Estimating Perceptual Uncertainty to Predict Robust Motion Plans

Arjun Gupta Michelle Zhang Saurabh Gupta

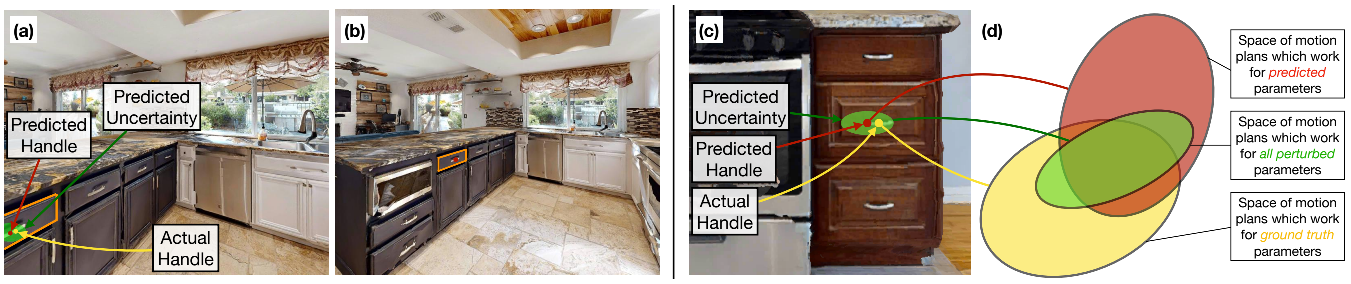
(a)
Predicted Uncertainty
Predicted Handle
Actual Handle
(b)
(c)
Predicted Uncertainty
Predicted Handle
Actual Handle
(d)
Space of motion plans which work for predicted parameters
Space of motion plans which work for all perturbed parameters
Space of motion plans which work for ground truth parameters

Fig. 1: State estimation from images has errors, predicted handle is off from actual handle in **(a)**. Objects in typical unoccluded views (as in **(b)**) have lower error than objects in atypical or occluded views (as in **(a)**). In this paper, we develop image-conditioned adaptive uncertainity predictors for such state estimators. In **(a)** and **(b)**, the maroon keypoint is the predicted handle location, and the green ellipsoid depicts one standard deviation in the handle keypoint uncertainty (very small in **(b)**). We utilize this uncertainty to output *robust motion plans* as shown in **(d)**. Rather than returning a motion plan that works just for the one inaccurate estimated state (shown by maroon set), we return a plan that works across multiple perturbations based on the predicted uncertainty (green set). A *robust* plan from the green set is more likely to succeed for the ground truth parameters, compared to a motion plan from the maroon set. We demonstrate that using our state uncertainty estimation leads to more robust downstream control.

***Abstract*— A typical sense-plan-act robotics pipeline is brittle due to the inherent inaccuracies in the output of the sensing module and the lack of awareness of the planning module to those inaccuracies. This paper develops a framework to predict uncertainty estimates for neural network-based vision models used for state estimation in robotics pipelines. Our uncertainty estimates are based directly on the image observation data and are explicitly trained to model the error distribution on a held-out calibration set. We also demonstrate how predicted uncertainties can be used to select robust control strategies. We conduct experiments on the mobile manipulation problem of articulating everyday objects (*e.g.* opening a cupboard) and demonstrate the quality of estimated uncertainty and its downstream impact on robustness of inferred control strategies.**

I. Introduction

A typical sense-plan-act robotics pipeline is brittle due to the inherent inaccuracies in the output of the sensing module and the lack of awareness of the planning module to those inaccuracies. Because of the lack of a way to faithfully characterize uncertainties in the output of the sensing model, past work either focuses on developing better sensing modules or makes planning overly conservative based on some constant hand-crafted notions of uncertainty. In this paper, we develop a) a framework to estimate uncertainty measures for state-of-the-art state estimators based on deep neural networks and b) demonstrate how downstream planning can make use of these uncertainties to output robust control strategies.

Our work is motivated by three key observations. First, we note that the observation itself may be suggestive of the level of uncertainty we could expect in the output of the vision module. Consider, for example, the different views of the same drawer in Figure 1 (a, b). Typical unoccluded object views as in Figure 1 (b) are more likely to have more accurate state estimation as compared to clipped views from odd angles as in Figure 1 (a). Thus, with the right modeling, it should be possible to extract uncertainty estimates. Our second insight is to cast uncertainty estimation directly as a learning problem itself. A popular line of past research estimates uncertainties by measuring variance in model ensembles [1]–[3] or data perturbations [4]. This however might provide an erroneously tight estimate when, for example, all models agree but are wrong. We instead directly learn an *observation-conditioned model* to predict the error distribution on a held-out calibration set. Our experiments suggest that this is a better uncertainty measure than one obtained from model ensembles. Our last insight is about use of uncertainties for making motion plans robust. Rather than predicting strategies that only work for the one predicted state estimate, a strategy that works for many perturbed samples (according to the uncertainty model) may be more likely to work for the actual instance (see Figure 1 (c, d)).

