Multimodal Sensor Fusion with Differentiable Filters

Michelle A. Lee*, Brent Yi*, Roberto Martín-Martín, Silvio Savarese, Jeannette Bohg

Abstract— Leveraging multimodal information with recursive Bayesian filters improves performance and robustness of state estimation, as recursive filters can combine different modalities according to their uncertainties. Prior work has studied how to optimally fuse different sensor modalities with analytical state estimation algorithms. However, deriving the dynamics and measurement models along with their noise profile can be difficult or lead to intractable models. Differentiable filters provide a way to learn these models end-to-end while retaining the algorithmic structure of recursive filters. This can be especially helpful when working with sensor modalities that are high dimensional and have very different characteristics. In contact-rich manipulation, we want to combine visual sensing (which gives us global information) with tactile sensing (which gives us local information). In this paper, we study new differentiable filtering architectures to fuse heterogeneous sensor information. As case studies, we evaluate three tasks: two in planar pushing (simulated and real) and one in manipulating a kinematically constrained door (simulated). In extensive evaluations, we find that differentiable filters that leverage crossmodal sensor information reach comparable accuracies to unstructured LSTM models, while presenting interpretability benefits that may be important for safety-critical systems. We also release an open-source library for creating and training differentiable Bayesian filters in PyTorch, which can be found on our project website: `https://sites.google.com/view/multimodalfilter`.

I. INTRODUCTION

Object manipulation in robotics is inherently multimodal: a robot can observe the object interaction with its cameras, register the motion of its own motors, and sense the forces and torques it applies with its end effector. The information from these modalities can be used to continuously estimate the state of those elements in the environment that are relevant to the manipulation task. Since the information from vision, proprioception, and haptics is complementary, using multimodal sensory inputs can help overcome limitations of single modalities due to occlusions, loss of contact, sensor failure, or differing sensor rates. Fusing these modalities in an optimal manner promises more accurate, robust and reliable manipulation. This idea has fueled research in sensor fusion and multimodal state estimation for decades [2, 7, 21].

Multimodal state estimation for manipulation is challenging. Visual, proprioceptive and haptic data live in very different sensor spaces that represent different physical properties,
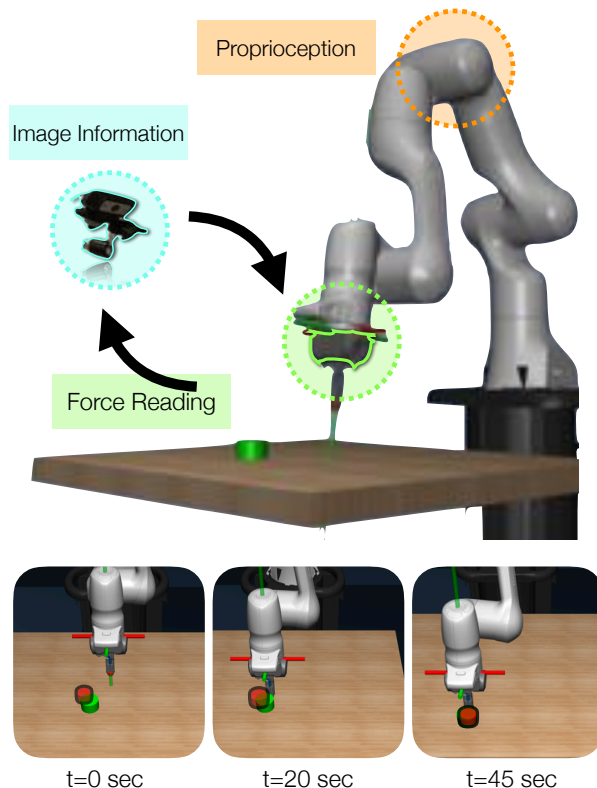
Fig. 1: We study the integration of modalities for recursive state estimation with differentiable filters; (top) Information from force sensing (local information), joint encoders and visual sensing (global information) is fused with different architectures, including new crossmodal strategies that use information from one modality to assess the uncertainty of another; (bottom) By combining local and global information during manipulation, the pose of an object (green peg) can be correctly estimated (red overlay).

have disparate dimensionality, and may arrive at different rates. Moreover, the models that map these signals to the task-relevant state (e.g. object motion) are often complex and require full knowledge of the environment. To alleviate the complexity of deriving these models, recently proposed methods [8, 15, 19] learn the measurement, forward, and noise models from labeled data. To apply deep neural networks to this learning task, these methods turn the state estimation process into a differentiable procedure that allows end-to-end backpropagation of errors. While these methods demonstrate improved performance and are more interpretable when compared to completely unstructured LSTMs, this was only shown with a single sensor modality or with sensors with very similar characteristics, such as RGB images and depth.

To fuse sensor information with different characteristics, an estimator needs to understand how to balance confidence between modalities. We present three types of fusion mechanisms for differentiable filters. The first one fuses data based

on uncertainty estimates per modality, i.e. unimodal information. The second ones concatenates unimodal features into a multimodal representations. And a third one uses crossmodal information to weight unimodal state estimates from multiple sensors. We integrate the proposed fusion mechanisms with parametric and non-parametric recursive state estimators. We compare these multimodal differentiable filters with an unstructured LSTM and evaluate their performance for clean simulated data, noisy simulated data, and real data.

Our study shows that differentiable filters, when compared to more expressive LSTM models, are able to provide important interpretability benefits without sacrificing performance. In our comparison of fusion mechanisms, we demonstrate that filters that only leverage unimodal information can be brittle and perform inconsistently across tasks. For filters that leverage crossmodal weights, we demonstrate additional opportunities for interpretability. By analyzing the outputs of individual components within the proposed crossmodal architectures, we show that these filters allow us to inspect the contributions of individual modalities, while also assessing when information from each modality can be trusted.

