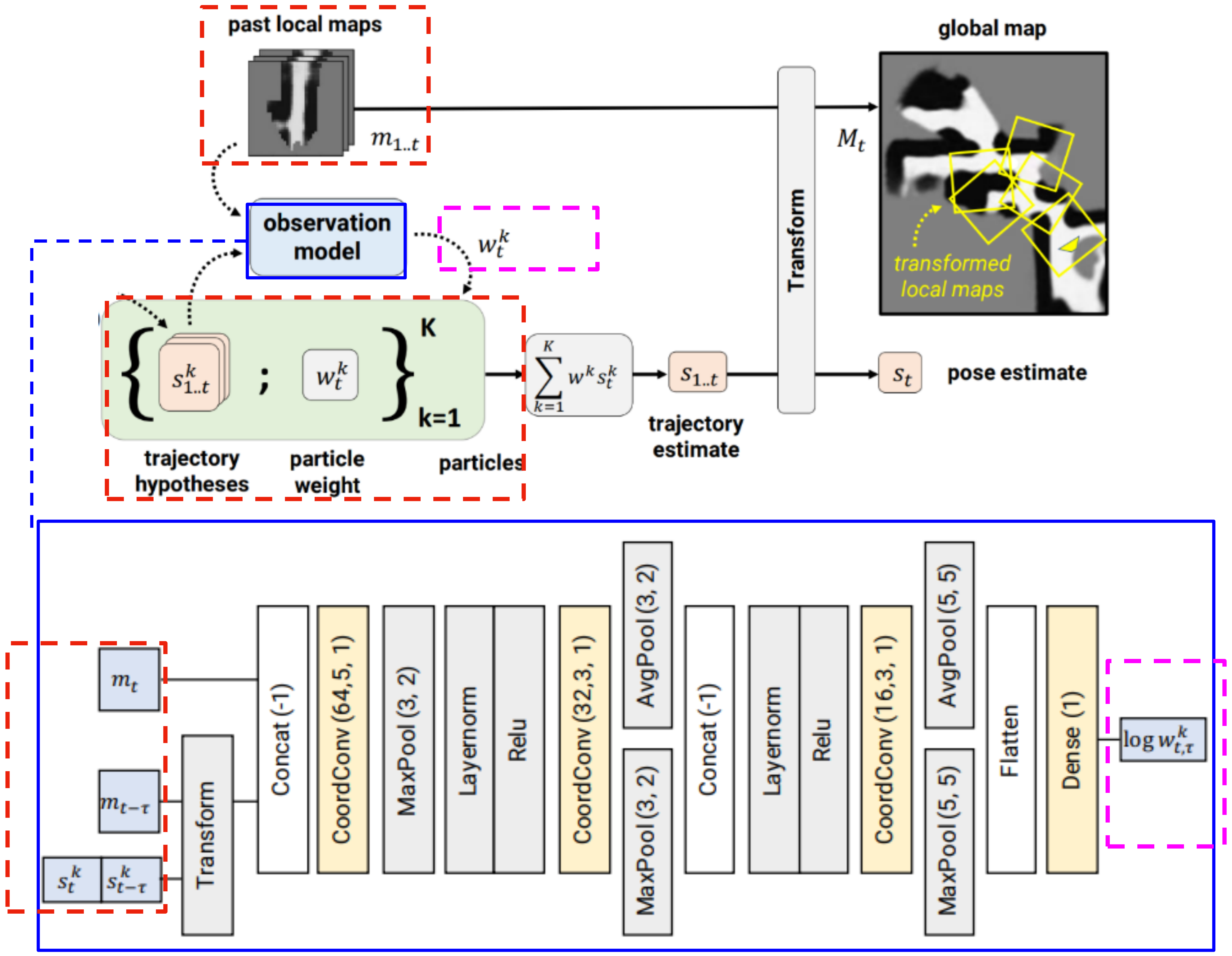
o_t: 160 x 90 x 4 image

m_t: 40 x 40 x Nch grid



Observation model

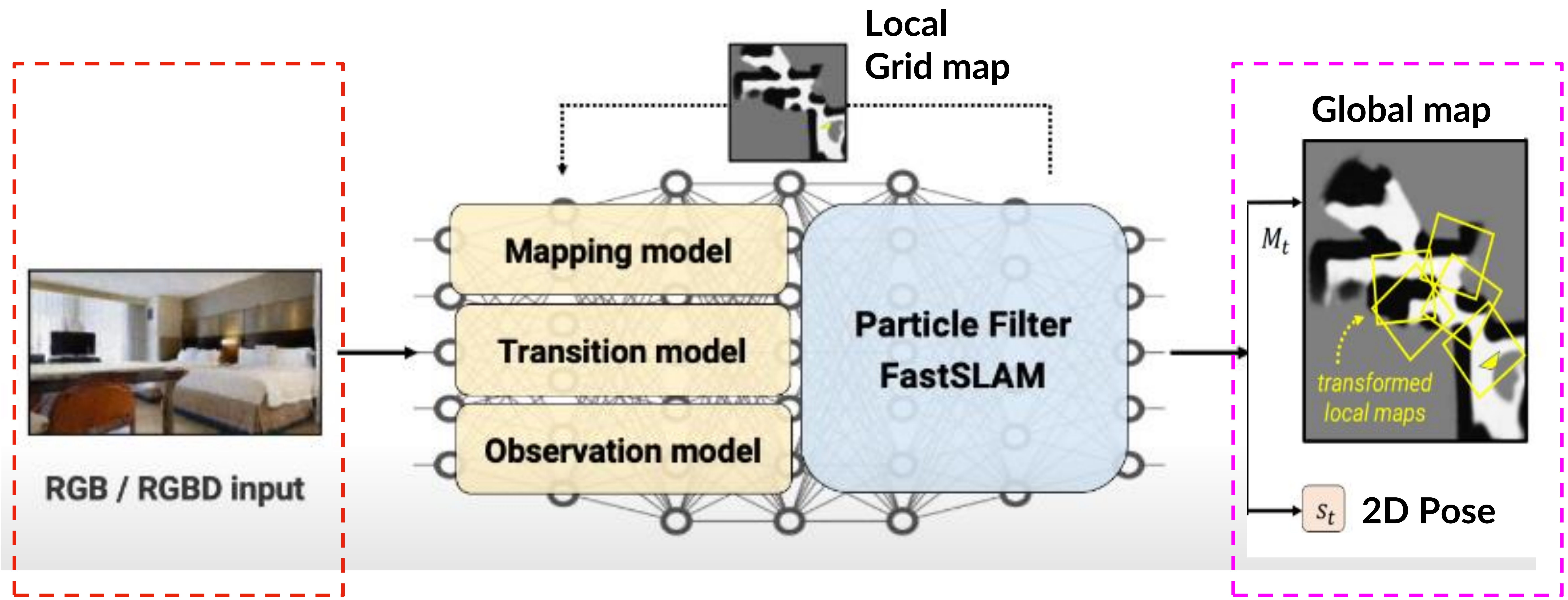
- **Input:** Local maps, $\mathbf{m}_t$, trajectory hypotheses, $\mathbf{s}^k_{1:t}$, with their particle weights, $\mathbf{w}^k_t$
- **Output:** Updated particle weights, $\mathbf{w}^k_t$
- **Serves 1 purposes:**
  - Aid in generating 2D pose
- Current local map gets compared with most recent local maps to output "compatibility"
- Trajectory estimate:
  - Weighted sum of particle poses
- Transform:
  - Rotation and translational image transformations to local maps given the relative poses
- Global map:
  - By transforming local maps along the trajectory estimate



s^k_t = Current state of particle k
$s^k_{t-\tau}$ = A past state of particle k

$\log w^k_{t,\tau}$ = log-likelihood of kth particle weight between time steps t and τ