We use the problem of articulating everyday objects (opening doors and cupboards) from [5] as our test-bed. Tackling this problem requires predicting articulation parameters for these objects from RGB-D images. We adapt Mask RCNN [6] for this task by adding additional heads to predict these articulation parameters and the associated uncertainties (Figure 2). While the parameter prediction heads are trained on the training set, the uncertainty prediction heads are trained on predictions from the parameter heads on

Authors are with the University of Illinois, Urbana-Champaign. EMails: {arjung2, mz32, saurabhg}@illinois.edu. Project website with more details: https://arjung128.github.io/uncertainty_aware_planning.

a *separate* calibration set to maximize the log likelihood of the ground truth given the prediction. This is a more faithful estimate of the errors that the model is likely to make on novel data (predictions on the training set are typically overly accurate). This way of modeling uncertainties incorporates our first two insights: it is conditioned on the observation and it directly learns to model the actual error distribution.

We also show how the obtained uncertainty estimates can be used for predicting robust motion plans. Typical motion planners (*e.g.* RRTs [7] or PRMs [8]) solve each problem in isolation and there is no obvious notion of robustness to state estimation. We thus adopt the trajectory optimization view from Gupta *et al.* [5] and predict *strategies* (essentially, initialization of a trajectory optimizer) that work for the most perturbations sampled from the learned uncertainty model rather than just the predicted parameters.

Our experiments evaluate the effectiveness of our proposed framework at a) predicting good uncertainties (as measured using the negative log-likelihood on the test set), and b) their downstream utility at improving robustness of generated plans. We compare against ensembling and a number of design choices (types of inputs, dealing with outliers, and parameterization of the error model). We find our design choices to be effective and that modeling uncertainty adaptively (*i.e.* predicting uncertainty specifically for each instance) is better than using a non-adaptive uncertainty. Code will be made publicly available upon acceptance.

II. Related Work

Uncertainty Estimation in Machine Learned Models. Unlike probabilistic models like Gaussian Processes [9], it has been challenging to provide uncertainty estimates on the output of expressive deep neural network models. Bayesian neural networks [10], [11], while computationally expensive offer the most principled solution. Researchers have proposed use of model ensembling [1], [2] (or multiple passes through a network with Dropout layers [3]) as scalable alternatives. [4] use invariance to image transformations as a measure of the model's confidence. Explicit calibration using a held-out set has also been pursued [12], *e.g.* histogram binning [13], isotonic regression [14], Platt's scaling [15]. Conformal prediction techniques convert such calibrations into confidence intervals [16]. Our work builds uncertainty estimates for state estimates predicted from an object detector and we demonstrate their utility towards obtaining robust motion plans. This is similar in spirit to recent work from Liu *et al.* [17] that obtains multiple instance segments to minimize double pick errors for an apparel picking robot.

Motion Planning under Uncertainty. A large body of work studies motion planning under uncertainty [18]–[20]. A full literature survey is beyond the scope of the paper but we provide an overview. Motion planning under uncertainty is most typically thought of as a partially observed Markov decision process (POMDP) [21], which are computationally difficult to solve [22], [23]. Our work can be thought of as solving a single-step POMDP: we use the uncertainty-predicting vision models to construct the *belief* and return motion plans that are optimal under this belief. We do not develop a new POMDP solver. Instead, our focus is on building vision models that output uncertainties to build beliefs and demonstrating their utility for downstream planning.

Vision Models for State Estimation. Prior work has adapted object detectors pipelines to estimate state for articulated objects *e.g.* [24]–[27]. Our focus is orthogonal and our contributions are compatible with these recent advances.

III. Overview

Building robust controllers requires us to a) build uncertainty estimates for vision models, and b) design control strategies that can take these uncertainity estimates into account. We first present UfC (Uncertainties from Calibration), our approach for estimating uncertainties in Section IV and then outline UAP (Uncertainty Aware Planning), the procedure for using them to find robust controllers in Section V. We then discuss a concrete application of these ideas for the problem of predicting motion plans to articulate everyday objects (*e.g.* opening cupboards and drawers) in Section VI. This application involves building vision models to predict articulation parameters for drawers and cupboards (handle locations and surface orientation). We describe how we build vision models to predict these parameters along with associated uncertainties, and then show how we can use these estimates to predict robust motion plans.

