A Details in α-MDF

This section provides a detailed overview of the previously mentioned α-MDF modules, and describes differentiable Ensemble Kalman filters as the underlying DFs framework for α-MDF.

A.1 Model Initialization and Embedding Functions

An auxiliary model $\mathcal{A}$ is supplied in the filtering process to support training by starting the filter via projecting the actual state $\boldsymbol{x}_{t-N:t-1}$ from low-dimensional space to latent space. The model is implemented using stochastic neural networks (SNNs) [42],

$$\mathbf{x}^i_{t-N:t-1} \sim \mathcal{A}(\mathbf{x}^i_{t-N:t-1}|\boldsymbol{x}_{t-N:t-1}),\ \forall i \in E, \tag{7}$$

where $\mathbf{x}^i_{t-N:t-1}$ is one latent state, the latent state ensemble is obtained by sampling $\mathcal{A}$ for E times. During inference, we employ the trained sensor encoders' output, which is the latent representation of RGB, depth, or proprioception, as the initial state to initiate the filtering process.

Regarding the prediction step of α-MDF, we apply positional embedding layers (sinusoidal functions) [32] in the transformer process model (Eq. 3) to generate $\boldsymbol{e}_{t-N:t-1}$ as the embedding for time-series data, $\boldsymbol{e}_{t-N:t-1} = f_{\mathcal{L}}(\mathbf{X}_{t-N:t-1}) \in \mathbb{R}^{d_x \times (N-1)}$. The positional embedding layer is utilized to label the state by index it with time t. When activating the action $\mathbf{a}_t$ in the process model, $\boldsymbol{e}_{t-N:t-1}$ is passed through a type embedding layer that indexes $\boldsymbol{e}_{t-N:t-1}$ and $\mathbf{a}_t$ with 0 and 1, and then fed to sinusoidal functions. Subsequently, the outputs obtained from the aforementioned procedures serve as input to the transformer process model for further processing.

A.2 Differentiable Ensemble Kalman Filter

Unlike prior proposals for differentiable filters, such as dEKF [9] and DPF [24], Differentiable Ensemble Kalman Filter [6] leverages recent advancements in stochastic neural networks (SNNs) [42]. Specifically, we draw inspiration from the work in [43], which established a theoretical connection between the Dropout training algorithm and Bayesian inference in deep Gaussian processes. As a result, we can use stochastic forward passes to produce empirical samples from the predictive posterior of a neural network trained with Dropout. Hence, for the purposes of filtering, we can implicitly model the process noise by sampling state from a neural network trained on the transition dynamics, i.e., $\mathbf{x}_t \sim f_{\boldsymbol{\theta}}(\mathbf{x}_{t-1})$. In contrast to previous approaches [24, 9], the transition network $f_{\boldsymbol{\theta}}(\cdot)$ models the system dynamics, as well as the inherent noise model in a consistent fashion without imposing diagonality.

Prediction Step: Similar to α-MDF, we use an initial ensemble of E members to represent the initial state distribution $\mathbf{X}_0 = [\mathbf{x}^1_0, \dots, \mathbf{x}^E_0]$, $E \in \mathbb{Z}^+$. We leverage the stochastic forward passes from a trained state transition model to update each ensemble member:

$$\mathbf{x}^i_{t|t-1} \sim f_{\boldsymbol{\theta}}(\mathbf{x}^i_{t|t-1}|\mathbf{x}^i_{t-1|t-1}),\ \forall i \in E. \tag{8}$$

Matrix $\mathbf{X}_{t|t-1} = [\mathbf{x}^1_{t|t-1}, \cdots, \mathbf{x}^E_{t|t-1}]$ holds the updated ensemble members which are propagated one step forward through the state space. Note that sampling from the transition model $f_{\boldsymbol{\theta}}(\cdot)$ (using the SNN methodology described above) implicitly introduces a process noise.

Update step: Given the updated ensemble members $\mathbf{X}_{t|t-1}$, a nonlinear observation model $h_{\boldsymbol{\psi}}(\cdot)$ is applied to transform the ensemble members from the state space to observation space. Following our main rationale, the observation model is realized via a neural network with weights $\boldsymbol{\psi}$. Accordingly, the update equations become:

$$\mathbf{H}_t\mathbf{A}_t = \mathbf{H}_t\mathbf{X}_t - \left[\frac{1}{E}\sum_{i=1}^{E} h_{\boldsymbol{\psi}}(\mathbf{x}^i_t), \cdots, \frac{1}{E}\sum_{i=1}^{E} h_{\boldsymbol{\psi}}(\mathbf{x}^i_t)\right], \quad (9) \qquad \tilde{\mathbf{y}}^i_t \sim s(\tilde{\mathbf{y}}^i_t|\mathbf{y}_t),\ \forall\, i \in E. \quad (10)$$

