UC San Diego

SE3-Pose-Nets: Structured Deep Dynamics Models for Visuomotor Planning and Control

Presenter: Kai-En Lin

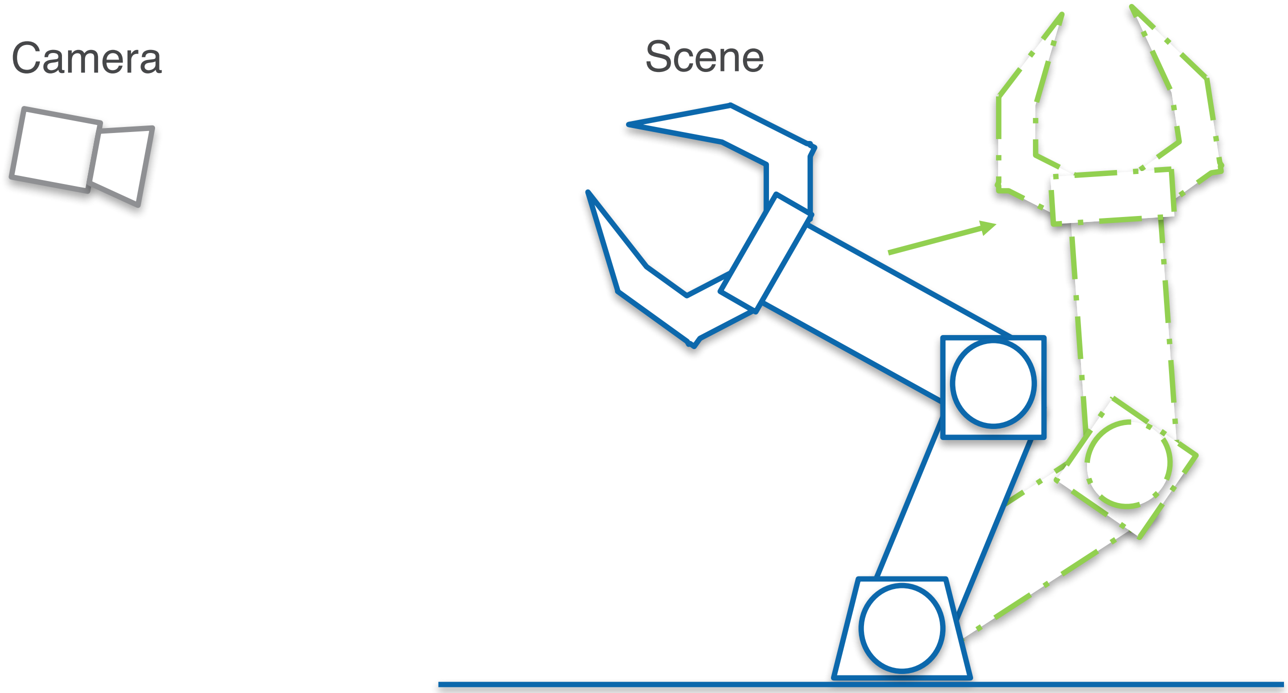
5/14/2020

Outline

- Introduction
- Related work
  - SE3-Nets
- Algorithm
- Experiments
- Conclusion
- Future Work

Introduction

- Problem statement
  - Observe a scene with a camera
  - Control the robot to reach a target



Introduction

- Traditional approach
  - Data-associate the observed scene to target
    - E.g. tracking different parts of the robot
  - Model the effect of applied actions to changes to the scene
    - E.g. knowing what happens after the action
- Deep Learning approach
  - Tries to learn similar models
  - Lacks the ability to associate objects/parts across scenes

Introduction

- Goal:
  - Devise a learning-based algorithm that allows:
    - Data-association
    - Modeling of the object dynamics
    - Correct prediction and control from the model

Related Work

- SE3-Nets
  - Segment object parts
  - Predict SE(3) transformation for each part to target
  - No explicit modeling of data association

Algorithm

- Given
  - an observation $\mathrm{x_t}$ of the scene (depth map / point cloud)
  - applied actions $\mathrm{u_t}$
- Predict
  - the transformed output point cloud $\mathrm{x_{t+1}}$

Algorithm

- We can decompose the problem of modeling scene dynamics into:
  1. Modeling scene structure
  2. Modeling the dynamics of individual parts
  3. Combining local pose changes to model the dynamics of the entire scene

- With deep learning:
  1. An encoder to distinguish individual parts and predict a 6D pose for each of them
  2. A pose transition network to model the dynamics in the pose space. Takes source pose and action to predict the change in poses
  3. A transform layer to apply SE(3) transforms to input point cloud using predicted pose deltas