IV. Estimating Prediction Uncertainties from Vision Models

Our methods, Uncertainties from Calibration (UfC), assumes access to a training dataset $\mathbf{D}$ and a calibration dataset $\mathbf{C}$. We train vision models on the training dataset $\mathbf{D}$ and train an uncertainty prediction model on the calibration dataset $\mathbf{C}$. Let's assume that training samples in datasets $\mathbf{D}$ and $\mathbf{C}$ take the form $\{(\mathbf{x}_i, y_i), \ldots\}$. We train state estimators on $\mathbf{D}$ via:

$$\arg\min_f \sum_{(\mathbf{x}_i, y_i)\in\mathbf{D}} (f(\mathbf{x}_i) - y_i)^2 \tag{1}$$

Given such a vision model f, we train a model $g(\mathbf{x})$ to represent $p_g(y_i|f(\mathbf{x}))$. Predictions of f on the training dataset $\mathbf{D}$ are overly accurate. Thus, it is unsuitable to train g on $\mathbf{D}$. This is where the held-out calibration set $\mathbf{C}$ comes in. We train g to model probability of the true state estimate y_i given the predicted state estimate $f(\mathbf{x})$ on samples from the calibration set $\mathbf{C}$. g is trained by minimizing the negative log likelihood (NLL):

$$\arg\min_g \sum_{(\mathbf{x}_i, y_i)\in\mathbf{C}} -\log p_g(y_i|f(\mathbf{x})) \tag{2}$$

In our work, we model p_g as a Gaussian distribution. Specifically, we only model the standard deviation σ of this Gaussian distribution and use $f(\mathbf{x})$ as the mean. Predicting a mean correction to $f(x)$ is unnecessary. If a meaningful mean could be learned, it would have already been learned inside f during training of the base model on $\mathbf{D}$.

Second, in practice we may not be given a separate calibration set, or training datasets may not be large enough to afford construction of a separate calibration set. We adopt a k-fold hold one out strategy to circumvent this issue. We split the data into k-folds, train f on $k-1$ folds and store its predictions on the k^{th} fold. We repeat this process k times over to obtain *held-out* predictions for training g.

Lastly, we found it useful to omit samples with large errors (*i.e.* a large $||y_i - f(\mathbf{x}_i)||_2$) while training $g(\mathbf{x})$.

V. Using Uncertainty Estimates for Predicting Robust Motion Plans

The vision model f along with the associated uncertainty model g allow us to a) assess the robustness of controllers and motion plans to errors in state estimation, and b) select controllers / motion plans that will be most robust to the expected amount of uncertainty. We broadly outline how we do this and then show a concrete examples for the task of opening doors and drawers in Section VI.

Output from f, g provide a probabilistic state estimate $\mathcal{N}(f(\mathbf{x}), g(\mathbf{x}))$. Let us assume we have different controllers or motion plans π_k that we are trying to chose from. These controllers can present some trade-offs: some controllers can be highly performant but only work in certain specific situations, while others could be less performant but could work in broader situations. Rather than selecting a controller that gives the best performance on the estimated state $f(\mathbf{x})$, we instead pick the one that is expected to work across most perturbations around the predicted state estimate $f(\mathbf{x})$ modulated by the predicted standard deviation $g(\mathbf{x})$. More formally, if $\rho(\pi_k, y)$ measures the fitness of a particular controller π_k, rather than selecting one via: $\pi_{\text{mode}} = \arg\max_k \rho(\pi_k, f(\mathbf{x}))$, we select one via:

$$\pi_{\text{robust}} = \arg\max_k \sum_{y \sim \mathcal{N}(f(\mathbf{x}), g(\mathbf{x}))} \rho(\pi_k, y) \quad (3)$$

As $\mathcal{N}(f(\mathbf{x}), g(\mathbf{x}))$ models the distribution of the actual state, π_{robust} is more likely to work than π_{mode}. In practice, we found rescaling the standard deviation of all state estimates by a single scalar factor of λ improved performance.

VI. Concrete Application: Predicting Motion Plans for Opening Drawers and Cupboards

A. Background

Articulating everyday objects (*e.g.* opening drawers and cupboards) requires precise motion of the end-effector under tight task constraints. This requires a) precise estimation of the articulation parameters for the object (handle location, radius of motion for cupboards / pull direction for drawers), b) mapping estimated articulation parameters into end-effector poses, and c) converting end-effector pose trajectories into robot joint trajectories that lead the end-effector to precisely track the desired trajectory. Step (b) of going from articulation parameters to end-effector trajectories is a deterministic function (let's denote it with β – refer to [5] for details).

We build upon past work from Gupta *et al.* [5] that assumes ground truth articulation parameters and tackles the problem of converting end-effector pose trajectories into robot joint angle trajectories. Specifically, rather than casting it as constrained motion planning problem, Gupta *et al.* view it as a trajectory optimization problem. They design SeqIK, a trajectory optimizer specifically suited to this task. SeqIK translates an initial robot configuration (base position and arm joint angles denoted by θ_0) into a *strategy* that can be decoded into a motion plan (desired joint angle trajectory) when provided with a desired end-effector pose trajectory $\mathbf{w}$, via SeqIK(θ_0)($\mathbf{w}$). They then train a neural network to predict a set of good initializations $\mathbf{\Theta}_0$ for SeqIK. Motion planning happens by *searching* through this small set of good initializations. This lets them produce accurate motion plans within only a few seconds. An advantage of this formulation is that a strategy SeqIK(θ_0) can be *adapted* to slightly different articulation parameters by just changing $\mathbf{w}$.