$\mathbf{H}_t\mathbf{X}_t$ is the predicted observation, and $\mathbf{H}_t\mathbf{A}_t$ is the sample mean of the predicted observation at t. Traditional Ensemble Kalman Filter treats observations as random variables. Hence, the ensemble can incorporate a measurement perturbed by a small stochastic noise to reflect the error covariance of the best state estimate [6]. In differentiable Ensemble Kalman Filter, we incorporate a Bayesian sensor encoder $s(\cdot)$. Sensor encoder serves to learn projections from observation space to latent space as in Eq. 10, where $\mathbf{y}_t$ represents the noisy sensor observation. Sampling from sensor encoder yields latent observations $\tilde{\mathbf{Y}}_t = [\tilde{\mathbf{y}}_t^1, \cdots, \tilde{\mathbf{y}}_t^{E)}]$. The KF update step can then be continued by using the learned observation and predicted observation:

$$\mathbf{K}_t = \frac{1}{E-1}\mathbf{A}_t(\mathbf{H}_t\mathbf{A}_t)^T(\frac{1}{E-1}(\mathbf{H}_t\mathbf{A}_t)(\mathbf{H}_t\mathbf{A}_t)^T + \mathbf{R})^{-1}. \tag{11}$$

The measurement noise model $\mathbf{R}$ is implemented using a multilayer perceptron (MLP), similar to the implementation in [9]. The MLP takes a learned observation $\tilde{\mathbf{Y}}_t$ at time t and produces a noise covariance matrix. The final estimate of the ensemble $\hat{\mathbf{X}}_t$ is obtained by performing the measurement update step, given by:

$$\hat{\mathbf{X}}_t = \mathbf{X}_t + \mathbf{K}_t(\tilde{\mathbf{Y}}_t - \mathbf{H}_t\mathbf{X}_t). \tag{12}$$

In inference, the ensemble mean $\bar{\mathbf{x}}_{t|t} = \frac{1}{E}\sum_{i=1}^{E}\mathbf{x}_{t|t}^i$ is used as the updated state.

A.3 Baselines

In our study, we examine two categories of baselines: (a) DFs baselines, which consist of existing methods such as those proposed in [9, 24, 22], and (b) sensor fusion strategies, as proposed in [18].

Table 4: Dimensions pertinent to each of the robot state estimation tasks.

Method	Visual Odometry		UR5 Manipulation		Soft Robot		
	State	Observation	State	Observation	State	Observation	Action
dEKF [9]	5	2	10	10	7	7	40
DPF [24]	5	2	10	10	7	7	40
dPF-M-lrn [9]	5	2	10	10	7	7	40
Feature Fusion [18]	-	-	10/13	10/13	7	7	40
Unimodal [18]	-	-	10/13	10/13	7	7	40
Crossmodal [18]	-	-	10/13	10/13	7	7	40
α-MDF	256	256	256	256	256	256	40

Dimensionality: Table 4 presents the dimensions for the state, observations, and actions utilized for each of the tasks. To ensure consistency, we opt for a dimension of 256 for α-MDF in all tasks, thus, enabling filtering over high-dimensional spaces. Unlike the baseline methods, which use low-dimensional state definitions, we filter over higher dimension spaces with α-MDF.

Differentiable Filters: To maintain consistency in the comparison of results against the DFs baselines, we train α-MDF with a single modality. The baselines in this category include the differentiable Extended Kalman filter (dEKF) [9], differentiable particle filter (DPF) [24], and the modified differentiable particle filter (dPF-M-lrn) [9], which uses learned process and process noise models. For dEKF, the Jacobian matrix in the prediction step can either be learned end-to-end or supplied if the motion model is known. DPF employs 100 particles for both training and testing and also incorporates an observation likelihood estimation model l. This module takes in an image embedding and produces a likelihood that updates each particle's weight. Unlike DPF, dPF-M-lrn implements a learnable process noise model. It also adopts a Gaussian Mixture Model for calculating the likelihood for all particles. It is worth noting that all the baseline methods perform Kalman filtering on low-dimensional actual state space, whereas α-MDF executes the filtering process in the latent space.

Sensor Fusion: Regarding sensor fusion baselines, we use three strategies discussed in [18], namely, Feature Fusion, Unimodal Fusion, and Crossmodal Fusion. The Feature Fusion strategy aims to process each modality individually and subsequently merge the modalities to generate a multimodal feature set using neural networks, which is then used for state estimation. The Unimodal Fusion treats each modality $\mathcal{N} \sim (\boldsymbol{\mu}_t^{M_1}, \boldsymbol{\Sigma}_t^{M_1})$ and $\mathcal{N} \sim (\boldsymbol{\mu}_t^{M_2}, \boldsymbol{\Sigma}_t^{M_2})$ as distributions and fuse two unimodal distribution as one normally distributed multimodal distribution $\mathcal{N} \sim (\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$:

$$\boldsymbol{\mu}_t = \frac{(\boldsymbol{\Sigma}_t^{M_1})^{-1}\boldsymbol{\mu}_t^{M_1} + (\boldsymbol{\Sigma}_t^{M_2})^{-1}\boldsymbol{\mu}_t^{M_2}}{(\boldsymbol{\Sigma}_t^{M_1})^{-1} + (\boldsymbol{\Sigma}_t^{M_2})^{-1}}, \;\; \boldsymbol{\Sigma}_t = ((\boldsymbol{\Sigma}_t^{M_1})^{-1} + (\boldsymbol{\Sigma}_t^{M_2})^{-1})^{-1}, \tag{13}$$

where the associative property can be used for fusing more than two modalities. For Crossmodal Fusion, information from one modality can be used to determine the uncertainty of the other ones, two coefficients are proposed as $\boldsymbol{\beta}_t^{M_1}$ and $\boldsymbol{\beta}_t^{M_2}$, where each coefficient has the same dimension of the state, the fused distribution is:

$$\boldsymbol{\mu}_t = \frac{\boldsymbol{\beta}_t^{M_1} \circ \boldsymbol{\mu}_t^{M_1} + \boldsymbol{\beta}_t^{M_2} \circ \boldsymbol{\mu}_t^{M_2}}{\boldsymbol{\beta}_t^{M_1} + \boldsymbol{\beta}_t^{M_2}}, \;\; \boldsymbol{\Sigma}_t = \frac{\boldsymbol{B}_t^{M_1} \circ \boldsymbol{\Sigma}_t^{M_1} + \boldsymbol{B}_t^{M_2} \circ \boldsymbol{\Sigma}_t^{M_2}}{\boldsymbol{B}_t^{M_1} + \boldsymbol{B}_t^{M_2}}, \tag{14}$$

where $\boldsymbol{B}_t^M = (\boldsymbol{\beta}_t^M)^T \boldsymbol{\beta}_t^M$. As mentioned in [18], each sensor encoder was independently trained and subsequently used for end-to-end training with DFs. We adopt a similar approach, but with a differentiable Ensemble Kalman Filter backbone in place instead. The resampling procedure from the fused distribution in this scenario is achieved by using the reparameterization trick [44].

B Additional Experiments

This section presents supplementary experimental results for each task. For (1) Visual Odometry Tasks, we offer full detailed experiments; however, for (2) Multimodal Manipulation Tasks and (3) Soft Robot Modeling Tasks, we concentrate mainly on ablation studies.

B.1 Visual Odometry Tasks

In this experiment, we investigate the performance of α-MDF on the popular KITTI Visual Odometry dataset [34]. We only consider RBG images as the input modality in order to make a fair comparison with the baselines [9, 24, 22]. Following the same evaluation procedure as our baselines, we define the actual state of the moving vehicle as a 5-dimensional vector $\boldsymbol{x} = [x, y, \theta, v, \dot{\theta}]^T$, including the position and orientation of the vehicle, and the linear and angular velocity w.r.t. the current heading direction θ. The raw observation $\mathbf{y}$ corresponds to the RGB camera image of the current frame and a difference image between the current frame and the previous frame, where $\mathbf{y} \in \mathbb{R}^{150 \times 50 \times 6}$ as shown in Fig. 8. The learned observation $\tilde{\mathbf{y}}$ is defined as $\tilde{\mathbf{y}} = [v, \dot{\theta}]^T$, since only the relative changes of position and orientation can be captured between two frames. We use the latent state $\mathbf{x} \in \mathbb{R}^{256}$ for α-MDF.

Data: The KITTI Visual Odometry dataset includes 11 trajectories capturing the ground truth pose (translation and rotation matrices) of a vehicle navigating urban areas at a data collection rate of approximately 10Hz. To facilitate the learning process, we standardize the data by normalizing each dimension to have a mean of 0 and a standard deviation of 1 during training. To process the provided pose data, we convert them to quaternions to capture the minimal changes between consecutive quaternion pairs. Subsequently, the results are converted back to radians to represent the angular velocity $\dot{\theta}$. This conversion ensures that the angular velocity remains minimal and falls within the range of $[-\pi, \pi]$.

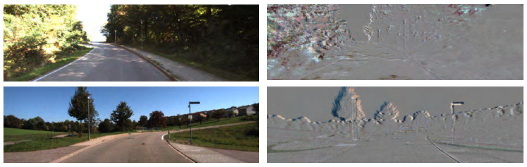

(a) RGB images (b) Difference map

Figure 8: KITTI visual inputs.

Results: The performance of state estimation is evaluated using an 11-fold cross-validation, whereby 1 trajectory is withheld at each time. The standard KITTI benchmark metrics, namely the translational error (m/m) and rotational error (deg/m), are reported in Table 5. The error metrics are computed from the test trajectory over all subsequences of 100 timesteps, as well as all subsequences of 100, 200, 400, and 800 timesteps. Figure 9 presents the performance of α-MDF and other differentiable filtering techniques. It is important to note that incorporating domain- and

Table 5: Result evaluations on KITTI Visual Odometry task measured in m/m and deg/m denote the translational error and the rotational error.