Algorithm

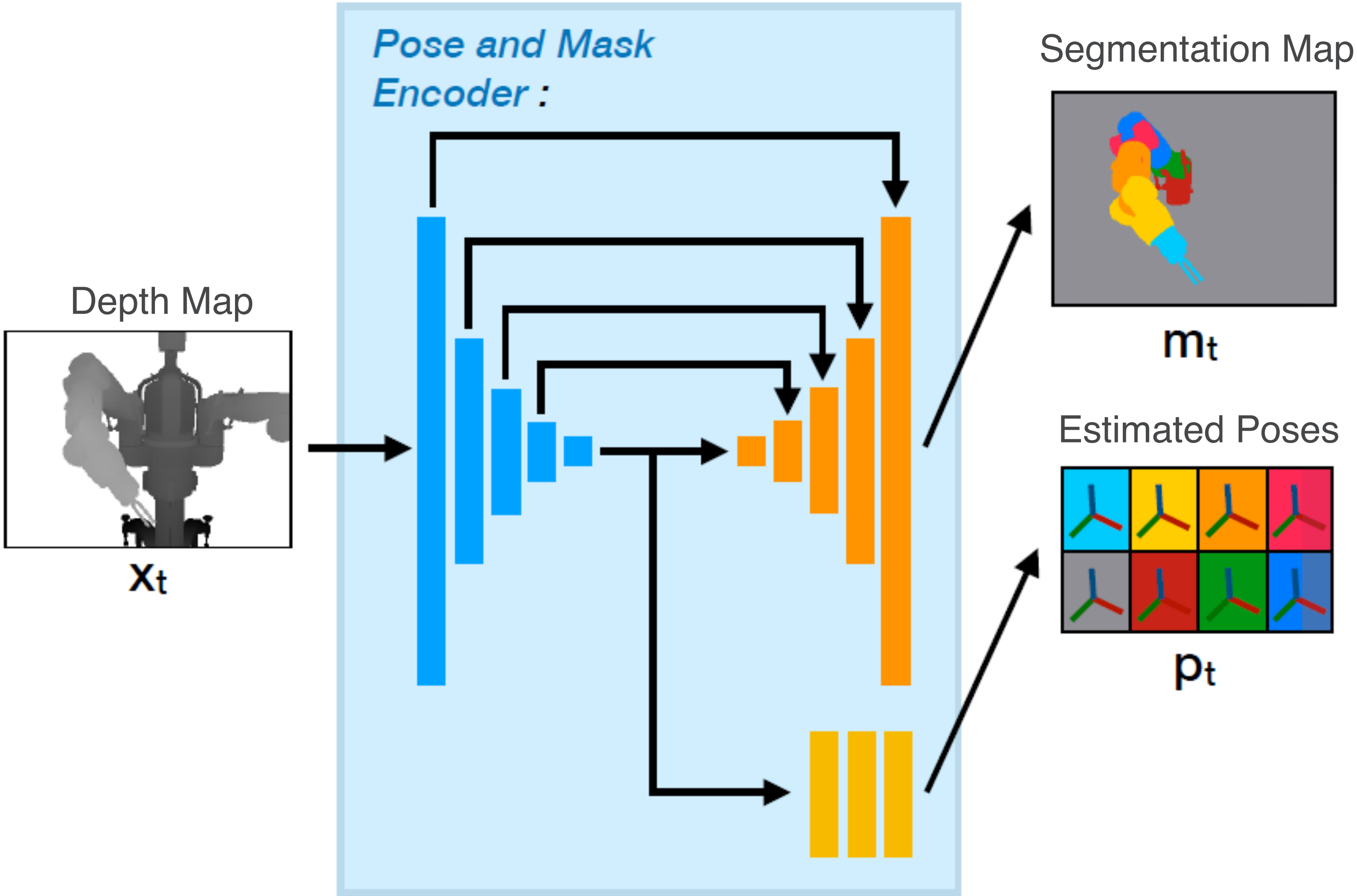
- We can decompose the problem of modeling scene dynamics into:
  1. Modeling scene structure
  2. Modeling the dynamics of individual parts
  3. Combining local pose changes to model the dynamics of the entire scene

- With deep learning:
  1. An encoder to distinguish individual parts and predict a 6D pose for each of them
  2. A pose transition network to model the dynamics in the pose space. Takes source pose and action to predict the change in poses
  3. A transform layer to apply SE(3) transforms to input point cloud using predicted pose deltas

Modeling Scene Structure

- An encoder takes the input 3D point cloud x_t and generates the following:
  - Masks for the moving parts (m_t)
  - 6D pose per segmented part (p_t)
    - 3D position
    - Orientation as 3-parameter axis-angle vector