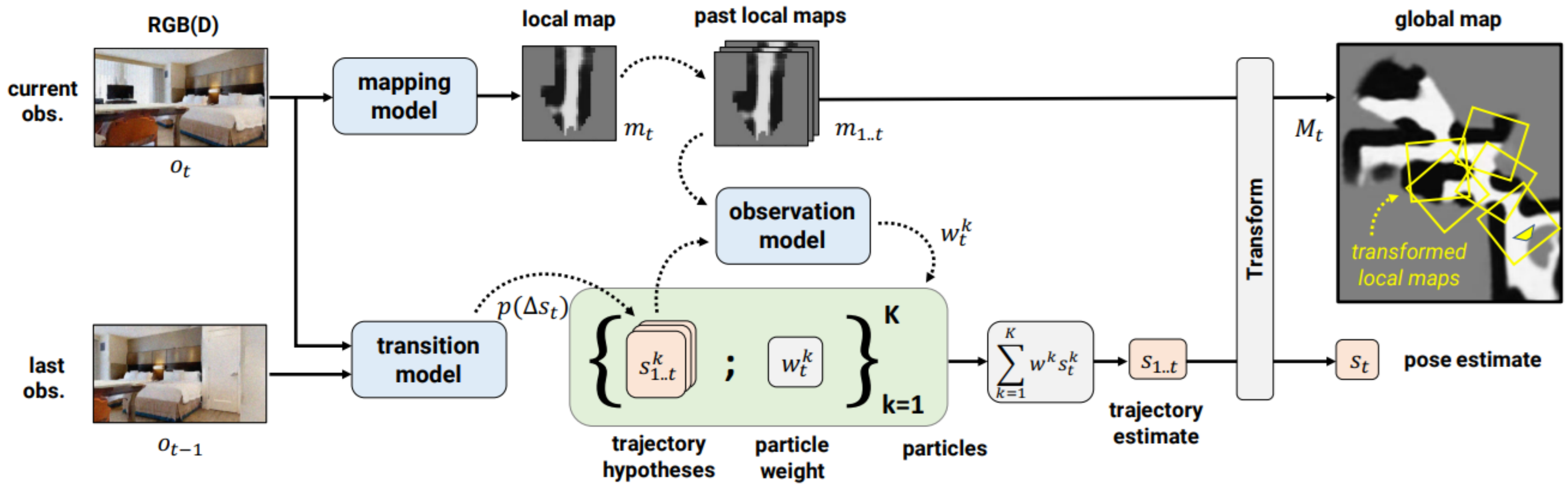
Differentiable SLAM-net Summarized



Input: RGB-D (or RGB) images

Output: 2D pose + global map

Training and inference

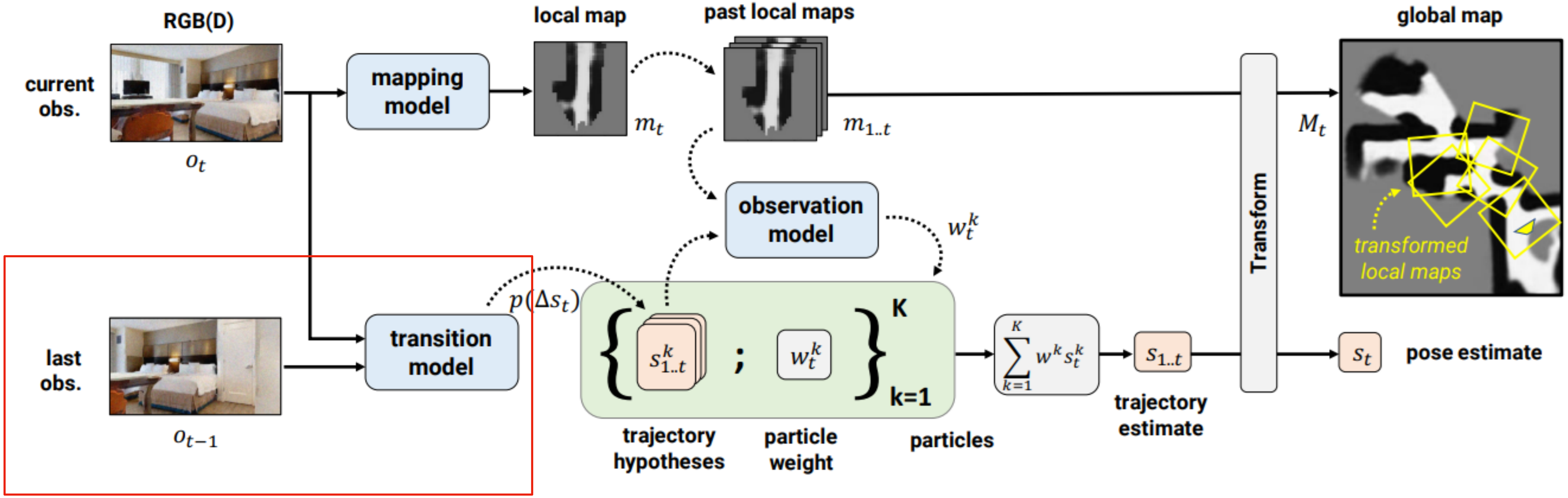


Training Data

- RGBD images
- 2D poses
- Ground truth global map

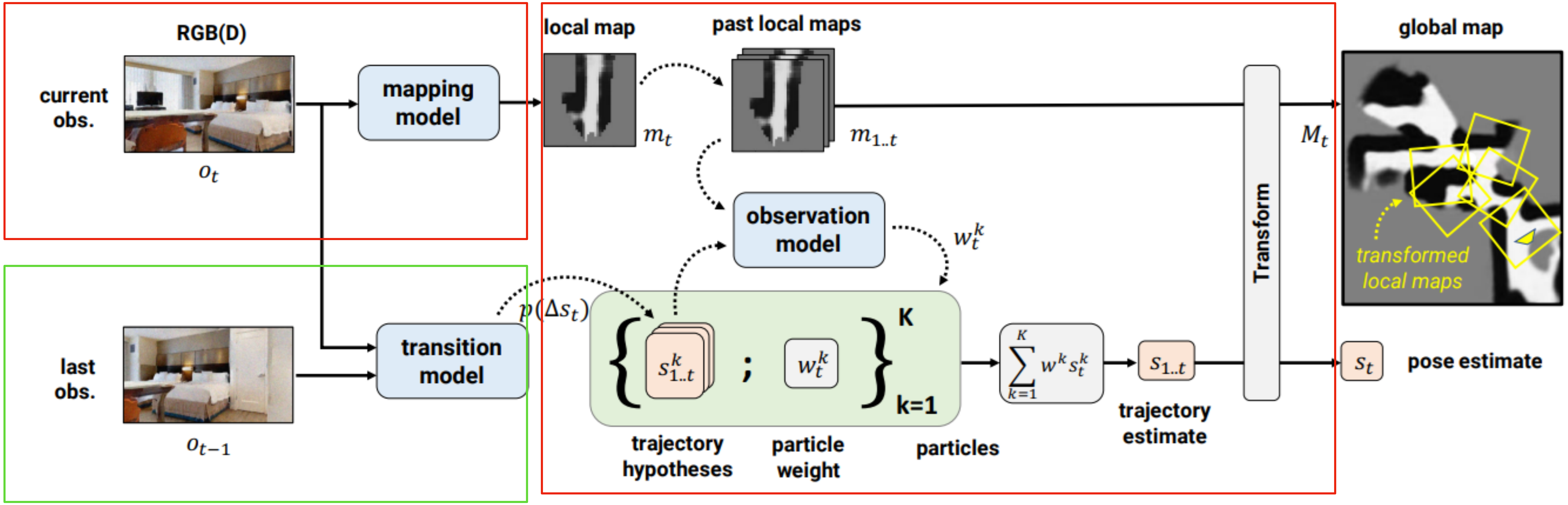