Method	Test 100		Test 100/200/400/800	
	m/m	deg/m	m/m	deg/m
dEKF [9]	0.2646±0.004	0.1386±0.002	0.3159±0.002	0.0923±0.005
DPF [24]	0.1344±0.002	0.1203±0.007	0.2255±0.001	0.0716±0.004
dPF-M-lrn [9]	0.1720±0.010	0.0974±0.009	0.1848±0.004	0.0611±0.003
α-MDF	**0.0718±0.001**	**0.0954±0.001**	**0.0379±0.002**	**0.0328±0.001**

Means±standard errors.

data-specific information, such as using stereo images [45], integrating LiDAR [46, 47], or applying SLAM and loop-closure related assumptions [45, 48], can yield lower error metrics. However, to ensure fair and comparable evaluations, we utilize the most commonly used setup when comparing filtering techniques in a task-agnostic fashion (as performed in [9, 24, 22]).

Table 5 presents the outcomes of our proposed method in comparison with the existing state-of-the-art DFs, namely dEKF, DPF, and dPF-M-lrn. In order to provide a fair comparison, we do not include unstructured LSTM models as baselines since prior works [22, 9] have shown that they do not achieve comparable results. The pre-trained sensor encoder with the same visual inputs is used and integrated into all the DF frameworks evaluated. In this experiment, the motion model of the vehicle is known, and the only unknown part of the state is the velocities. In light of the above, we adopt a learnable process model to update state variables alongside an established motion model to update the (x, y, θ) variables. While the computed Jacobian matrix is supplied in training and testing for dEKF, our α-MDF demonstrates significant improvements compared to dEKF, DPF, and dPF-M-lrn. Specifically, we observed a reduction in the translational error of approximately 88%, 83%, and 79% for Test 100/200/400/800. The results also reflect a considerable reduction in rotational error of approximately 64%, 54%, and 46% as compared to each of the baselines. Our analysis of α-MDF reveals that conducting filtering on high-dimensional observations in the latent space yields better results than conducting filtering on the actual state space.

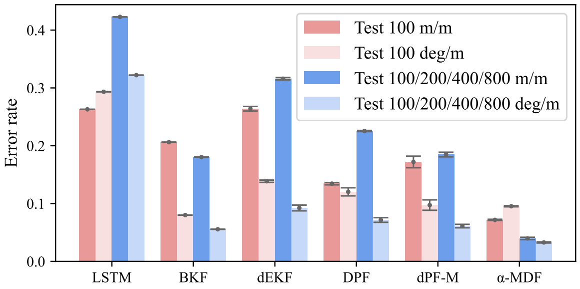
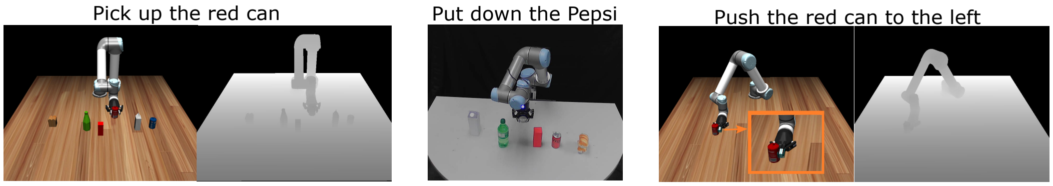


Figure 9: Visual Odometry results with different differentiable filters: the error rate for LSTM and BKF are reported from [22], dEKF, DPF, and dPF-M are reproduced.

B.2 Multimodal Manipulation Tasks

Task Setup: For α-MDF, we define the latent state $\mathbf{x} \in \mathbb{R}^{256}$ for all the manipulation tasks. The actual state of the UR5 robot is described by $\boldsymbol{x}_R$, which consists of the seven joint angles (J_1-J_7) and the Cartesian coordinates (x, y, z) of the robot's end-effector. This Cartesian coordinate system is centered at the manipulation platform's origin point $(0, 0, 0)$. On the other hand, the state of the object being manipulated is represented by $\boldsymbol{x}_O$, which only includes the Cartesian coordinates (x, y, z) of the object. The input modalities for each of the three tasks differ. In task (1), input is given through three modalities: $\mathbf{y}^1$, $\mathbf{y}^2$, and $\mathbf{y}^3$. The first modality $\mathbf{y}^1 \in \mathbb{R}^{224\times224\times3}$ is a camera image captured from a frontal angle. The second modality $\mathbf{y}^2 \in \mathbb{R}^{224\times224\times1}$ depicts depth maps from the same camera view. Lastly, $\mathbf{y}^3$ is a proprioceptive input source with dimensions $\mathbb{R}^7$, representing the joint angles' values. In this task, the proprioceptive input specifically refers to the joint angles as the source. In task (2), input is given by only two modalities: $\mathbf{y}^1$ and $\mathbf{y}^3$, but from a real-world perspective. In task (3), input is received from four modalities: $\mathbf{y}^1$, $\mathbf{y}^2$, $\mathbf{y}^3$, and $\mathbf{y}^4$. $\mathbf{y}^4$ contains the Force/torque (F/T) sensor readings from the robot gripper, where $\mathbf{y}^4 \in \mathbb{R}^6$, while the first two modalities are identical to task (1).