Modeling Scene Structure



Algorithm

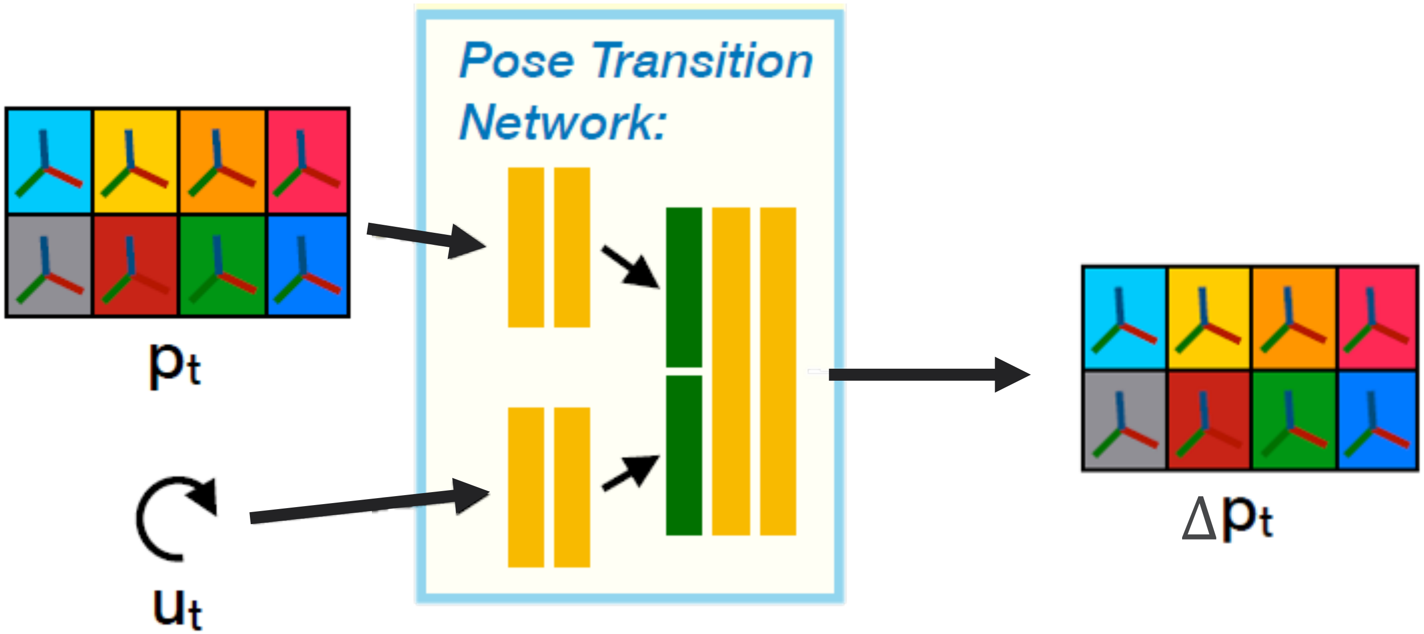
- We can decompose the problem of modeling scene dynamics into:
  1. Modeling scene structure
  2. Modeling the dynamics of individual parts
  3. Combining local pose changes to model the dynamics of the entire scene

- With deep learning:
  1. An encoder to distinguish individual parts and predict a 6D pose for each of them
  2. A pose transition network to model the dynamics in the pose space. Takes source pose and action to predict the change in poses
  3. A transform layer to apply SE(3) transforms to input point cloud using predicted pose deltas

Modeling Part Dynamics

- A fully-connected pose transition network takes the predicted poses from the encoder ($\mathrm{p_t}$) and applied actions ($\mathrm{u_t}$) as input and predicts:
  - The change in pose ($\Delta\mathrm{p_t}$) for all K segmented parts (6D vector)

Modeling Part Dynamics



Algorithm

- We can decompose the problem of modeling scene dynamics into:
  1. Modeling scene structure
  2. Modeling the dynamics of individual parts
  3. Combining local pose changes to model the dynamics of the entire scene

- With deep learning:
  1. An encoder to distinguish individual parts and predict a 6D pose for each of them
  2. A pose transition network to model the dynamics in the pose space. Takes source pose and action to predict the change in poses
  3. A transform layer to apply SE(3) transforms to input point cloud using predicted pose deltas