II. RELATED WORK

A. Multimodal Fusion

Traditionally, sensor fusion is performed within a recursive filter by individually modelling each observation and then integrating the resulting information into a common state estimate [1, 11, 14, 27]. These methods support the intuition that an optimal combination of information from different modalities improves the overall estimation accuracy. The general schema of these approaches includes a Bayesian filter (see Sec. III) and a multimodal measurement model that quantifies per modality how likely the current measurement is given the predicted state. Other approaches use measurements in a crossmodal manner where information from one sensor modality helps to interpret another [6, 25, 26]. Each of these methods require the user to identify and define analytical forward and measurement models that may be hard to specify for some dynamical systems or intractable to compute online. In this paper, we reduce the need for predefined models for multimodal and crossmodal fusion by learning both the measurement and the forward model from annotated data with neural networks.

When using probabilistic recursive state estimation with observations from multiple sensors, the information of the modalities can only be optimally integrated if the uncertainty of each sensor has been correctly characterized [3, 28]. Traditionally, these uncertainties have to be hand-tuned so that the posterior state estimate is close to ground truth. In this paper, we learn the uncertainty of each sensor modality, thereby alleviating handtuning.

Instead of fusing sensors recursively within the filters, Caron et al. proposed to weight state estimates from uni- and multimodal Kalman filters [4] and to switch between measurement models of uni- and multimodal particle filters [5]. However, the weighting mechanisms depend on user-defined and hand-tuned thresholds for each sensor. In our work, we propose and evaluate fusion models that learn to merge state estimates from single modality filters by leveraging crossmodal information.

[22, 31] have proposed a multimodal state estimation framework for planar pushing (similar to two of the three case studies we use in this paper) with factor graphs (iSAM [16]), however they used fiducials for tracking instead of raw RGB data. While the authors show that iSAM provides more accurate and robust state estimates, it still requires careful specification of modality-specific cost functions as well as forward and measurement models, which we learn from data.

B. Differentiable filters

Differentiable filters provide an approach for learning forward and measurement models from data while retaining the algorithmic structure of a recursive Bayes filter. Differentiable filters can thus be advantageous for systems whose dynamics and sensor observations are hard to model analytically, while making it possible to retain the interpretability of state representation and uncertainty that is often vital for safety-critical systems. Jonschkowski et al. [15] and Karkus et al. [18] each proposed differentiable versions of the particle filter [28], as applied to simple localization tasks. Their methods learn an estimator that shows improved results over an unconstrained LSTM. Similarly, Haarnoja et al. [8] proposed a differentiable version of the EKF. The authors reach similar results after testing on toy visual tracking and real-world localization tasks. In addition, other lines of research have proposed to use recurrent neural networks to learn latent state representations [9, 13, 23, 24]. All methods above explored how to combine filtering techniques with learning approaches, but they do not systematically study how to fuse information from different sensor modalities. In this paper, we investigate new differentiable filtering architectures to fuse information from the heterogenous sensor modalities of vision, touch, and proprioception.

III. BACKGROUND ON FILTERING

We consider the problem of estimating the state $\mathbf{x}$ of a system from a sequence of (multimodal or unimodal) observations $\mathbf{z}$ and control inputs $\mathbf{u}$. We represent our knowledge and uncertainty about our estimate with a distribution over the current state $\mathbf{x}_t$ conditioned on all the previous observations $\mathbf{z}_{1:t}$ and control inputs $\mathbf{u}_{1:t}$. We denote this distributions as belief $bel(\mathbf{x}_t) = p(\mathbf{x}_t \mid \mathbf{u}_{1:t}, \mathbf{z}_{1:t})$. One solution to compute this belief are Bayes filters.

Traditionally, developing a Bayes filter requires analytically formulating a forward model to predict the next state based on current state and control input, $p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_{t-1})$, a measurement model, $p(\mathbf{z}_t \mid \mathbf{x}_t)$, to compute the likelihood of an observation given a state, and the noise associated with predictions and observations. Formulating these models and quantifying the noise often requires making strong assumptions on the properties of the underlying system. A way to avoid making these assumptions is to extract this information directly from labeled data of observations and ground truth states. This can be achieved with the recent

family of differentiable Bayes filters. Here, we study differentiable filters for the special case of fusing heterogeneous sensory data from multiple modalities. In this section, we will first give a brief summary of Bayes filters before we provide a unified notation for differentiable filters. For a more in-depth discussion of Bayes filters, we refer to [28].

A. Bayes Filters

The Bayes filter algorithm provides an optimal solution for state estimation in a system that follows the Markov assumption and in which observations are conditionally independent. In this filter, the belief is updated in two steps. In the prediction step, a motion model $p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_{t-1})$ is used to predict the belief $\overline{bel}(\mathbf{x}_t)$ about the current state $\mathbf{x}_t$ given the previous state $\mathbf{x}_{t-1}$ and control input $\mathbf{u}_{t-1}$

$$\overline{bel}(\mathbf{x}_t) = \int p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_{t-1}) bel(\mathbf{x}_{t-1}) d\mathbf{x}_{t-1} \qquad (1)$$

In the update step, we correct this initial prediction given a sensory observation $\mathbf{z}_t$ plugged into a measurement model $p(\mathbf{z}_t \mid \mathbf{x}_t)$ that describes the likelihood of this observation given the predicted state:

$$bel(\mathbf{x}_t) = \eta \, p(\mathbf{z}_t \mid \mathbf{x}_t) \overline{bel}(\mathbf{x}_t) \qquad (2)$$

where η is a normalization factor.

The Kalman filter [17] is the optimal estimator for a system with linear models and Gaussian noise. In robotics, however, most systems of interest have non-linear models and may follow a more complex non-Gaussian distribution. For these cases, there are non-linear Bayes filters that make different approximations to estimate the belief $bel(\mathbf{x}_t)$.

Of these, the most widely used include the Extended Kalman Filter (EKF) and the Particle Filter [28]. The EKF is used when the involved system models are not linear but linearizable through a Taylor expansion. Researchers resort to particle filters when the non-linearities of the system cannot be linearized or when the underlying state distribution cannot assumed to be Gaussian, e.g. when distributions have multiple peaks. In this work we will study how to fuse information from multiple sensor modalities using differentiable versions of the EKF and particle filter.