(a) Task 1 (b) Task 2 (c) Task 3

Figure 10: The multimodal manipulation experiment involves the following subtasks: (a) Task 1 utilizing RGB, depth, and joint modalities, (b) Task 2 utilizing only RGB and joint modality, and (c) Task 3 utilizing RGB, depth, joint, and Force/torque (F/T) sensor modalities. The F/T sensor is mounted on the grabber, as depicted by the orange box.

Table 6: Ablation study on UR5 manipulation task with different combination of the modalities.

	RGB	Depth	Joint	F/T	Joint (deg)	EE (cm)	Obj (cm)
Task (1)	✓				2.78±0.09	1.06±0.01	-
		✓			3.65±0.10	1.38±0.05	-
			✓		9.53±0.20	3.22±0.14	-
		✓	✓		2.39±0.11	1.01±0.02	-
	✓		✓		2.69±0.01	1.09±0.03	-
	✓	✓			**1.91±0.08**	**0.64±0.03**	-
	✓	✓	✓		2.19±0.09	0.75±0.01	-
Task (2)	✓				7.49±0.06	3.81±0.17	-
			✓		5.47±0.08	3.32±0.04	-
	✓		✓		**5.24±0.04**	**3.04±0.01**	-
Task (3)		✓	✓	✓	2.93±0.01	2.26±0.02	3.26±0.01
	✓		✓	✓	3.16±0.20	2.34±0.04	3.66±0.30
	✓	✓		✓	1.42±0.08	0.93±0.01	**1.47±0.02**
	✓	✓	✓		**1.37±0.02**	0.94±0.01	1.78±0.06
	✓	✓	✓	✓	1.41±0.04	**0.90±0.01**	1.65±0.01

Means±standard errors.

Data: Data collection is conducted for both simulation with MuJoCo [49] and real-world scenarios. We record the UR5 robot operating on a random object by performing one of "pick", "push", and "put down" actions. We collect 2,000 demonstrations in simulation for task (1), and 100 on the real robot for task (2), with changing the location of each object for each demonstration. For task (3), we collect 2,000 demonstrations in simulation with adding the tactile sensors. We use ABR control and robosuite [50] in addition to MuJoCo to ensure rigorous dynamics in the simulator. Each demonstration sequence has a length of approximately 350 steps with a timestep of 0.08 seconds. An 80/20 data split is utilized for training and testing each task. It should be noted that in all tasks, we normalize the joint modality $\mathbf{y}^3$ and apply Gaussian noise to each joint angle, drawn from the distribution $\mathcal{N} \sim (0, \sigma^2\mathbf{I})$ where $\sigma^2 = 0.1$. We collect the F/T sensor readings directly from MuJoCo's native touch sensor. Moreover, the depth maps obtained from MuJoCo are with no noise therefore can be regarded as high-fidelity data.

Ablation Study: In addition to the findings presented in Section 4.2, we perform a comprehensive ablation analysis for each manipulation task to address the question, "How does the use of multiple modalities compare to a subset of modalities for state estimation with differentiable filters?". Table 6 displays the outcome for each task with various number of modalities using MAE metric. The highest margin of error is indicated by the red shading, while the complete modality is labeled by green shading for each task. Interestingly, even thought using all modalities can generate comparable results, in certain tasks, utilizing all modalities does not necessarily guarantee superior performance compared to utilizing a subset of modalities. Through our experiments in Task (1), it becomes apparent that the optimal performance is achieved by utilizing the subset of modalities $[\mathbf{y}^1, \mathbf{y}^2]$, which yields an improvement of joint angles ($2.19° \rightarrow 1.91°$). In Task (3), we observe that diverse subsets of modalities lead to superior state estimation results for joint angles, EE, and the object locations respectively. Analysis of Table 6 indicates an important role played by the depth map $\mathbf{y}^2$ when considering all observations. This suggests that $\mathbf{y}^2$ is treated as high-fidelity data during training, thereby contributing the most towards the final results.