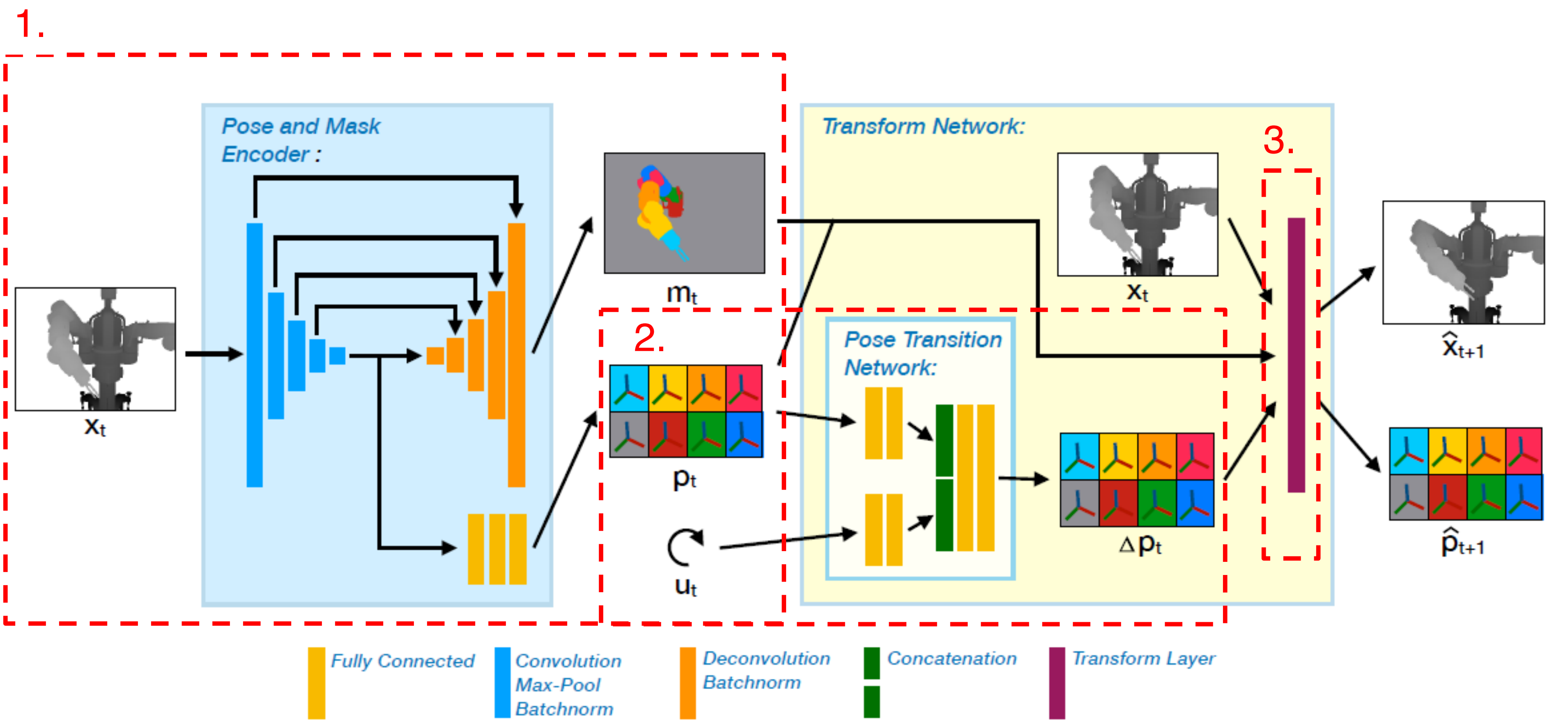
Predicting Scene Dynamics

- No trainable parameters
- Given point cloud ($\mathrm{x_t}$), the predicted scene segmentation ($\mathrm{m_t}$) and the change in poses ($\Delta \mathrm{p_t}$), calculates the point cloud in the next frame ($\mathrm{x_{t+1}}$):

$$\hat{x}_{t+1}^{j} = \sum_{k=1}^{K} m_t^{kj} \left(R_t^k x_t^j + T_t^k \right),$$

where R_t^k is rotation, T_t^k translation

Algorithm



1.
Pose and Mask
Encoder :
x_t
m_t
2.
p_t
u_t
Transform Network:
3.
x_t
Pose Transition
Network:
Δp_t
$\hat{x}_{t+1}$
$\hat{p}_{t+1}$
Fully Connected
Convolution
Max-Pool
Batchnorm
Deconvolution
Batchnorm
Concatenation
Transform Layer