Objective: The end-to-end training objective is the sum of **Huber losses** for the *2D pose error and 1D orientation error*

Multi stage training



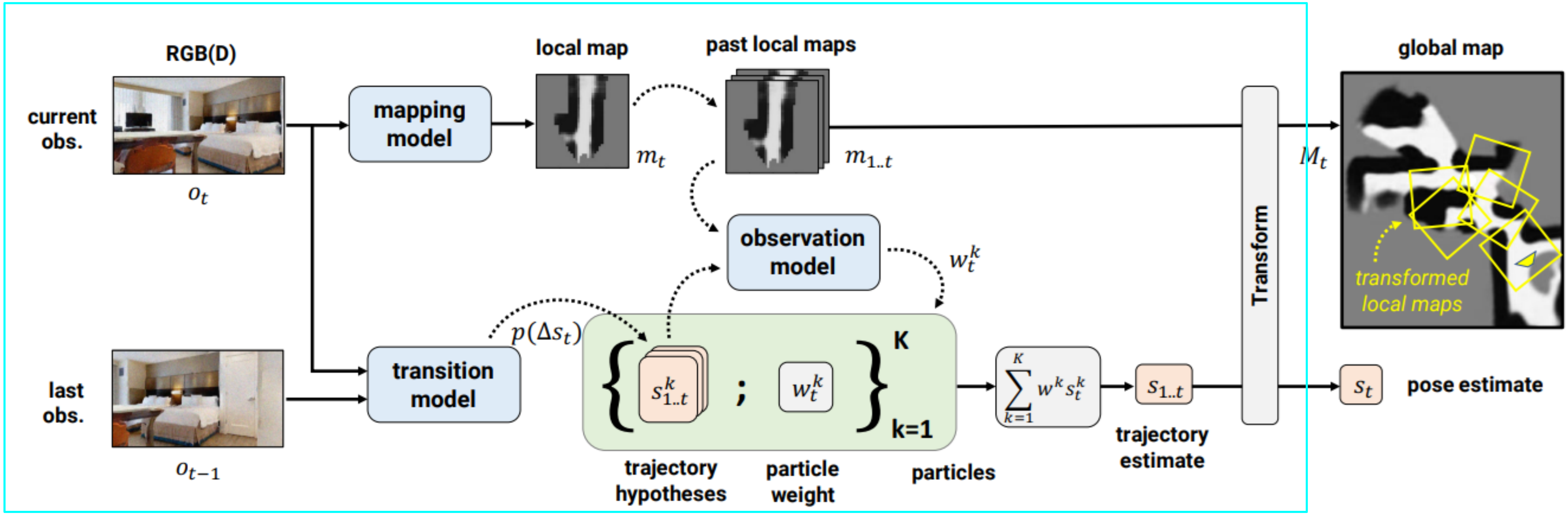
1. **First stage**: Train the transition model
2. Second stage : Train the mapping and observation model together
3. Third stage: Fine tune the whole pipeline