B. Problem Formulation

We extend their framework to work with predicted articulation parameters as opposed to ground truth articulation parameters. Given access to vision models that make (possibly inaccurate) predictions for articulation parameters, the goal is to output *strategies* that will be able to produce motion plans for the actual object.

C. Applying UfC and UAP

We build a vision model f to predict articulation parameters. Given an input RGB-D image of the object $\mathbf{x}$, f predicts the articulation parameters $f(\mathbf{x})$ which can be converted into end-effector trajectory via $\beta(f(\mathbf{x}))$. Predictions $f(\mathbf{x})$ may not be perfect and selecting a strategy seqIK(θ_0) purely based on how well it will work for $\beta(f(\mathbf{x}))$ may not work for the actual articulation parameters. This is where the uncertainty modeled by $g(\mathbf{x})$ comes in. It allows us to select the strategy SeqIK(θ_0) that works for a set of articulation parameters around $f(\mathbf{x})$ and thus has a higher chance of working for the actual object.

1) Building Vision Model (i.e. f): More specifically, we adopt Mask RCNN [6] to a) detect drawers and cupboards and b) predict articulation parameters for the detected instances. We do this by adding additional heads. For drawers, these additional heads predict a) the 2D coordinate of the handle, and b) the orientation of the drawer surface in the top-view. The 2D handle coordinate is lifted to 3D using the depth image. For cupboards, we similarly predict the 2D coordinate for the handle and surface orientation. We assume that the object extents can be reliably estimated from the extent of the 2D segment already predicted by Mask RCNN. We treat the 3 outputs as separate independent scalar predictions. Figure 2 shows the architectural modifications.

2) Learning Uncertainty Estimator (i.e. g): For the uncertainty estimator, we attach 2 additional heads to Mask RCNN: one each for handle and the surface normal uncertainty (see Figure 2). We model the distributions (x and y keypoint error, and surface normal error) as independent Gaussians, though other choices (joint modeling / other distributions) could also be made. Each of these heads consists

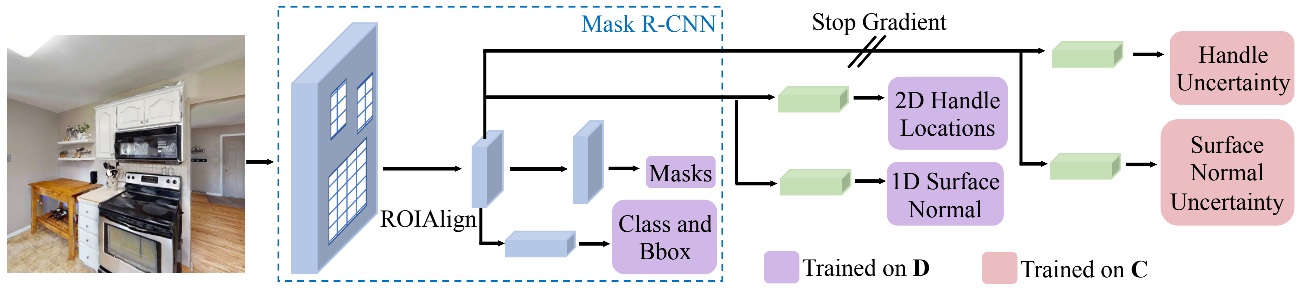

Fig. 2: Overview of state estimator f and the uncertainty predictors g. Both are realized by adding additional heads to Mask RCNN [6]. f is trained on the training set $\mathbf{D}$, while g is trained on the calibration set $\mathbf{C}$. Both f and g share the same backbone, but to control overfitting in g gradients don't flow back from g to the backbone.

of three convolutional layers, followed by a fully connected layer to predict category-specific standard deviation for the Gaussian distribution. To prevent overfitting, we keep the Mask RCNN backbone frozen (the stop gradient in Figure 2).

Rather than using a separate calibration set, we generate held-out predictions through 5-fold training and prediction: we split the training set into five disjoint subsets, train on four folds, generate predictions on the fifth folds, and repeat this process five times over. These leads to held-out predictions on the training set, which we use to train g. Lastly, since the error distribution is long-tailed, we remove 10% samples with the worst errors.