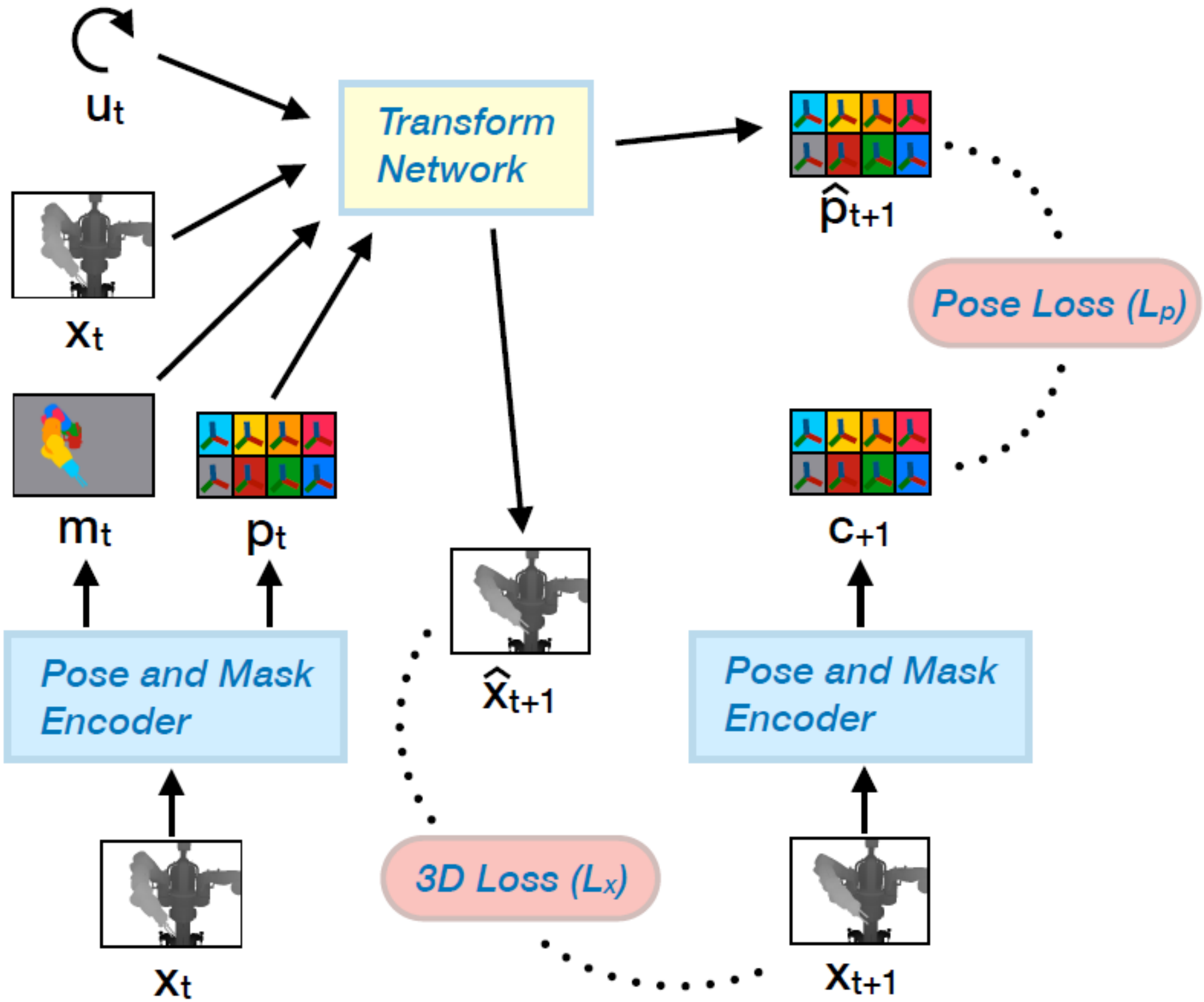
Training

- Supervision
  - Point-wise data associations across a pair of point clouds $(\mathrm{x}_t, \mathrm{x}_{t+1})$
  - Related by an action (u_t)

Training

- Total loss $L = L_x + \gamma L_p$
  - 3D Loss L_x
  - Pose consistency loss L_p
  - $\gamma = 10$

Training



ut
Transform Network
xt
mt
pt
p̂t+1
Pose Loss (Lp)
c+1
x̂t+1
Pose and Mask Encoder
Pose and Mask Encoder
3D Loss (Lx)
xt
xt+1

Training

- 3D Loss L_x

$$L_x = \frac{1}{N}\sum_{i=1}^{HW}\frac{\left(\hat{x}_{t+1}^{i} - \tilde{x}_{t+1}^{i}\right)^2}{\alpha\tilde{f}^{i} + \beta},$$

  where HW is the number of points,
  $\alpha = 0.5$, $\beta = 1e-3$,
  $\tilde{f}^{i} = \tilde{x}_{t+1}^{i} - x_t^i$, scaling factor to make the loss scale-invariant