B. Differentiable Filters

For complex physical systems, it is often a challenge to formulate a dynamics and observation model that is both accurate and tractable to compute. Recently, differentiable versions of the most popular non-linear filtering algorithms have been proposed such that dynamics and observation models along with their noise parameters can be learned.

1) Extended Kalman Filter: The EKF allows non-linear forward and observation models but still assumes that the state follows a Gaussian distribution such that $bel(\mathbf{x}_t) \sim N(\mu_t, \Sigma_t)$. Specifically, we assume that the system dynamics and measurements follow the following nonlinear functions:

$$\mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{u}_{t-1}, \mathbf{q}_t) \qquad (3)$$
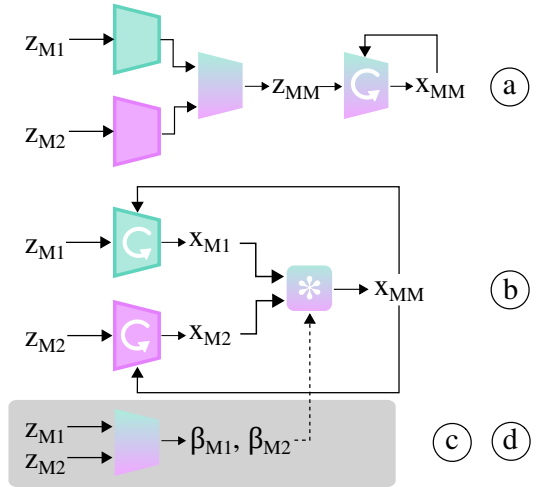$$\mathbf{z}_t = h(\mathbf{x}_t, \mathbf{r}_t) \qquad (4)$$

Fig. 2: Multimodal Architectures: **a) Fusion of Features**: modalities are processed and merged into a multimodal feature for state estimation; **b) Unimodal Weighted Fusion**: estimates from two unimodal recursive filters are merged based on their uncertainty; c) and d) **Crossmodal Weighted Fusion**: both modalities are used to learn coefficients to fuse the unimodal estimates; the coefficients are used for **c) weighted average of states** or **d) weighted measurement model**

where the random variables $\mathbf{q}$ and $\mathbf{r}$ are the process and observation noise. In the EKF, the prediction step is as follows

$$\hat{\mu}_t = f(\mu_{t-1}, \mathbf{u}_{t-1}, 0) \qquad (5)$$
$$\hat{\Sigma}_t = \mathbf{A}_{t-1} \Sigma_{t-1} \mathbf{A}_{t-1}^T + \mathbf{Q}_{t-1} \qquad (6)$$

where $\mathbf{A}_t$ is the Jacobian $\frac{\partial f(\mu_t, \mathbf{u}_t, 0)}{\partial \mu_t}$ and $\mathbf{Q}$ the covariance of the process noise $\mathbf{q} \sim N(0, \mathbf{Q})$ which we assume to be Gaussian with zero mean. Therefore, $\overline{bel}(\mathbf{x}_t) \sim N(\hat{\mu}_t, \hat{\Sigma}_t)$.

The update step is as follows:

$$\mathbf{K}_t = \hat{\Sigma}_t (\mathbf{H}_t^T \mathbf{H}_t \hat{\Sigma}_t \mathbf{H}_t^T + \mathbf{R}_t) \qquad (7)$$
$$\mu_t = \hat{\mu}_t + \mathbf{K}_t (\mathbf{z}_t - \mathbf{H}_t \hat{\mu}_t) \qquad (8)$$
$$\Sigma_t = (\mathbf{I}_n - \mathbf{K}_t \mathbf{H}_t) \hat{\Sigma}_t \qquad (9)$$

where $\mathbf{H}_t$ is the Jacobian $\frac{\partial h(\mu_t, 0)}{\partial \mu_t}$ and $\mathbf{R}$ the covariance of the measurement noise $\mathbf{r} \sim N(0, \mathbf{R})$ which we assume to be Gaussian with zero mean.

For a differentiable version of the EKF, we implement the dynamics model in Eq. 5 with a *Multi-Layer Perceptron* (MLP) with trainable weights θ: $f_\theta(\mu_{t-1}, \mathbf{u}_{t-1}, 0)$. Learning a true measurement model that maps states to expected observations in our raw sensor space is an underdetermined problem and prone to overfitting. We follow [8] and instead learn a discriminative virtual sensor. Our virtual sensor $g_\theta(\mathbf{D})$ uses as input raw sensory data $\mathbf{D}$ and outputs a vector $\mathbf{z}_t$ containing observations of the full state $\mathbf{x}$ or parts of the state. Therefore, the measurement model and its Jacobian $\mathbf{H}$ are either identity or selection matrices. Similar to the dynamics model, we implement the virtual sensor $g_\theta(\mathbf{D})$ with an MLP with trainable weights θ.

2) Particle Filter: While the EKF assumes a Gaussian distribution for the underlying distribution of the system state, a particle filter can model arbitrary distributions by representing them with a set of particles. Concretely, we will learn a measurement model in the form of a neural network

that outputs the log-likelihood of the observation predicted for each particle. Specifically, we approximate the belief with a set of particles $\mathcal{X}_t = \mathbf{x}_t^{[1]}, \mathbf{x}_t^{[2]}, \cdots, \mathbf{x}_t^{[n]}$ with weights $w_t^{[1]}, w_t^{[2]}, \cdots, w_t^{[n]}$. Like any Bayes filter, this non-parametric filter has a predict and update step. The prediction step in Eq. 1 is implemented by sampling a random perturbation for each particle from a generative motion model:

$$\forall_i : \mathbf{x}_t^{[i]} \sim p(\mathbf{x}_t \mid \mathbf{u}_{t-1}, \mathbf{x}_{t-1}^{[i]}) \tag{10}$$

The update step in Eq. 2 is implemented by setting the weight $w_t^{[i]}$ of each particle equal to the likelihood of the current measurement $\mathbf{z}_t$ being generated by the predicted state $\mathbf{x}_t^{[i]}$ of this particle:

$$\forall_i : w_t^{[i]} \sim p(\mathbf{z}_t \mid \mathbf{x}_t^{[i]}) \tag{11}$$

The particle set is then re-sampled proportionally to the weight of each particle. Note that a virtual sensor is needed only for the EKF and not the particle filters. For the particle filters, $\mathbf{z}_t$ equals the raw sensory data $\mathbf{D}$ without any processing.