3) Using f and g for Robust Motion Plan Prediction: MPAO [5] returns an initialization θ_0 (from the set of initializations Θ_0 returned from a neural network) that leads to a *good plan*. Here, a good plan is one that a) tracks the desired end-effector trajectory and b) there isn't an anticipated collision with the articulated object, the robot itself, or the geometry depicted in the depth image. Let's denote the function that does these checks with IsPlanGood. MPAO uses ground truth articulation parameters to obtain the reference end-effector trajectory ($\mathbf{w}$) to test the goodness of the plan, *i.e.* it returns $\arg\max_{\theta_0 \in \Theta_0} \text{IsPlanGood}\left[\mathbf{w}, \text{SeqIK}(\theta_0)(\mathbf{w})\right]$.

Naively, one could just use articulation parameters predicted from our Mask RCNN model with this procedure to obtain motion plans, *i.e.*

$$\arg\max_{\theta_0 \in \Theta_0} \text{IsPlanGood}\left[\beta(f(\mathbf{x})), \text{SeqIK}(\theta_0)(\beta(f(\mathbf{x})))\right] \quad (4)$$

However, because predictions may be off from ground truth parameters, initializations that work for $f(\mathbf{x})$ may not actually work for $\mathbf{w}$. To combat this, we make use of our learned uncertainty estimator g to find initializations that are robust to state estimation errors via:

$$\arg\max_{\theta_0 \in \Theta_0} \sum_{y \sim \mathcal{N}(f(\mathbf{x}), g(\mathbf{x}))} \text{IsPlanGood}\left[\beta(y), \text{SeqIK}(\theta_0)(\beta(y))\right] \quad (5)$$

We approximate the sum using 20 samples. This yields a new *robust* ranking of the top-100 initializations, from which the best initialization is returned.

VII. Experiments

Our experiments evaluate: a) how does the quality of our uncertainty estimates compare to those obtained from ensembling (Section VII-B), b) what design choices lead to good models for predicting uncertainty (Section VII-C), and c) whether the use of uncertainty estimators leads to more robust downstream control (Section VII-D). We start by noting the performance of Mask RCNN for the state estimation task at hand (Section VII-A) and then proceed to present the main results.

TABLE I: Median Negative Log Likelihood (NLL) over the test set (lower is better). Our method outperforms uncertainty estimates from ensembling.

	Prismatic (Handle)	Hinge (Handle)	Prismatic (Surf. Norm.)	Hinge (Surf. Norm.)
Ensembling	3.87	4.00	-1.80	-1.87
Ours	**-2.20**	**-2.58**	**-1.92**	**-1.93**

A. State Estimation Performance

We focus our study to the two categories with the most data in the dataset from [5]: drawers and vertical hinges (horizontal hinges (the remaining 2 categories) had an order of magnitude less data and state estimators had very poor performance). On the test set, our model achieves a 8.75 pixel mean and 2.15 pixel median L2 keypoint error for the handle location, and a 0.06 radian mean and 0.01 radian median absolute error for the surface normal prediction across prismatic and vertical hinge objects.

B. Uncertainty Evaluation

Our primary baseline is an ensembling-based approach that uses the standard deviation across five predictions from the 5 bootstrapped Mask RCNN models as the uncertainty estimate. We use the median negative log likelihood to measure the effectiveness of the uncertainty modeling (mean NLL was dominated by outliers). Table I shows the results. We observe that ensembling obtains worse NLL than our method across all 4 cases: 2 articulation types $\times$ 2 articulation parameters.

C. Uncertainty Evaluation (Ablations)

We perform a number of ablations to study which design choices lead to the best uncertainty modeling.

1) Model Input: First, we investigate what inputs to g are the best for modeling uncertainty. We experiment with 3 variants. *a) No input*, this amounts to simply fitting a Gaussian to the errors. *b) 2D object location in the image*, represented

TABLE II: Median NLL over the test set. Conditioning on image features leads to best uncertainty modeling.

	Prismatic (Handle)	Hinge (Handle)	Prismatic (Surf. Norm.)	Hinge (Surf. Norm.)
No Input	-2.02	-1.63	-1.46	-1.46
2D Object Location	**-2.22**	-2.39	-1.60	-1.68
Object Appearance	-2.20	**-2.58**	**-1.92**	**-1.93**

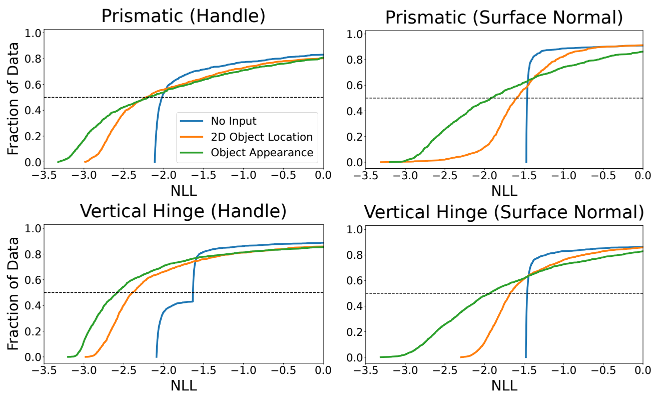

Fig. 3: Cumulative plot of negative log likelihoods (lower better) on the test set for different attributes of different categories. Even though we only report the medians in the other tables, the entire NLL distribution improves.

as the top-left and bottom-right corners ($[x, y, x+w, y+h]$) of the detection box. This can capture trends such as objects in the center of the image have lower error than objects at the corners. *c) Object Appearance*, represented as ROI-Align features of the detection box.