Training

- Pose consistency loss L_p

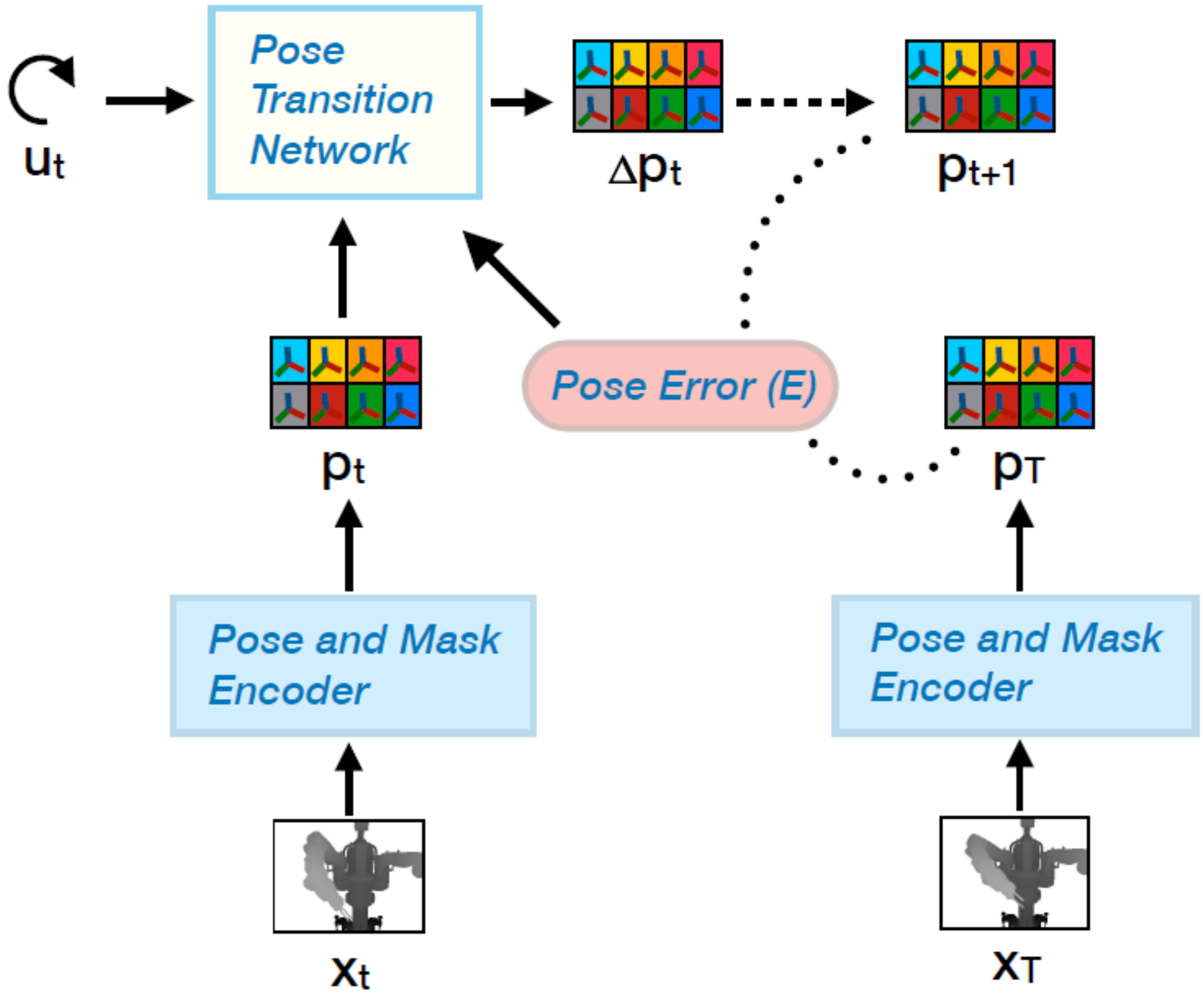
$$L_p = \frac{1}{I}\sum_{i=1}^{I}\left(\hat{p}_{t+1}^{i} - p_{t+1}^{i}\right)^2,$$

where $\hat{\mathrm{p}}_{t+1} = \mathrm{p}_t \oplus \Delta\mathrm{p}_t$, the expected pose at time t+1

Closed-Loop Visuomotor Control Using SE3-Pose-Nets

- Visual servoing
  - Given the current image and the target image, generate controls to reach the target
- SE3-Pose-Nets solve this by using the latent pose space to data-associate the observations and minimizing the error between initial pose p_0 and the final pose p_T.

Closed-Loop Visuomotor Control Using SE3-Pose-Nets



Closed-Loop Visuomotor Control Using SE3-Pose-Nets

Algorithm 1 Reactive visuomotor control

Given: Target point cloud ($\mathbf{x}_T$)
Given: Pre-trained encoder (h_{enc}) and transition model (h_{trans})
Given: Maximum control magnitude: u_{max}
Compute target pose: $\mathbf{p}_T = h_{enc}(\mathbf{x}_T)$
while not converged **do**
  Receive current observation ($\mathbf{x}_t$)
  Predict current pose: $\mathbf{p}_t = h_{enc}(\mathbf{x}_t)$
  Initialize control to all zeros: $\mathbf{u}_t = 0$
  Predict change in pose: $\Delta\mathbf{p}_t = h_{trans}(\mathbf{p}_t, \mathbf{u}_t)$
  Predict next pose: $\hat{\mathbf{p}}_{t+1} = \mathbf{p}_t \oplus \Delta\mathbf{p}_t$
  Compute pose error: $E = \frac{1}{I}\sum_{i=1}^{I}(\hat{p}^i_{t+1} - p^i_T)^2$
  Compute gradient of error w.r.t. control: $g = \dfrac{dE}{dU_t}$
  Compute control: $\mathbf{u}_t = -u_{max} * \dfrac{g}{||g||}$
  Execute control $\mathbf{u}_t$ on the robot

Experiments

- SE3-Pose-Nets performs worse than other models when predicting scene dynamics
  - The pose space might make the training problem harder
  - Constraint on pose consistency is different from the prediction problem

Setting	SE3-POSE-NETS	SE3-POSE-NETS + Joint Angles	SE3-NETS	SE3-NETS + Joint Angles	Flow	Flow + Joint Angles
Simulated	0.044	0.038	0.030	**0.024**	0.035	0.030
Real	0.234	0.224	0.221	**0.212**	0.228	0.218

TABLE I: Average per-point flow MSE (cm) across tasks and networks, normalized by the number of points M that move in the ground truth data (motion magnitude > 1mm). Our network achieves results slightly worse than the baseline networks on both simulated and real data. However, it is also solving additional tasks necessary for control.