3) LSTM: A recurrent filter resembles the structure of a long-short term memory: an internal state that is recurrently merged with the latest input signal to generate an updated output. Therefore, we consider LSTM architectures as a baseline to differentiable filters, which we call `LSTM Baseline`. In contrast to a differentiable filter, there is no explicit separation between prediction and update steps in an LSTM. There are also no explicit measurement or forward models. An LSTM also does not make the Markov Assumption. Previous studies [8, 15, 18] demonstrated that differentiable architectures that leverage the algorithmic structure of a Bayes filter lead to faster learning and better generalization than generic LSTM architectures.

IV. ARCHITECTURES FOR MULTIMODAL FUSION

We study different strategies to integrate the information from multiple sensor modalities into a coherent state estimate. We now describe these strategies as visualized in Fig. 2.

A. Feature Fusion

One strategy to fuse information from multiple modalities in a differentiable filter is to first extract features from each modality separately, and then estimate recursively the state using the fused unimodal features (Fig. 2, a). This is achieved by a separate encoder network for each modality, and a fully connected network to integrate the unimodal features into a multimodal one. The multimodal feature is then used as an observation in the recurrent filter architecture. We call our filters `Feature Fusion EKF` and `Feature Fusion PF`.

B. Unimodal Weighted Fusion

This architecture (Fig. 2, b) is a fusing procedure for multiple state estimates that are each normally distributed. In this case, we assume we have two unimodal EKF filters that provide independent state estimates $bel(\mathbf{x}_t^{M_1}) \sim \mathcal{N}(\mu_t^{M_1}, \Sigma_t^{M_1})$ and $bel(\mathbf{x}_t^{M_2}) \sim \mathcal{N}(\mu_t^{M_2}, \Sigma_t^{M_2})$. We fuse the two unimodal

beliefs by multiplying the two distributions to produce a normally distributed multimodal belief $bel(\mathbf{x}_t^{MM})$:

$$bel(\mathbf{x}_t^{MM}) = \mathcal{N}(\mu_t^{MM}, \Sigma_t^{MM}) \tag{1}$$

$$\mu_t^{MM} = \frac{(\Sigma_t^{M_1})^{-1}\mu_t^{M_1} + (\Sigma_t^{M_2})^{-1}\mu_t^{M_2}}{(\Sigma_t^{M_1})^{-1} + (\Sigma_t^{M_2})^{-1}} \tag{2}$$

$$\Sigma_t^{MM} = ((\Sigma_t^{M_1})^{-1} + (\Sigma_t^{M_2})^{-1})^{-1} \tag{3}$$

This product of two Gaussians is equivalent to the Product of Experts [12]. In this architecture, there is no crossmodal information flow since the modalities are assumed independent of each other: the information from one does not help estimating or assessing the uncertainty of the other. We refer to the resulting model as `Unimodal Fusion EKF`.

C. Crossmodal Weighted Fusion

In this architecture, we also assume that there are unimodal filters providing individual state estimates. Similar to the `Unimodal Fusion EKF`, we assume that the estimates are normally distributed and independent. The integration of their estimates is goverened by the coefficients $\beta_t^{M_1}$ and $\beta_t^{M_2}$, which weight the contributions of each unimodal value (estimated from observations of sensors $M1$ or $M2$) into a fused estimate. Each coefficient is inferred from the information contained in the multimodal signal. In this way, information from one modality is used to assess the uncertainty about the other and vice versa, creating a crossmodal information flow. This architecture assumes that the final and each unimodal belief can be faithfully represented by a Gaussian. We apply this fusion architecture to integrate information from several differentiable Extended Kalman filters and we call the resulting model `Crossmodal Fusion EKF`.

In our implementation, $\vec{\beta}_t^{M_i}$ consists of elements $\beta_{t,1}^{M_i}, \beta_{t,2}^{M_i}, \ldots \beta_{t,n}^{M_i}$, where n is the dimensionality of the state space. The weighting of the covariance matrix is done through a positive semi-definite matrix $\mathbf{B}_t^{M_i} \in \mathbb{R}^{n \times n}$, defined as:

$$\mathbf{B}_t^{M_i} = \begin{bmatrix} \beta_{t,1}^{M_i} & \cdots & \beta_{t,n}^{M_i} \\ \vdots & & \vdots \\ \beta_{t,1}^{M_i} & \cdots & \beta_{t,n}^{M_i} \end{bmatrix}^T \begin{bmatrix} \beta_{t,1}^{M_i} & \cdots & \beta_{t,n}^{M_i} \\ \vdots & & \vdots \\ \beta_{t,1}^{M_i} & \cdots & \beta_{t,n}^{M_i} \end{bmatrix}.$$

The integration is then a weighted average as follows:

$$\mu_t^{MM} = \frac{\vec{\beta}_t^{M_1} \odot \mu_t^{M_1} + \vec{\beta}_t^{M_2} \odot \mu_t^{M_2}}{\vec{\beta}^{M_1} + \vec{\beta}_t^{M_2}} \tag{4}$$

$$\Sigma_t^{MM} = \frac{\mathbf{B}_t^{M_1} \odot \Sigma_t^{M_1} + \mathbf{B}_t^{M_2} \odot \Sigma_t^{M_2}}{\mathbf{B}_t^{M_1} + \mathbf{B}_t^{M_2}} \tag{5}$$

Note that in this case, the superscript M_1 and M_2 does not indicate the originating modality but the estimated state that it is applied to.