Multi stage training



1. First stage: Pre train the transition model
2. **Second stage** : Pre train the mapping and observation model together
3. Third stage: Fine tune the whole pipeline

Multi stage training



1. First stage: Pre train the transition model
2. Second stage : Pre train the mapping and observation model together
3. **Third stage**: Fine tune the whole pipeline

Training: Other details

- Computational and space complexity: In backpropagation through large computational graph
  - To avoid this,
    - Short trajectories
    - Less particles (*K = 32 in this case*)
    - Testing time: Full trajectories estimation with $K = 128$

- Learning rate is decayed
  - If the validation error does not improve for 4 epochs
  - Perform 4 such decay steps, after which training terminates, and the model with the lowest validation error is stored
- Implemented in Tensorflow based on the open-source code of PF-net

Visual Navigation with SLAM-net

Coupled the SLAM-net pipeline with motion planner

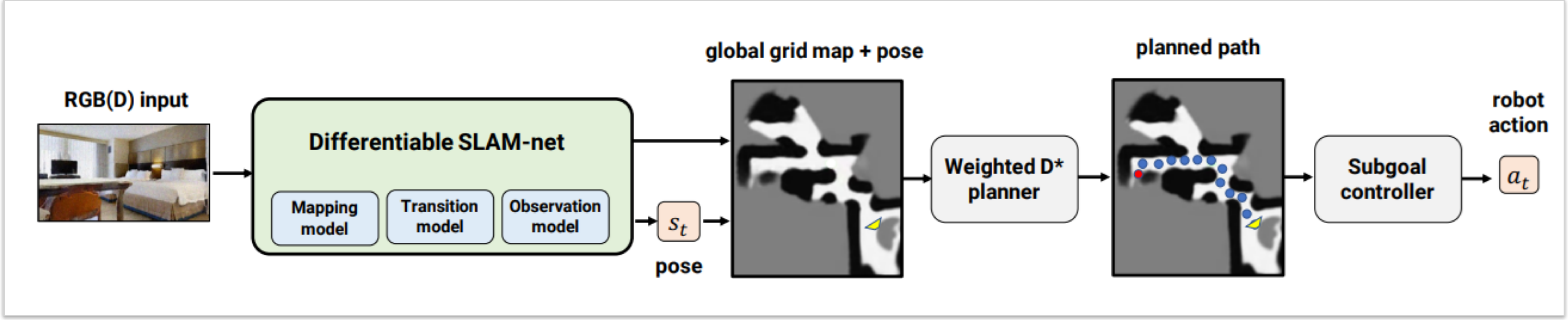


Figure: Visual navigation pipeline with the Differentiable SLAM-net, a path planner, and a motion controller.

- Makes the pipeline scalable
- Integrated with D-star planner
- Tested on Habitat 2020 PointNav challenge
- Collision recovery mechanism

Experiments

Datasets

Experiments are conducted in Habitat simulator

1. **Gibson dataset:**
   a. **72 scenes for training**
   b. 7 scenes for validation and 7 scenes for testing
2. **Replica and Matterport data:** Used for testing transfer learning
3. **KITTI data:**
   a. 06 and 07 - validation
   b. 09 and 10 - testing
   c. Rest - training