Experiments

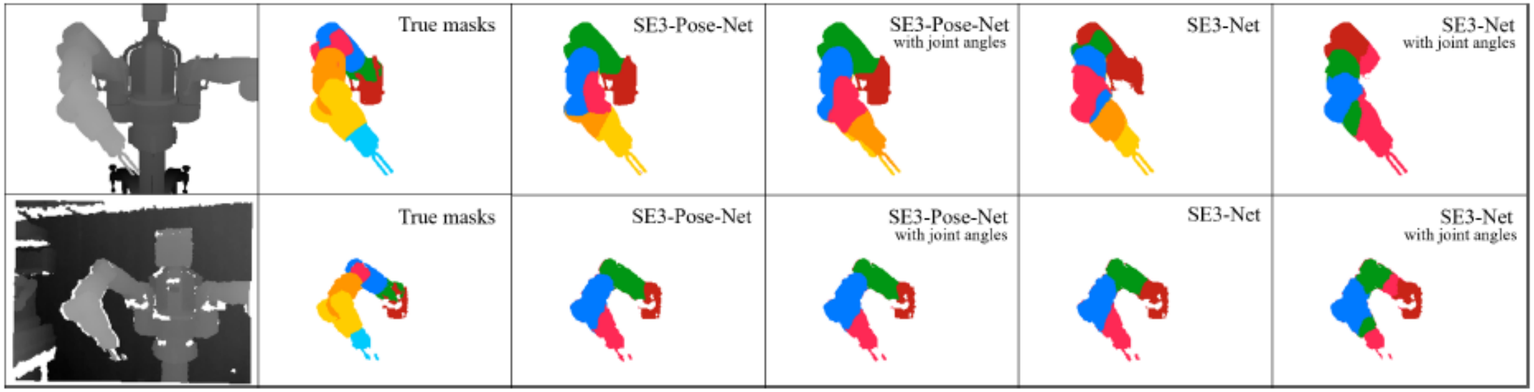


Fig. 3: Masks generated by different networks on simulated (top) and real data (bottom). From left to right: Ground truth depth, ground truth masks, masks predicted by the SE3-POSE-NET, SE3-POSE-NET with joint angles, SE3-NET and SE3-NET with joint angles.

Experiments

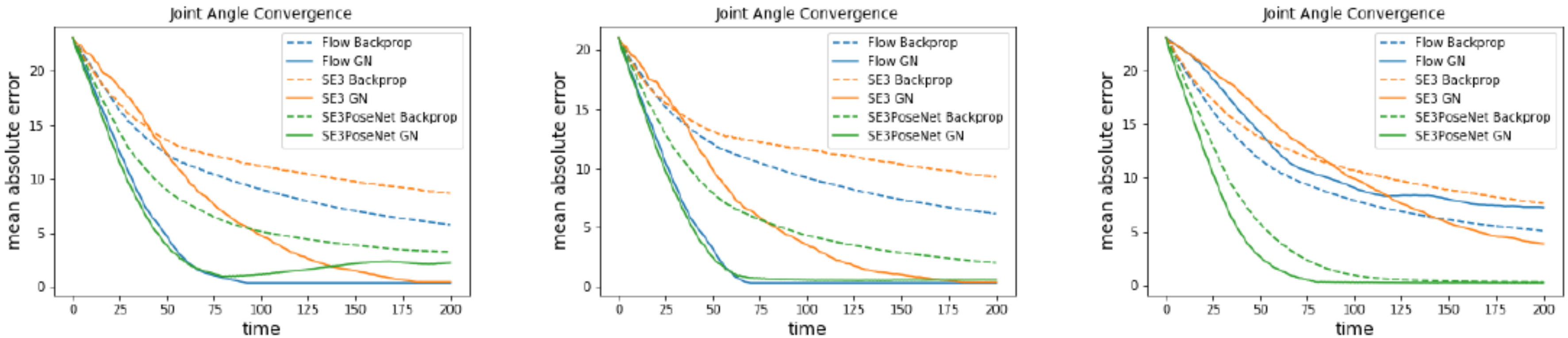


Fig. 4: Convergence of joint angle error in simulated Baxter control tasks. (left): without joint angles, (middle) without joint angles and detected failure case removed (for all methods), (right) with joint angles. SE3-POSE-NETS perform as well or better than baseline methods even though baseline models have additional information in the form of ground truth-associations.