D. Unimodal Weighted Measurement Models

In this architecture, we assume unimodal measurement models $p(\mathbf{z}_t \mid \mathbf{x}_t)^{M_1}$ and $p(\mathbf{z}_t \mid \mathbf{x}_t)^{M_2}$ that provide unimodal weights $w_t^{M_1,[i]}$ and $w_t^{M_2,[i]}$ for each particle. Unlike in Section

IV-B, where we combine estimated states from each filter, we instead combine the likelihoods that are the outputs of each measurement model. One way to do this is with a mixture model, where we compute our particle weights as the sum of our unimodal weights:

$$w_t^{[i]} = w_t^{M1,[i]} + w_t^{M2,[i]} \qquad (6)$$

Using the standard particle filter dynamics update and resampling procedures, we can apply this architecture to integrate estimates from a particle filter with a single dynamics model and multiple unimodal measurement models. We call the resulting model `Unimodal Fusion PF`.

E. Measurement Models with Learned Crossmodal Weights

Finally, we can also add a crossmodal information flow to the fusion of our unimodal measurement models assumed in Section IV-D. We do this by learning a mixture weighting model that generates non-negative scalar coefficients β_t^{M1} and β_t^{M2} from the full multimodal observation input. This is a differentiable generalization of the discrete measurement model switching method employed by [5] for fusing global and local sensor information in a land vehicle positioning problem. The weight of each particle is therefore set to:

$$w_t^{[i]} = \beta_t^{M1} * w_t^{M1,[i]} + \beta_t^{M2} * w_t^{M2,[i]} \qquad (7)$$

We call the resulting model `Crossmodal Fusion PF`.

V. Implementation

A. Network Architectures

For the differentiable EKF and differentiable PF estimators, we use a shared **dynamics model** with terms parameterized by trainable weights θ:

$$\mathbf{x}_t = \mathbf{x}_{t-1} + f_{1,\theta}(\mathbf{x}_{t-1}, \mathbf{u}_{t-1}) \cdot \sigma(f_{2,\theta}(\mathbf{x}_{t-1}, \mathbf{u}_{t-1})) \qquad (1)$$

where $f_{1,\theta}$ is a state update vector of the same dimension as $\mathbf{x}_t$, $f_{2,\theta} \in \mathbb{R}$ is a scalar gating/scaling term, and σ is the sigmoid function $\sigma(z) = (1 + e^{-z})^{-1}$. Separating the relative state update into these two terms allows the network to independently learn the direction and magnitude of the state update. For the particle filter, we maintain the diversity of particles by injecting additive Gaussian noise in our state space after each dynamics udpate.

The network that outputs $f_{1,\theta}$ and $f_{2,\theta}$ feeds the inputs $\mathbf{x}_{t-1}$ and $\mathbf{u}_{t-1}$ through three-layer encoders. The outputs are concatenated and passed to a seven-layer set of shared layers.

The **measurement models** of the differentiable PFs are trained using as inputs observations (images, F/T signals and/or end-effector positions) and states (object position on the table) as stored in each particle. The output of the measurement model is the log-likelihood of these observations given the state of each particle. Image inputs are encoded with a set of 2D convolutions, while proprioception and haptics are processed with standard fully-connected layers. The encoded features are concatenated, and then fed to a shared series of output layers.

The **virtual sensor model** architectures are nearly identical to our particle filter measurement models, but instead of outputting a log likelihood, the networks output an estimated state and state covariance for our EKF.
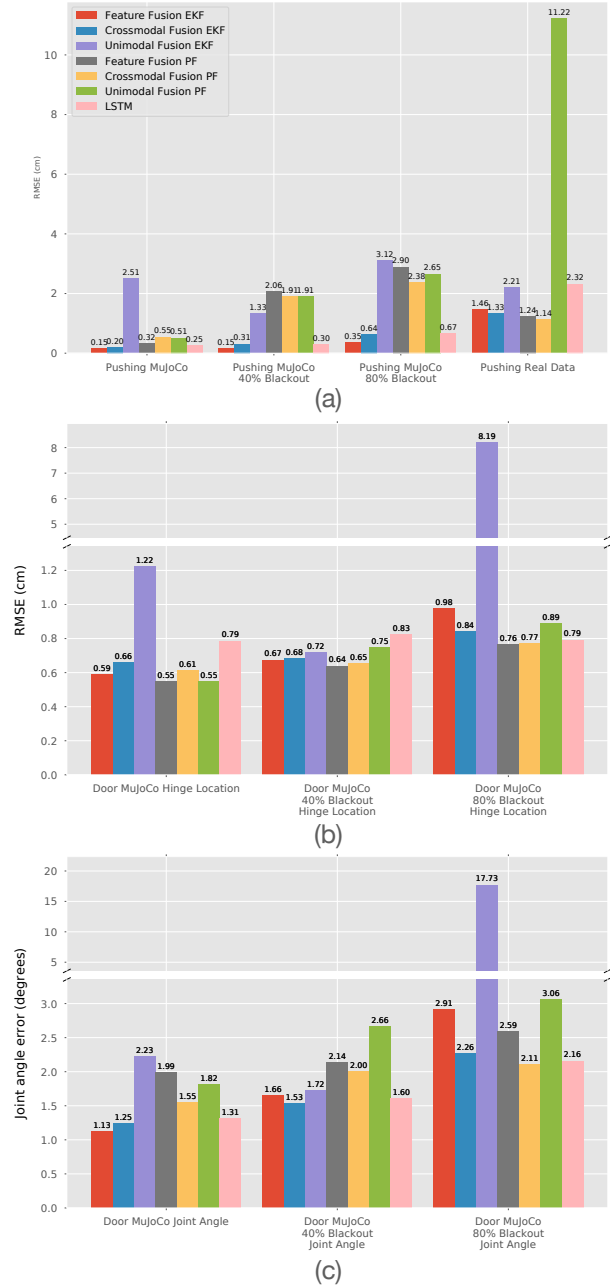

Fig. 3: We show the results of our tasks: pushing and door manipulation. In (a), RMSE error in centimeters for each filter in our validation set for MuJoCo pushing simulation data, MuJoCo pushing simulation data with 40% and 80% image blackout, and real world pushing. In (b) and (c) respectively, RMSE error in centimeters and joint angle error in degrees for each filter in our validation set for the MuJoCo door simulation data and MuJoCo simulation data with 40% and 80% image blackout.