Henceforth, we conduct an additional ablation analysis to ascertain whether or not the use of a combination of high-fidelity and low-fidelity sensor inputs offers a potential benefit. As noted during data collection, the proprioceptive input $\mathbf{y}^3$ comprising joint angles is obtained via adding Gaussian noise and is therefore considered a low-fidelity input. Figure 11 illustrates the scenario of using $\mathbf{y}^3$ and not using $\mathbf{y}^3$ while applying distinct levels of Gaussian blur in the image and depth space. Notably, without employing $\mathbf{y}^3$, the state estimation performance deteriorates as the level of blur increases. On the other hand, $\mathbf{y}^3$ - despite being classified as a low-fidelity modality - contributes to the final state estimation. In particular, at the highest level of blur, incorporating $\mathbf{y}^3$ yields a 29% improvement in joint angle estimation and a 17% improvement for end-effector locations.

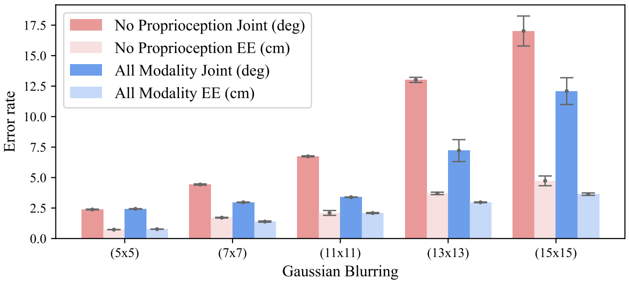


Figure 11: State estimation results are shown after introducing diverse levels of noise to $[\mathbf{y}^1, \mathbf{y}^2]$. The red group depicts results using $[\mathbf{y}^1, \mathbf{y}^2]$ modality, while the blue group represents results using $[\mathbf{y}^1, \mathbf{y}^2, \mathbf{y}^3]$ modality.

B.3 Soft Robot Modeling Tasks

This section presents a comprehensive analysis of the tensegrity robot structure, the bending motion mechanism, and pertinent sensory information, followed by a description of additional experimental outcomes related to this task.

Preliminaries: Our research utilizes a tensegrity robot arm (developed in [39]) that follows a strict tensegrity structure featuring struts, cables (including spring-loaded and actuated cables), and five layers of arm-like tensegrity structures, which produce continuous bending postures when exposed to external forces. The longitudinal length is maintained by stiff cables, while the bending direction is solely determined by external forces. We determine the robot's kinematics through data from Inertial Measurement Units (IMUs), optical motion capture (MoCap), and proportional pressure control valves, with each of the five struts in each layer featuring an IMU. We also record the video by placing a camera in front of the robot while collecting all sensory data. A soft robot's state at t is a 7-dimensional vector $\mathbf{x}_t = [x, y, z, \mathbf{q}_x, \mathbf{q}_y, \mathbf{q}_z, \mathbf{q}_w]^T$, indicating its position and orientation relative to the base frame (layer 1's bottom). $\mathbf{q}$ represents the robot's posture. The system's action is the pressure vector of its 40 pneumatic cylinder actuators ($\mathbf{a}_t \in \mathbb{R}^{40}$). Its raw observation is comprised of 5 IMU readings ($\mathbf{y}_t^3 \in \mathbb{R}^{30}$), with each IMU measuring a 6-dimensional vector of accelerations and angular velocities relative to its location. Fig. 12 illustrates the locations of the IMUs on the struts (blue cubes) in each layer.

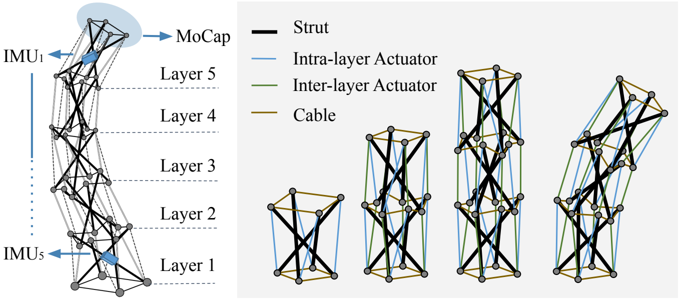


Figure 12: The tensegrity robot features 5 flexible layers, each a tensegrity module with struts, cables, and actuators. Its sensory data includes IMUs, MoCap, and pressure vector readings from pneumatic cylinders.

Data: The complete set of modalities comprises $[\mathbf{y}^1, \mathbf{y}^2, \mathbf{y}^3]$, where $\mathbf{y}^1 \in \mathbb{R}^{224\times224\times3}$ represents RGB images, $\mathbf{y}^2 \in \mathbb{R}^{224\times224}$ is synthetic depth maps which we generate from DPT repo [40] utilizing "Intel/dpt-large", and $\mathbf{y}^3 \in \mathbb{R}^{30}$ is proprioceptive inputs (IMUs). The dataset is generated by performing optical motion capture on the real tensegrity robot hand tip while randomly supplying desired pressure vectors to the pneumatic cylinder actuators. The action $\mathbf{a}_t \in \mathbb{R}^{40}$, 5 IMU readings $\mathbf{y}_t^3 \in \mathbb{R}^{30}$, and a 7-dimensional state $\mathbf{x}_t$ are recorded, with 40-dimensional pressure vectors being