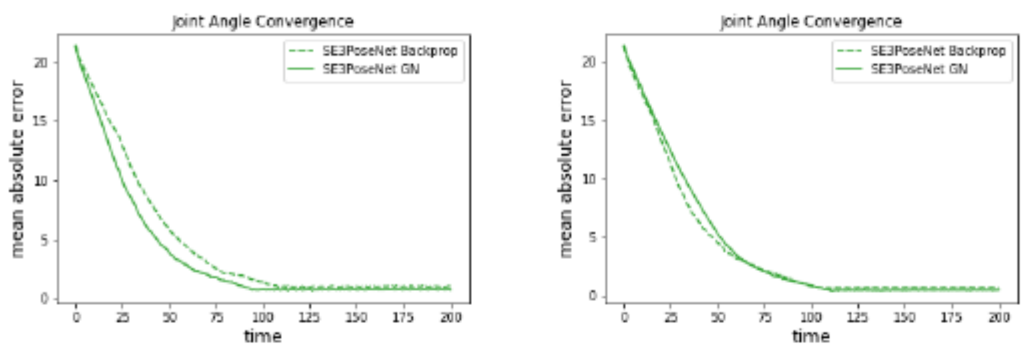
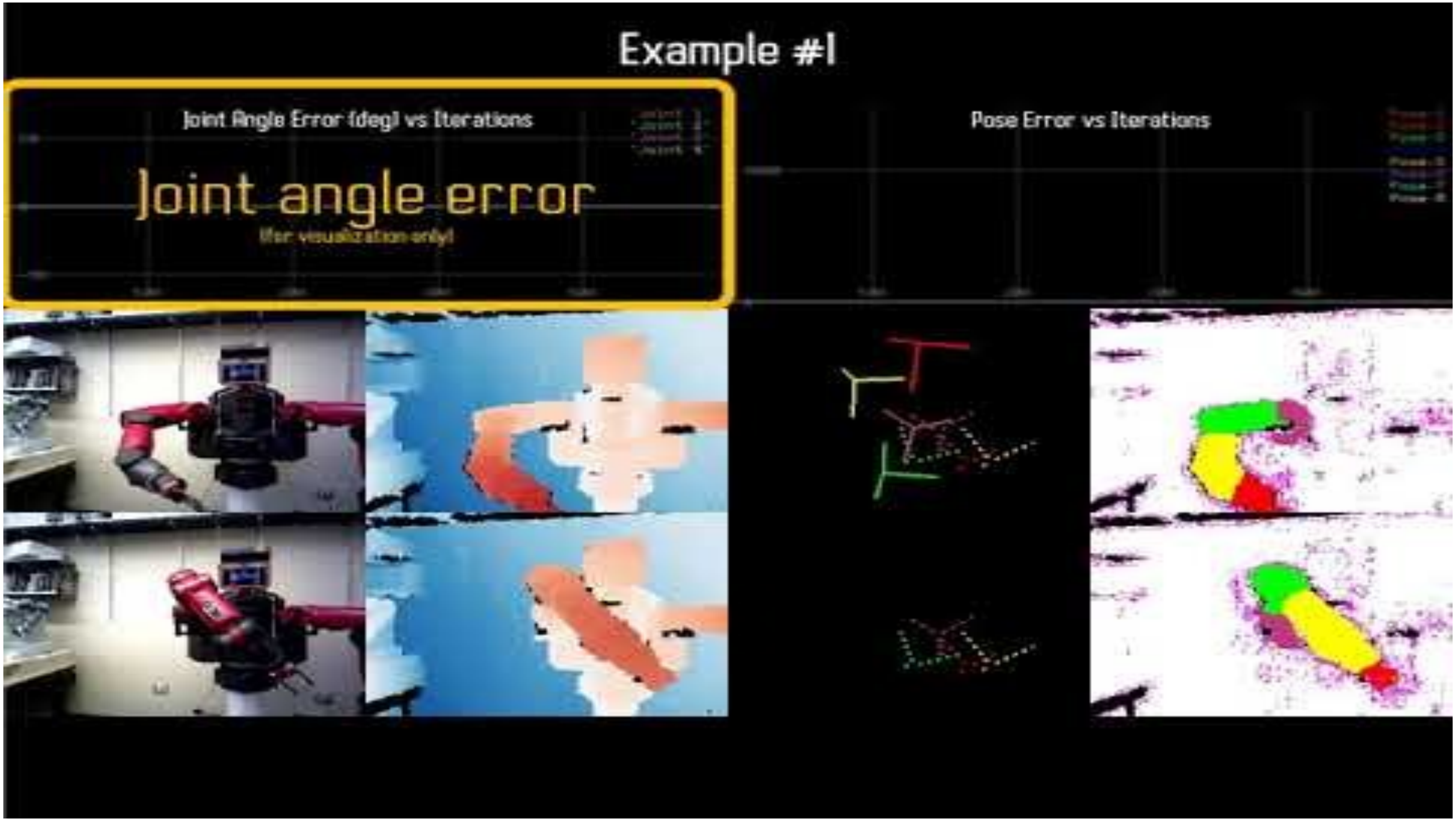


Fig. 5: Convergence of joint angle error on real Baxter control tasks (left) without joint angles (right) with joint angles (averaged across joint 0,1,2,3).

Experiments



Conclusion

- SE3-Pose-Nets is an end-to-end framework for learning predictive models that enable control of objects in a scene
- It learns a consistent pose space for each individual part
- Does not require external data association
- The network enables computation of controls in the low dimensional pose space

Future Work

- SE3-Pose-Nets has difficulties handling joints further down the kinematic chain
- Extending the system to interact with and manipulate external objects
- Long-term planning to utilize the latent pose space