Table II reports the median negative log likelihood over the test set for different input choices. 2D object location in the image is predictive of the uncertainty but appearance features leads to the best uncertainty estimates overall. The gap between conditioning on image features and the objcet location is particularly large for the surface normal error estimation. Figure 3 shows cumulative plots of the fraction of data as a function of the negative log likelihood on the test set. Not only is the median better when the uncertainty estimators use image features, but a larger fraction of the samples attain a low loss relative to the baselines.

2) Uncertainty Parameterization: p_g is modelled as a Gaussian distribution, We investigate how to best model this Gaussian. Specifically, for the most performant models (ones that use object appearance), we experiment with variants where g predicts a) only the mean, b) only the standard deviation, or c) both the mean and standard deviation of the Gaussian distribution. When the model only predicts a mean, we use the standard deviations from the *No Input* model above to compute the negative log likelihood. Mean is set to 0 for the model that only predicts a standard deviation.

Table III shows the results. Predicting only the mean does poorly. This is not surprising as, if it was possible to predict the mean of the error to any extent, then the original Mask RCNN models would have done so to reduce their prediction errors. Predicting both the mean and standard deviation is more expressive, and does better. While this method could also in principle memorize errors on the training set and predict low standard deviations, in practice, limited model

TABLE III: Median NLL over the test set. Predicting standard deviation only leads to best uncertainty modeling.

	Prismatic (Handle)	Hinge (Handle)	Prismatic (Surf. Norm.)	Hinge (Surf. Norm.)
Mean Only	0.59	0.60	0.62	0.62
Mean and Std	-2.07	-2.41	**-1.95**	**-2.02**
Std Only	**-2.20**	**-2.58**	-1.92	-1.93

TABLE IV: Median NLL over the test set. Removing outliers from the training set helps uncertainty modeling significantly.

	Prismatic (Handle)	Hinge (Handle)	Prismatic (Surf. Norm.)	Hinge (Surf. Norm.)
w/ Outliers	-2.05	-1.94	**-1.94**	**-1.97**
w/o Outliers	**-2.20**	**-2.58**	-1.92	-1.93

capacity and regularization makes this difficult to do. Finally, we observe that predicting only the standard deviation of the distribution leads to the best performance. Intuitively, this suggests that while such an uncertainty estimation model cannot predict the exact error, it can learn to model the distribution of the error.

3) Removing Outliers: As noted above, there are a non-trivial number of large-error outliers in the dataset. We did not train on instances with a handle error larger than the 90th-percentile error value. We check if this was necessary.

Table IV shows our results. We observe that the median loss is much lower for the model trained *without* outliers for handles. We do not see a gap for surface normal, as the outliers for surface normals were not removed from the training set. Note that the test set is identical between the two methods, *i.e.* it contains outliers. This result suggests that the uncertainty estimator caters to the outliers when trained with outliers: if the model predicts a small standard deviation, which would be optimal for the vast majority of the dataset, the loss it would incur due to the extreme outliers would be enormous, discouraging it from doing so.

4) Visualizations: Figure 4 (top) visualizes uncertainty predictions for various views of the same object. We observe that more typical views of the object and when the object is close to the center of the image lead to low uncertainty estimates, whereas oblique views of the object as well as clipping seem to increase the uncertainty of the model.

Figure 4 (bottom) visualizes the predicted uncertainty using our image-based uncertainty estimation models across the dataset. We observe that the predicted handle uncertainty tends to be aligned with the handles, *i.e.* vertically-oriented handles tend to have a larger vertical uncertainty, and vice versa. Since the handles were labelled by human annotators to select one keypoint, this reveals an interesting dataset bias where different human annotators may have selected different points along the handle as labels. This suggests that the image-based models are able to extract signal for modeling uncertainty from visual cues, unlike methods which are not conditioned on the image.