Comparison

1. ORB SLAM
2. Fast SLAM
3. Learned Visual odometry
4. Blind baseline

Discussion

1. Noisy images
2. Transfer learning
3. Planning
4. Limitations

Analytical Results

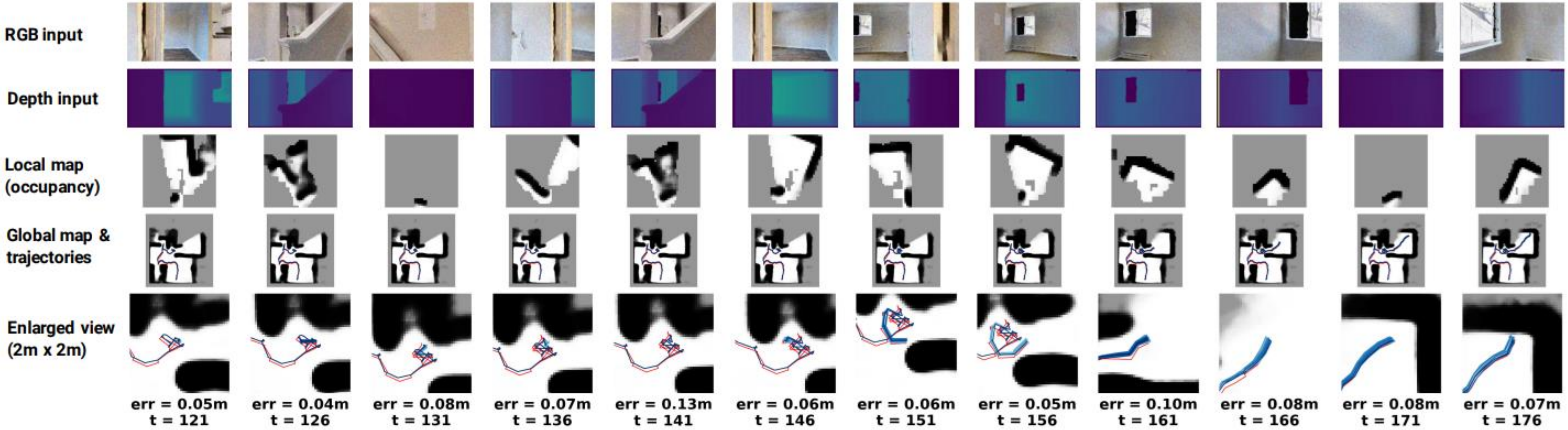
Sensor		RGBD		RGBD		RGBD		RGB	
Trajectory generator		traj_expert		traj_exp_rand		traj_nav		traj_expert	
Metric	runtime↓	SR↑	RMSE↓	SR↑	RMSE↓	SR↑	RMSE↓	SR↑	RMSE↓
SLAM-net (ours)	0.06s	**83.8%**	**0.16m**	**62.9%**	**0.28m**	**77.1%**	**0.19m**	**54.3%**	**0.26m**
Learned visual odometry	0.02s	60.0%	0.26m	24.8%	0.63m	30.5%	0.47m	28.6%	0.40m
FastSLAM [47]	–	21.0%	0.58m	0.0%	3.27m	21.9%	0.69m	X	X
ORB-SLAM [48]	0.08s	3.8%	1.39m	0.0%	3.59m	0.0%	3.54m	X	X
Blind baseline	0.01s	16.2%	0.80m	1.0%	4.13m	3.8%	1.50m	16.2%	0.80m

Three different navigation policies:

- The shortest-path expert (**traj_expert**);
- The shortest path expert mixed with random actions (**traj_exp_rand**);
- our final navigation pipeline (**traj_nav**).

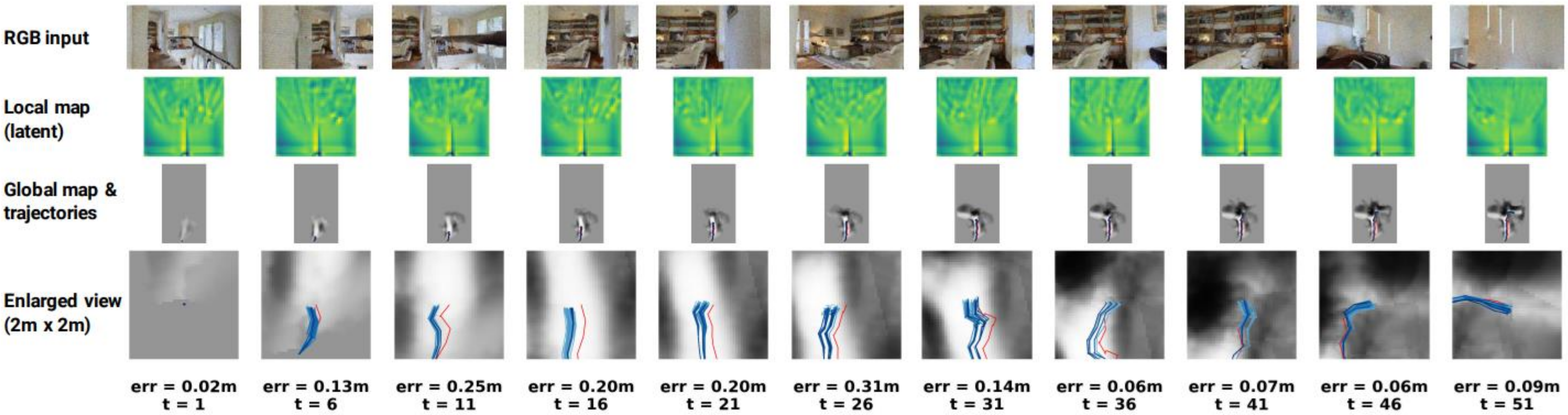
	RGBD	RGB
	SR↑	SR↑
Default conditions	3.8%	X
No sensor noise	7.5%	18.0%
No sensor and actuation noise	18.0%	20.4%
High frame rate	30.4%	X
Ideal conditions	86.0%	43.5%