In the **crossmodal weight models**, the inputs are also observations (images, F/T signals, and/or end-effector positions), and the outputs are the learned crossmodal fusion coefficients used to balance estimates from each modality. Each modality is run through an encoder with the same architecture as our

measurement/virtual sensor models, and concatenated before being fed to the output layers that produce our final weights.

The `LSTM Baseline model` is designed to have a similar architecture and parameter count as the rest of our estimators, with the same encoder architectures for each modality and control inputs. Encoded features from each modality and action inputs are concatenated and passed through a series of shared layers; in contrast to our crossmodal weight models, these layers are terminated with two LSTM layers. The output of the final LSTM layer is then mapped through an additional fully-connected layer before an affine mapping is applied to produce the state estimate.

We use ReLU activations and ResNet-style skip connections [10] throughout all of our networks.

To ensure that comparisons are fair, we ran an architecture search for each model. For the image encoders, we tried three architecture variations for mapping outputs of our 2D convolutions to our fully-connected layers: (a) a simple flatten operation, (b) spatial softmaxes, and (c) mean pooling. We found the simple architecture (a) to perform well for models that outputted scalar weights and log-likelihoods, while option (c) generalized best for networks that directly regressed XY coordinates. We additionally explored varying network widths and LSTM hidden state dimensions; final results are reported using 64 units for all fully-connected layers and a hidden state size of 512 for each LSTM layer.

B. Training Procedure

We first pre-train the dynamics and measurement models before fine-tuning them in an end-to-end manner. The **dynamics models** (Sec. III-B) are pre-trained to minimize single-step prediction errors, and then for 4, 8, and finally 16-step prediction errors.

The **particle filter measurement models** are pre-trained to predict the observation-conditioned log probability density function of a multivariate Gaussian centered at the ground-truth state for our particle filter. Likewise, the **EKF virtual sensor models** are pre-trained to generate observation-conditioned predicted states. Measurement uncertainties are learned exclusively end-to-end, and not pre-trained.

After pre-training, we use backpropagation through time to train each of our state estimation models end-to-end until convergence, over subsequences of increasing length. 30 particles are used for training the particle filter. Optimization was done using the Adam optimizer on NVIDIA Quadro P4000, Tesla K80, and Tesla V100 GPUs.

VI. Experimental Evaluation

We study the fusion architectures described in Sec. IV to estimate the state in three experimental scenarios: pushing a planar object in simulation, pushing a planar object in the real world, and opening a door in simulation. The simulation experiments allow us to evaluate the fusion architectures in a controlled and noise-free setup, while the real data evaluation allows us to test the robustness of the fusion techniques in a realistic domain. In both simulation and the real world, we first generate a dataset of sensor data and ground truth states,

use part of the dataset to train our differentiable models and a different part to test the resulting models.

A. Experimental Setup

In the planar pushing tasks, we estimate the 2D position of the unknown object on a table surface, $\mathbf{x}_t = (x_t, y_t)$, while the robot intermittently interacts with the object. In the door manipulation task, we estimate the 2D position of the door's revolute joint and its joint angle, $\mathbf{x}_t = (x_t, y_t, \theta_t)$, while the door is being opened. In these scenarios, we compare the performance of the aforementioned fusion architectures when using as input vision (images), haptics (force-torque sensor readings), and proprioception (encoder values we use to infer the end-effector pose). For each of our *Crossmodal Fusion* and *Unimodal Fusion* filters, we consider the image as one weighted modality and force/proprioception as another.

For the simulation experiments, we collect data with MuJoCo [29]. We create a Mujoco pushing dataset consisting of 1000 trajectories with 250 steps at 10 Hz, of a simulated Franka Panda robot arm pushing a circular puck. The pushing actions are generated by a heuristic controller that tries to move the end-effector to the center of the object.

For the evaluation with real-world data, we use the three ellipse shapes from the MIT pushing dataset [30], which consists of more than a million real robot push sequences on eleven different objects with four different surface materials. We use the tools described in Kloss et al. [20] to obtain additional annotations for the remaining state component, for rendering gray-scale images, and for stitching together 1000 pushing trajectories with 45 steps at 18 Hz.

Finally, we collect a dataset for the door task in Mujoco that consists of 600 trajectories with 800 steps at 10 Hz each, of the Franka Panda robot pushing and pulling a kinematically constrained door object. Actions are generated by a heuristic controller that pushes and pulls on the door in random directions. Half of the trajectories are recorded with the gripper of the robot closed around the handle of the door; the other half is recorded without directly grabbing the handle.

In all experiments, the multimodal inputs are gray-scaled images (1 x 32 x 32) from an RGB camera, forces (and binary contact information) from a force/torque sensor, and the 3D position of the robot end-effector. While wrench signals in the real-world dataset only have 3 dimensions, force in the x and y direction, and torque in the z direction (F_x, F_y, and τ_z), we use all six dimensions of force and torque in the simulation dataset.

To see how each filter adjusts to sensor failure and noise, we randomly black out the input images with probabilities of 0.4 and 0.8 in the simulation datasets for pushing and the door task. For evaluating each filter, we split the trajectories into train and test sets. We use 10 trajectories from the simulated pushing task, 50 trajectories from the real-world data pushing task, and 20 trajectories for the door opening task.