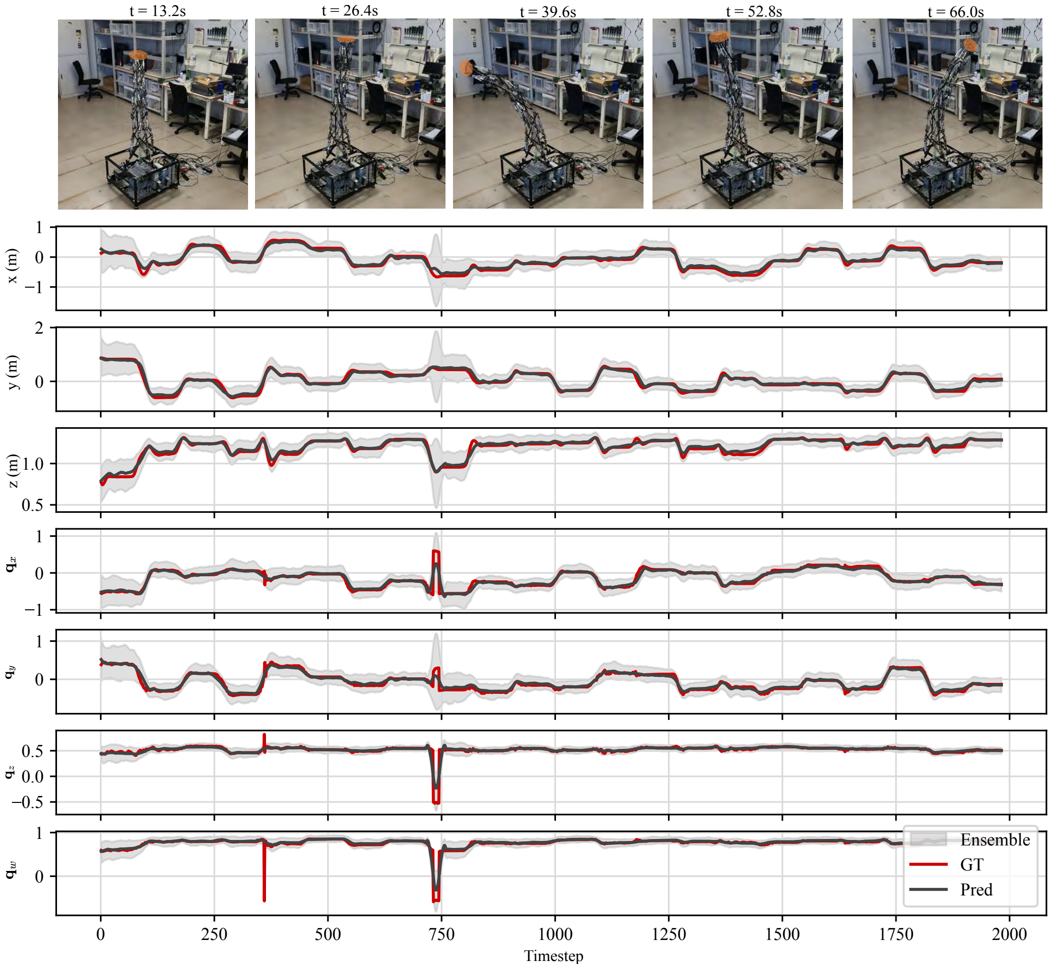


Figure 13: Predicted end-effector (EE) positions and quaternion vectors $\mathbf{q}$ in the soft robot modeling task. The **top** row displays the actual robot posture at the corresponding time, with the orange circle indicating the EE positions, which are not included in the RGB modality input.

used as a control signal. A total of 12,000 trials of robot motion are collected, with each trial involving moving the robot from its current equilibrium posture to the next equilibrium posture by applying the new desired pressure. All data are collected via a ROS2 network with a sampling frequency of 30Hz and are synchronized using the "message_filters" package.

Ablation Study: In addition to the results presented in Section 4.3, we evaluate various combinations of modalities to determine whether an optimal subset of modalities can be identified to attain comparable outcomes without using all modalities during the filtering operation. As demonstrated in Table7, utilizing only one modality fails to achieve comparable results, with the highest accuracy (2.07cm) exclusively from employing $\mathbf{y}^1$ (RGB). The lowest error in posture estimation for the robot is obtained by leveraging $[\mathbf{y}^1, \mathbf{y}^2]$, showing slight improvement (0.10→0.12) over leveraging the full modalities $[\mathbf{y}^1, \mathbf{y}^2, \mathbf{y}^3]$. However, the lowest MAE error for the EE position persists even when all modalities are employed. Interestingly, using solely $\mathbf{y}^3$ re-

Table 7: Ablation study on Tensegrity robot.