D. Downstream Application

We use the setup described in Section VI-A and Section VI-C.3 to investigate whether our perceptual uncertainty

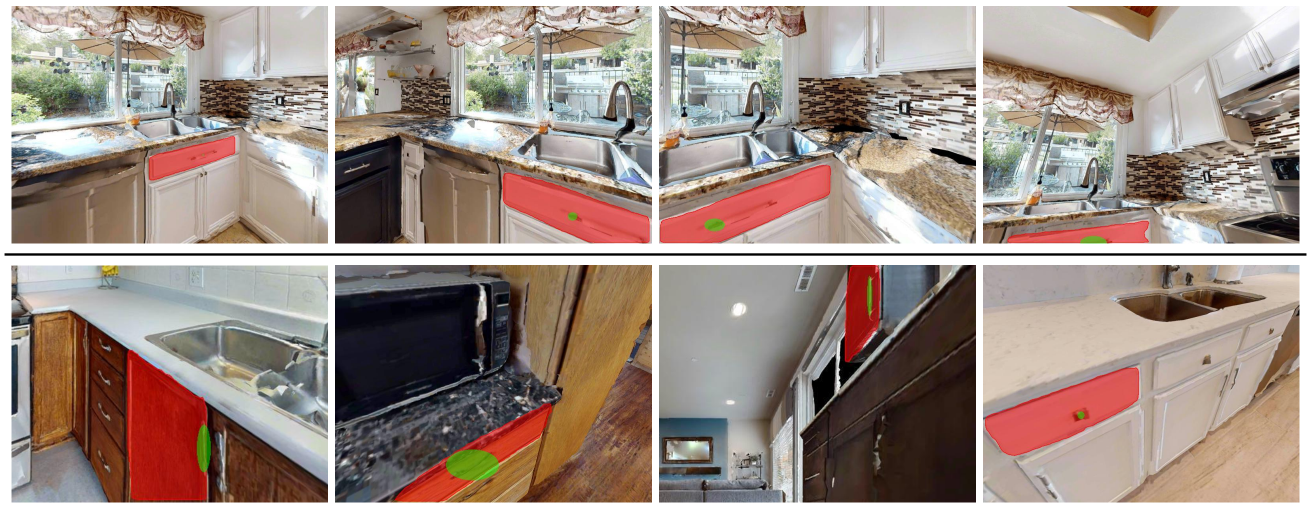
Fig. 4: Top: Uncertainty in keypoint estimation (green ellipsoid depicts 1 standard deviation), for the same object, varies as a function of viewpoint. Poorer views (atypical or occluded or truncated) have higher uncertainty. **Bottom:** Uncertainty in 2D handle location as captured by our model. Our image-based uncertainty models correctly capture the variation in the handle location in the dataset (can be marked anywhere along the handle length).

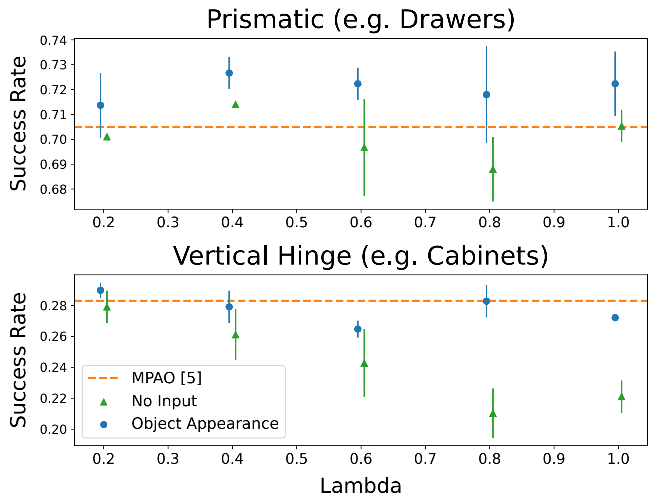

Fig. 5: Success rate as a function of λ for downstream task on val set.

models can lead to improved downstream robustness. We specifically compare against a) using no uncertainity, b) using a non-adaptive category-level uncertainty (*i.e.* the one coming from no-input UfC), and c) using adaptive uncertainties output by our full UfC model.

Evaluation Protocol. Given a ranking of initializations, we decode each initialization to a motion plan for the Mask RCNN predicted articulation parameters, and perform an internal check: $\text{IsPlanGood}\left[\beta(f(\mathbf{x})), \text{SeqIK}(\theta_0)(\beta(f(\mathbf{x})))\right]$ Once an initialization passes the internal check, it is returned. This initialization is used to find a motion plan for ground truth articulation parameters. A success is defined as $\text{IsPlanGood}\left[\mathbf{w}, \text{SeqIK}(\theta_0)(\mathbf{w})\right]$, with the only difference being that collision checking is done with the full environment scan as opposed to the partial view from the depth image.

λ **Parameter.** While UfC predicts uncertainties exhibited by the vision system, it is possible that the uncertainties that are useful to extract robust motion plans are correlated with perceptual uncertainties but not identical. To account for this possible mismatch, we use $\lambda g(\mathbf{x})$ as the standard deviation for sampling perturbations for UAP, where λ is a category specific scalar hyper-parameter.

TABLE V: Downstream results on the test set.