B. Experimental Results

1) Pushing Task: In the pushing task, we evaluate each filter by comparing the root mean squared error (RMSE) of
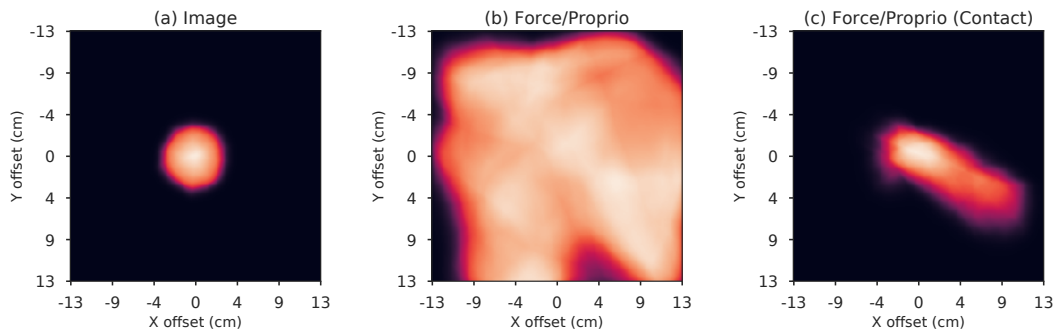
Fig. 4: Examining outputs from unimodal measurement models within the proposed fusion architectures can help us understand how our neural networks are fusing each sensing modality. Using a `Crossmodal Fusion PF` model trained end-to-end on the `Mujoco door dataset`, we plot how the likelihood of a state changes as a function of positional offsets from the ground-truth door hinge position: **(a)** using the unimodal image measurement model, **(b)** using the force/proprioception model on a timestep before the robot has made contact with the door, and **(c)** using the force/proprioception model after the robot gripper has made contact with the door.

the estimated X and Y location of the pushed object. We show the results of each filter in Fig. 3.

Notably, we found that the `Unimodal Fusion` filters were brittle and inconsistent. The `Unimodal Fusion EKF` performs worse on the blackout-free dataset than on the dataset with 40% blackout images, and `Unimodal Fusion PF` had an average of 11cm error on the real pushing dataset.

On the other hand, all models that utilized crossmodal information (the `Crossmodal Fusion` filters, `Feature Fusion` filters, and `LSTM`) were able to learn how to estimate the location of the object with less than 3cm of error in the simulated dataset and less than 2.5cm on the real dataset.

While the EKF and LSTM estimators perform better than the particle filters on our simulation dataset, we notice that this gap is completely closed on the real-world dataset. We speculate that this is because the particle filter is better at handling discontinuous interactions: our simulation dataset consists of only a single pushing motion per trajectory, while our real-world dataset repeatedly makes and breaks contact.

2) Door Task: In the Mujoco door dataset, each filter is trained to estimate a 3-DoF state with different characteristics: the *x* and *y* coordinates of the location of the door hinge (stationary within a trajectory) and the hinge angle (changing based on robot's actions). We evaluate the filter performance on the door task with two metrics: the absolute error of the estimated hinge angle and the RMSE error of the hinge's location (we assume the hinge joint pointing in the *z* direction). We show our results in Fig. 3.

With the exception of `Unimodal Fusion EKF`, we see comparable results across all models for the door estimation task. For joint angle estimation, `Crossmodal Fusion` models had slightly better performance than `Feature Fusion` models. `LSTM` models remain fairly consistent with both joint angle and hinge location estimations, across all datasets.

C. Discussion

In the `Crossmodal Fusion` filters, we can inspect the outputs of the weighting models to understand how each modality is balanced. Fig. 5 depicts the predictions from each modality and the learned weights for the fusion of information by the `Crossmodal Fusion EKF` on the

real-world pushing dataset. We observe that the filter relies more on the information from proprioceptive and force-based modalities when the robot is in contact and pushing the object. When the robot stops pushing the object or loses contact, the filter learns to put more weight on information from the image-based EKF and less on the proprioception/haptics-based EKF.

When we use a `Crossmodal Fusion` filter, we can also examine outputs from each unimodal measurement model to help us understand what the filter is learning from each sensing modality. Fig. 4 depicts the hinge location likelihood from each unimodal measurement model in an end-to-end trained `Crossmodal Fusion PF`. From these visualizations, we observe that while the image input contains information to roughly infer both the *x* and *y* coordinates of the location of the hinge (placed at the origin in this case), the force/proprioception signal only reveals that the hinge location should be along the axis between the true hinge location and the end-effector location. Given the combined information of proprioception and force/torque in one timestep, the hinge location cannot be inferred unequivocally, only its distribution along the connecting line. By fusing likelihoods from these two unimodal measurement models, the learned filter is able to produce a better estimate of the position than it could if it only had access to one of them.

We expect the interpretability benefits of the proposed `Crossmodal Fusion` architectures to be especially advantageous for building estimators deployed in settings where safety and reliability are critical. In addition to using the belief uncertainty intrinsic to a Bayesian filter to inform decision-making, we can also use our understanding of how the estimator is interpreting each input modality to better diagnose why, how, and where in our estimation pipeline failures happen. By selectively freezing weights in parts of the estimator — for example, the image filter if we discover an issue with our force/proprioception filter — we can then patch these failures in an end-to-end manner while minimizing unintended impacts.

Similar to the findings of Kloss et al. [19], we find that the performance of each type of multimodal filter depends heavily on the task, with no unique best performing filter.
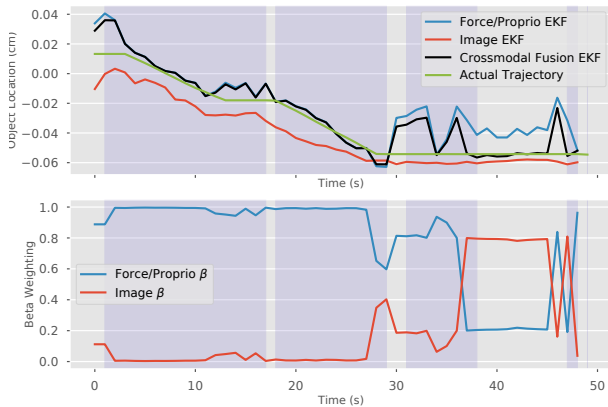
Fig. 5: For the real pushing task, we show the estimated states of the filters (top) and the weights for the unimodal filters (bottom) from the `Crossmodal Fusion EKF`. Contact is indicated by the purple background. The `Crossmodal Fusion EKF` learns to weight the force and proprioceptive unimodal EKF when the robot is touching the object, and switch to the image unimodal EKF when the robot breaks contact with the object.