Results



RGB input
Depth input
Local map (occupancy)
Global map & trajectories
Enlarged view (2m x 2m)
err = 0.05m t = 121
err = 0.04m t = 126
err = 0.08m t = 131
err = 0.07m t = 136
err = 0.13m t = 141
err = 0.06m t = 146
err = 0.06m t = 151
err = 0.05m t = 156
err = 0.10m t = 161
err = 0.08m t = 166
err = 0.08m t = 171
err = 0.07m t = 176

Results


RGB input
Local map
(latent)
Global map &
trajectories
Enlarged view
(2m x 2m)
err = 0.02m
t = 1
err = 0.13m
t = 6
err = 0.25m
t = 11
err = 0.20m
t = 16
err = 0.20m
t = 21
err = 0.31m
t = 26
err = 0.14m
t = 31
err = 0.06m
t = 36
err = 0.07m
t = 41
err = 0.06m
t = 46
err = 0.09m
t = 51

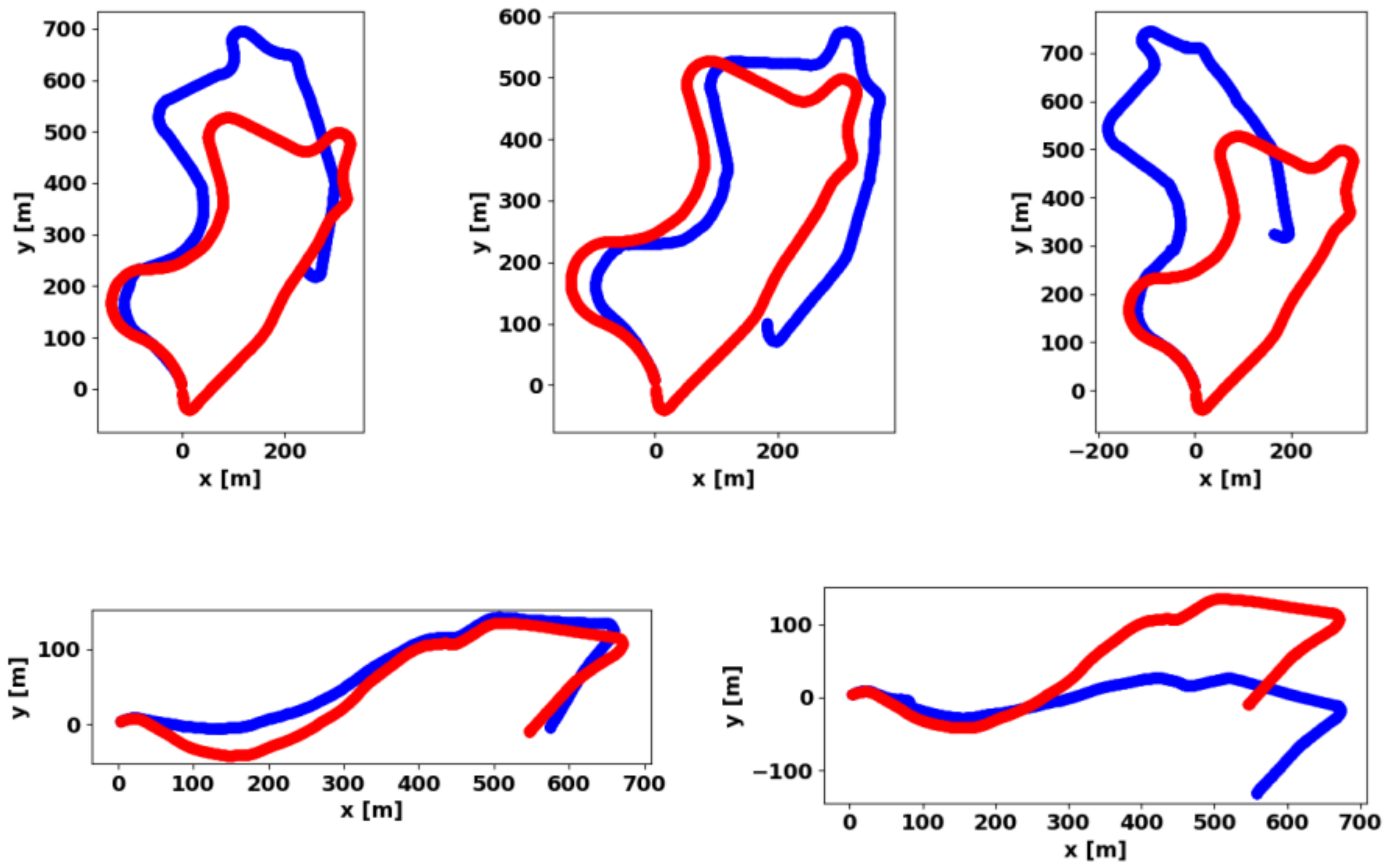
Ablation Results

Sensor Metric		RGBD SR↑	RGB SR↑
(1)	SLAM-net (default)	**77.1%**	**55.2%**
(2)	No joint training	66.7%	8.6%
(3)	Occupancy map only	75.2%	23.8%
(4)	Latent map only	70.5%	**55.2%**
(5)	Occupancy + latent map	**77.1%**	44.8%
(6)	Fixed comparisons (8)	44.8%	29.5%
(7)	Dynamic comparisons (4)	73.3%	41.9%
(8)	Dynamic comparisons (8)	**77.1%**	**55.2%**
(9)	Dynamic comparisons (16)	**77.1%**	40.0%
(10)	K=1 (VO)	30.5%	26.7%
(11)	K=8	60.0%	35.2%
(12)	K=32 (training)	72.4%	39.1%
(13)	K=64	75.2%	46.7%
(14)	K=128 (evaluation default)	77.1%	**55.2%**
(15)	K=256	79.1%	44.8%
(16)	K=512	**82.9%**	48.6%

Limitations

- Limitations
  - Limited evaluation on real-world datasets
  - Limited comparison with recent learning-based SLAM approaches
  - Lack of analysis of failure cases
  - High dependence on the depth aspect of data is relatively more.

- Future directions
  - Explore multi-modal sensor fusion
  - Generalization to other tasks
  - Robustness analysis(depth & dynamic invariance)



Trajectory Metric	Kitti-09 RMSE↓	Kitti-10 RMSE↓
SLAM-net (ours)	83.5m	15.8m
SLAM-net (best of 5)	56.9m	12.8m