	Prismatic	Hinge
MPAO [5]	**0.804**	0.349
No Input	0.791	0.349
Object Appearance (ours)	**0.804**	**0.358**

Figure 5 shows the effect of varying λ for the different methods on the validation set. We observe that at the optimal λ value, uncertainties from the image-based UfC model outperform both the no error modeling (MPAO [5]) and non-adaptive uncertainity baseline. We select the best λ for each category for each method: $\lambda = 0.4$ for prismatic objects and $\lambda = 0.2$ for hinge-joints.

Results. Table V shows results on the test set. We observe that the trends from the validation set translate to the test set, and image-based UfC is better than or on-par with no explicit uncertainty modeling as well as the non-adaptive no-image uncertainty model. Furthermore, comparison to the non-adaptive model demonstrates that poor error modeling can actually *hurt* downstream performance, underscoring the necessity of good uncertainty modeling.

VIII. Conclusion

Motivated by the need to make planning aware of the inevitable errors in the output of state estimators, we developed an explicit calibration approach to estimate uncertainty for deep learning based state estimators typically used in robotics. We demonstrated that this leads to better uncertainty estimates than those obtained from ensembling approaches. We then developed a technique that uses the predicted uncertainties to generate control strategies that are robust to errors in state estimation and experimentally demonstrated the utility of our uncertainties over simpler alternatives.

References

[1] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," *Advances in neural information processing systems*, vol. 30, 2017.

[2] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy, "Deep exploration via bootstrapped dqn," *Advances in neural information processing systems*, vol. 29, 2016.

[3] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*. PMLR, 2016, pp. 1050–1059.

[4] Y. Bahat and G. Shakhnarovich, "Confidence from invariance to image transformations," *arXiv preprint arXiv:1804.00657*, 2018.

[5] A. Gupta, M. Shepherd, and S. Gupta, "Predicting motion plans for articulating everyday objects," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2023.

[6] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *ICCV*, 2017.

[7] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *ICRA*, 2000.

[8] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[9] C. E. Rasmussen, C. K. Williams *et al.*, *Gaussian processes for machine learning*. Springer, 2006, vol. 1.

[10] R. M. Neal, *Bayesian learning for neural networks*. Springer Science & Business Media, 2012, vol. 118.

[11] D. J. MacKay, "A practical bayesian framework for backpropagation networks," *Neural computation*, vol. 4, no. 3, pp. 448–472, 1992.

[12] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *International conference on machine learning*. PMLR, 2017, pp. 1321–1330.

[13] B. Zadrozny and C. Elkan, "Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers," in *Icml*, vol. 1, 2001, pp. 609–616.

[14] ——, "Transforming classifier scores into accurate multiclass probability estimates," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002, pp. 694–699.

[15] J. Platt *et al.*, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," *Advances in large margin classifiers*, vol. 10, no. 3, pp. 61–74, 1999.

[16] A. N. Angelopoulos and S. Bates, "A gentle introduction to conformal prediction and distribution-free uncertainty quantification," *arXiv preprint arXiv:2107.07511*, 2021.

[17] Y. Liu, N. Mishra, P. Abbeel, and X. Chen, "Distributional instance segmentation: Modeling uncertainty and high confidence predictions with latent-maskrcnn," *arXiv preprint arXiv:2305.01910*, 2023.

[18] M. Lauri, D. Hsu, and J. Pajarinen, "Partially observable markov decision processes in robotics: A survey," *IEEE Transactions on Robotics*, vol. 39, no. 1, pp. 21–40, 2022.

[19] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial intelligence*, vol. 101, no. 1-2, pp. 99–134, 1998.

[20] A. Somani, N. Ye, D. Hsu, and W. S. Lee, "Despot: Online pomdp planning with regularization," *Advances in neural information processing systems*, vol. 26, 2013.

[21] R. D. Smallwood and E. J. Sondik, "The optimal control of partially observable markov processes over a finite horizon," *Operations research*, vol. 21, no. 5, pp. 1071–1088, 1973.

[22] C. H. Papadimitriou and J. N. Tsitsiklis, "The complexity of markov decision processes," *Mathematics of operations research*, vol. 12, no. 3, pp. 441–450, 1987.

[23] C. Lusena, J. Goldsmith, and M. Mundhenk, "Nonapproximability results for partially observable markov decision processes," *Journal of artificial intelligence research*, vol. 14, pp. 83–103, 2001.

[24] H. Jiang, Y. Mao, M. Savva, and A. X. Chang, "Opd: Single-view 3d openable part detection," in *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIX*. Springer, 2022, pp. 410–426.

[25] X. Sun, H. Jiang, M. Savva, and A. X. Chang, "Opdmulti: Openable part detection for multiple objects," 2023.

[26] J. Liu, A. Mahdavi-Amiri, and M. Savva, "PARIS: Part-level reconstruction and motion analysis for articulated objects," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2023.

[27] X. Li, H. Wang, L. Yi, L. Guibas, A. L. Abbott, and S. Song, "Category-level articulated object pose estimation," *arXiv preprint arXiv:1912.11913*, 2019.