VII. Conclusion

We presented a study of fusion architectures within the field of differentiable recursive filters for state estimation. For our experiments, we used the case studies of planar pushing and door opening to integrate visual, haptic, and proprioceptive data. We show that fusing these sensory inputs in differentiable filters that leverage crossmodal information can provide valuable opportunities for interpretability without sacrificing performance. This is particularly true for our proposed `Crossmodal Fusion` architectures, which allow for fine-grained analysis of how each modality contributes to our final state estimate.

In future work, we will investigate the proposed architectures for a broader set of manipulation tasks and with other state estimation techniques such as Unscented Kalman filters or factor graphs. Furthermore, we want to study how to combine multimodal state estimation with policy learning, which may allow us to select optimal actions for revealing task-relevant information.

References

[1] P. K. Allen, "Integrating vision and touch for object recognition tasks", *The International Journal of Robotics Research*, vol. 7, no. 6, pp. 15–33, 1988.

[2] T. Baltrušaitis, C. Ahuja, and L.-P. Morency, "Multimodal machine learning: A survey and taxonomy", *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 2, pp. 423–443, 2018.

[3] Y. Bar-Shalom and X.-R. Li, *Multitarget-multisensor tracking: principles and techniques*. YBs Storrs, CT, 1995, vol. 19.

[4] F. Caron, E. Duflos, D. Pomorski, and P. Vanheeghe, "Gps/imu data fusion using multisensor kalman filtering: Introduction of contextual aspects", *Information fusion*, vol. 7, no. 2, pp. 221–230, 2006.

[5] F. Caron, M. Davy, E. Duflos, and P. Vanheeghe, "Particle filtering for multisensor data fusion with switching observation models: Application to land vehicle positioning", *IEEE transactions on Signal Processing*, vol. 55, no. 6, pp. 2703–2719, 2007.

[6] C. G. Cifuentes, J. Issac, M. Wüthrich, S. Schaal, and J. Bohg, "Probabilistic articulated real-time tracking for robot manipulation", *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 577–584, 2016.

[7] H. F. Durrant-Whyte, "Sensor models and multisensor integration", *The International Journal of Robotics Research*, vol. 7, no. 6, pp. 97–113, 1988.

[8] T. Haarnoja, A. Ajay, S. Levine, and P. Abbeel, "Backprop kf: Learning discriminative deterministic state estimators", in *Advances in Neural Information Processing Systems*, 2016, pp. 4376–4384.

[9] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson, "Learning latent dynamics for planning from pixels", in *International Conference on Machine Learning*, 2019, pp. 2555–2565.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[11] P. Hebert, N. Hudson, J. Ma, T. Howard, T. Fuchs, M. Bajracharya, and J. Burdick, "Combined shape, appearance and silhouette for simultaneous manipulator and object tracking", in *2012 IEEE International Conference on Robotics and Automation*, IEEE, 2012, pp. 2405–2412.

[12] G. E. Hinton, "Products of experts", 1999.

[13] M. Igl, L. Zintgraf, T. A. Le, F. Wood, and S. Whiteson, "Deep variational reinforcement learning for pomdps", in *International Conference on Machine Learning*, 2018, pp. 2117–2126.

[14] J. Ilonen, J. Bohg, and V. Kyrki, "Three-dimensional object reconstruction of symmetric objects by fusing visual and tactile sensing", *The International Journal of Robotics Research*, vol. 33, no. 2, pp. 321–341, 2014.

[15] R. Jonschkowski, D. Rastogi, and O. Brock, "Differentiable Particle Filters: End-to-End Learning with Algorithmic Priors", in *Proceedings of Robotics: Science and Systems (RSS)*, 2018.

[16] M. Kaess, A. Ranganathan, and F. Dellaert, "Isam: Incremental smoothing and mapping", *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1365–1378, Dec. 2008.

[17] R. Kalman, "A new approach to linear filtering and prediction problems", *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[18] P. Karkus, D. Hsu, and W. S. Lee, "Particle filter networks with application to visual localization", in *Conference on Robot Learning*, 2018, pp. 169–178.

[19] A. Kloss and J. Bohg, "On learning heteroscedastic noise models within differentiable bayes filters", 2018.

[20] A. Kloss, S. Schaal, and J. Bohg, "Combining learned and analytical models for predicting action effects", *arXiv preprint arXiv:1710.04102*, 2017.

[21] D. Lahat, T. Adali, and C. Jutten, "Multimodal data fusion: An overview of methods, challenges, and prospects", *Proceedings of the IEEE*, vol. 103, no. 9, pp. 1449–1477, Sep. 2015.

[22] A. S. Lambert, M. Mukadam, B. Sundaralingam, N. Ratliff, B. Boots, and D. Fox, "Joint inference of kinematic and force trajectories with visuo-tactile sensing", in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 3165–3171.

[23] T. A. Le, M. Igl, T. Rainforth, T. Jin, and F. Wood, "Auto-encoding sequential monte carlo", in *International Conference on Learning Representations*, 2018.

[24] X. Ma, P. Karkus, D. Hsu, and W. S. Lee, "Particle filter recurrent neural networks", *arXiv preprint arXiv:1905.12885*, 2019.

[25] R. Martín-Martín and O. Brock, "Building kinematic and dynamic models of articulated objects with multi-modal interactive perception", in *2017 AAAI Spring Symposium Series*, 2017.

[26] ——, "Cross-modal interpretation of multi-modal sensor streams in interactive perception based on coupled recursion", in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2017, pp. 3289–3295.

[27] Z.-C. Marton, D. Pangercic, N. Blodow, and M. Beetz, "Combined 2d–3d categorization and classification for multimodal perception systems", *The International Journal of Robotics Research*, vol. 30, no. 11, pp. 1378–1402, 2011.

[28] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press Cambridge, 2000, vol. 1.

[29] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control", in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2012, pp. 5026–5033.

[30] K.-T. Yu, M. Bauza, N. Fazeli, and A. Rodriguez, "More than a million ways to be pushed. a high-fidelity experimental dataset of planar pushing", in *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, IEEE, 2016, pp. 30–37.

[31] K.-T. Yu and A. Rodriguez, "Realtime state estimation with tactile and visual sensing. application to planar manipulation", in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 7778–7785.