RGB	Depth	IMUs	EE (cm)	$\mathbf{q}(10^1)$
✓			2.07±0.03	0.31±0.08
	✓		2.77±0.01	0.19±0.05
		✓	8.99±0.02	0.79±0.03
	✓	✓	2.08±0.03	0.14±0.02
✓		✓	1.73±0.05	0.12±0.02
✓	✓		1.74±0.06	**0.10±0.02**
✓	✓	✓	**1.67±0.09**	0.12±0.01

Means±standard errors.

sults in the highest state estimation error, which aligns with the lowest attention value visualized in Fig 14. As depicted in Fig. 14, it is evident that α-MDF prioritizes $\mathbf{y}^1$ over other modalities. Interestingly, the attention values change upon turning off certain modalities while the system remains stable and functional.

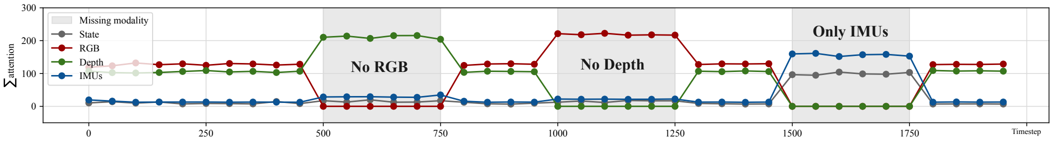


Figure 14: The corresponding accumulated attention values for each modality during testing. The gray areas show certain modalities are selected or not selected.

C Training Details

Table 8 provides an exhaustive enumeration of all learnable modules utilized in α-MDF, which includes three primary components: the state transition model $f_{\boldsymbol{\theta}}$, the sensor encoders $[s^1(\cdot), s^2(\cdot), \cdots, s^M(\cdot)]$, and the attention gain (AG) module. We adopt self-attention layers with dimension 256 and 8 heads, denoted as "Self Attn", in the state transition model. The cross-attention layers, denoted as "Cross Attn", is with dimension 32 and 4 heads in the AG module. The sensor encoders utilized in our approach and all baseline models are identical, with s^1 acting on image-like modalities, utilizing ResNet18 [51] for learning high-dimensional observation representations, while s^2 pertains to low-dimensional modalities such as joint angles. The auxiliary model $\mathcal{A}$ and the decoder $\mathcal{D}$ shares a similar structure to s^2, but with different number of neurons. Note that x is the dimension of the actual state.

Table 8: α-MDF's learnable sub-modules.

$f_{\boldsymbol{\theta}}$:	3× SNN(256, ReLu), Positional Embedding, 3× Self Attn(256,8), 2× SNN(256, ReLu), 1× SNN(d_x, -)
s^1:	1× ResNet18(h,w,ch), 2× fc(2048, ReLu), 1× SNN(512, ReLu), 1× SNN(d_x, -)
s^2:	1× SNN(128, ReLu), 1× SNN(256, ReLu), 1× SNN(512, ReLu), 1× SNN(d_x, -)
AG:	Positional Embedding, 1× Cross Attn(32, 4, mask)
$\mathcal{A}$:	1× SNN(128, ReLu), 1× SNN(256, ReLu), 1× SNN(512, ReLu), 1× SNN(1024, ReLu), 1× SNN(d_x, -)
$\mathcal{D}$:	1× fc(256, ReLu), 1× SNN(128, ReLu), 1× SNN(32, ReLu), 1× SNN(x, -)

fc: Fully Connected, SNN: Stochastic Neural network.

During α-MDF training, we employ the curriculum outlined in Algorithm 1. Note that some tasks may require pre-training the sensor encoders before performing end-to-end training the entire framework. For each task, we train α-MDF model with utilizing batch size of 64 on a single NVIDIA A100 GPU for roughly 48 hours. For all the tasks, we use the Adamw [52] optimizer with a learning rate of 1e-4.

Algorithm 1 Condition in Latent Space: training algorithm return the weights ω

```
Input: $\alpha$-MDF, dataloader $\left(\{\boldsymbol{x}_t\}_{t-N}^{t+1}, \{\mathbf{y}_t^m\}_{m=1}^M, \{\mathbf{y}_{t+1}^m\}_{m=1}^M, \{\mathbf{a}_t\}_{t-1}^{t+1}\right)$
Output: weights $\omega$
while not converged do
    Call dataloader with a random timestep $t$.
    for timestep $t \leftarrow t$ to $t+1$ do
        $e_1 \leftarrow \sum_{m=1}^M \|\mathcal{D}(s^m(\mathbf{y}_t^m)) - \boldsymbol{x}_t\|_2^2$ according to Eq. 6
        $e_2 \leftarrow \mathcal{L}_{f_{\boldsymbol{\theta}}}(\mathbf{X}_t) + \mathcal{L}_{e2e}(\hat{\mathbf{X}}_t)$ according to Eq. 6
        $e_t \leftarrow e_1 + e_2$
    end for
    $\omega \leftarrow$ Train $(\alpha$-MDF$, e_t + e_{t+1})$
end while
return $\omega$
```