Learned visual odometry	71.1m	73.2m
ORB-SLAM-RGB [49]	**7.62m**	**8.68m**

Conclusions

- Introduced a learning-based differentiable SLAM approach with strong performance on challenging visual localization data and on downstream robot navigation, achieving SOTA in the Habitat 2020 PointNav task.

- The authors believe that their work on differentiable SLAM-net may lay foundation to a new class of methods that learn robust, task oriented features for SLAM for both optimization and particle filtering-based approaches.

- A differentiable SLAM pipeline can enable the development of end-to-end learning-based SLAM algorithms which leads to automatic learning of feature representations and parameter tuning.

Thank you

Next Time: Scene-Level Representations

- Seminar 7: Semantic Scene Graphs and Explicit Representations
  1. Image Retrieval using Scene Graphs, Johnson et al., 2015
  2. Semantic Robot Programming for Goal-Directed Manipulation in Cluttered Scenes, Zeng et al., 2018
  3. Semantic Linking Maps for Active Visual Object Search, Zeng et al., 2020
  4. Hydra: A Real-time Spatial Perception System for 3D Scene Graph Construction and Optimization, Hughes et al., 2022
- Seminar 8: Neural Radiance Fields and Implicit Representations
  1. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis, Mildenhall et al., 2020
  2. iMAP: Implicit Mapping and Positioning in Real-Time, Sucar et al., 2021
  3. NeRF-SLAM: Real-Time Dense Monocular SLAM with Neural Radiance Fields, Rosinol et al., 2022
  4. NeRF-Supervision: Learning Dense Object Descriptors from Neural Radiance Fields, Yen-Chen et al., 2022
  5. NARF22: Neural Articulated Radiance Fields for Configuration-Aware Rendering, Lewis et al., 2022

DeepRob

Seminar 6
Visual Odometry and Localization
University of Michigan and University of Minnesota