Deep Forward and Inverse Perceptual Models for Tracking and Prediction

Alexander Lambert, Amirreza Shaban, Zhen Liu and Byron Boots
Institute for Robotics & Intelligent Machines
Georgia Institute of Technology, Atlanta, GA, USA
{alambert6, amirreza, liuzhen1994}@gatech.edu, bboots@cc.gatech.edu

***Abstract*—We present a non-parametric perceptual model for generating video frames with deep networks, and provide a framework for its use in tracking and prediction tasks on a real robotic system. This is shown to greatly outperform standard de-convolutional methods for image generation, producing clear photo-realistic images. For tracking, we incorporate the sensor model into an Extended Kalman Filter and estimate robot trajectories. As a comparison, we introduce a secondary framework consisting of a discriminative, inverse model for state estimation, and compare this approach to that using a generative model.**

I. Introduction & Related Work

Several fundamental problems in robotics, such as state estimation, prediction, and motion planning rely on accurate models that can map state to measurements (forward models) or measurements to state (inverse models). Classic examples include the measurement models for global positioning systems, inertial measurement units, or beam sensors that are frequently used in simultaneous localization and mapping [16], or the forward and inverse kinematic models that map joint configurations to workspace and *vice-versa*. Some of these models can be very difficult to derive analytically, and, in these cases, roboticists have often resorted to machine learning to infer accurate models directly from data. For example, complex nonlinear forward kinematics have been modeled with techniques as diverse as Bayesian networks [3] and Bezier Splines [17], and many researchers have tackled the problem of learning inverse kinematics with nonparametric methods like locally weighted projection regression (LWPR) [18, 6], mixtures of experts [2], and Gaussian Process Regression [13]. While these techniques are able to learn accurate models, they rely heavily on prior knowledge about the kinematic relationship between the robot state space and work space.

Despite the important role that forward and inverse models have played in robotics, there has been little progress in defining these models for very high-dimensional sensor data like images and video. This has been disappointing: cameras are a cheap, reliable source of information about the robot and its environment, but the precise relationship between a robot pose or configuration, the environment, and the generated image is extremely complex. A possible solution to this problem is to *learn* a forward model that directly maps the robot pose or configuration to high-dimensional perceptual space or an inverse model that maps new images to the robot pose or configuration. Given these learned models, one could directly and accurately perform a range of important tasks including state estimation, prediction, and motion planning.

In this paper, we will explore the idea of directly learning forward and inverse perceptual models that relate high-dimensional images to low dimensional robot state. In particular, we use deep neural networks to learn both forward and inverse perceptual models for a camera pointed at the manipulation space of a Barret WAM arm. While recent work on convolutional neural networks (CNNs) provides a fairly straightforward framework for learning inverse models that can map images to robot configurations, learning accurate generative (forward) models remains a challenge.

Generative neural networks have recently shown much promise in addressing the problem of mapping low-dimensional encodings to a high-dimensional pixel-space [4, 12, 8, 14]. The generative capacity of these approaches is heavily dependent on learning a strictly parametric model to map input vectors to images. Using deconvolutional networks for learning controllable, kinematic transformations of objects has previously been demonstrated, as in [5, 15]. However, these models have difficulty reproducing clear images with matching textures, and have mainly been investigated on affine transformations of simulated objects.

Learning to predict frames has also been conducted on two-dimensional robot manipulation tasks. In [7], the authors propose using an LSTM-based network to predict next-frame images, given the current frame and state-action pair. In order to model pixel transformations, the authors make use of composited convolutions with either unconstrained or affine kernels. The generated image frames produce some semblance of linear motion in the scene, however do not manage to replicate multi-degree-of-freedom dynamics. Given that forward prediction is conducted by recursive input of predicted frames, the error is compounded and the quality of predictions quickly degrades over future timesteps.

Instead of generating images directly after applying transformations in a low-dimensional encoding, another approach is to learn a transformation in the high-dimensional space of the output. One can then re-use pixel information from observed images to reconstruct new views from the same scene, as proposed in [19]. Here, the authors learn a model to generate a flow-field transformation from an input image-pose pair derived from synthetic data. This is then subsequently applied to a reference frame, effectively rotating the original

image to a previously unseen viewpoint. Using confidence masks to combine multiple flow-fields generated from different reference frames is also proposed. However, these frames are selected randomly from the training data.

In the current work, we propose deep forward and inverse perceptual models for a camera pointed at the manipulation space of a Barrett WAM arm. The forward model maps a 4-dimensional arm configuration to a $(256 \times 256 \times 3)$-dimensional RGB image and the inverse model maps $(256 \times 256 \times 3)$-dimensional images to 4-dimensional configurations. A major contribution of this work is a new forward model that extends the image-warping model in [19] with a *non-parametric* reference-frame selection component. We show that this model can generate sharp *near photo-realistic* images from a never-before-seen 4-dimensional arm configurations and greatly outperforms state of the art deconvolutional networks [5, 15]. We also show how to design a convolutional inverse model, and use the two models in tandem for tracking the configuration of the robot from an unknown initial configuration and prediction to future never-before-seen images on the real robot.

II. Sensor Model Design

Tracking and prediction are important tasks in robotics. Consider the problem of tracking state and predicting future images, illustrated in Fig.1. Given a history of RGB-image observations $O_t = (o_i)_{i=0}^{t}$ where $o \in \mathbb{R}^m$, we wish to track the state of the system $x \in \mathbb{R}^n$ for the corresponding sequence $X_t = (x_i)_{i=0}^{t}$. Additionally, given a designated goal image o_T, we wish to define a target end state x_T. Both these state-estimation objectives can be accomplished with an inverse sensor model, a parametrized function $g : \mathbb{R}^m \mapsto \mathbb{R}^n$. Having mapped the planning problem in joint-space, we can perform online motion-planning to acquire a sequence of expected states $\overline{X}_t = (\overline{x}_i)_{i=t+1}^{T}$. Classically, model-based motion-planning would rely on cost functions computed in simulation. However, we can translate a particular motion plan into observation space using a forward sensor model, allowing future observations to be predicted given a sequence of commands and trajectory. This could also permit the use of a cost function to be defined in observation space, which is subsequently leveraged by the motion planner to ensure trajectories are compatible with the real-world scene.

A. Forward Sensor Model

We use a generative observation model g to perform a mapping from joint-space values x_i to pixel-space values o_i for a trajectory sequence of future states $\overline{X}_t$. This is performed using image-warping layers as proposed in [19]. Given an input x_i and a reference state-image pair $(x_r; o_r)$, we learn to predict the pixel flow field $\overrightarrow{h}(x_i; x_r)$ from reference image o_r to the image o_i. The final prediction is made by warping the reference image o_r with the predicted flow field $\overrightarrow{h}$:

$$g(x_i) = \text{warp}(o_r; \overrightarrow{h}): \quad (1)$$

As long as there is a high correlation between visual appearance of the reference image o_r and the output image o_i, warping the reference image results in higher-fidelity predictions when compared to models which map input directly to RGB values [15].

We propose a variation of this idea that is extremely effective in practice. Our network is comprised of two parts: 1) A *non-parametric* component, where given a state value x_i, a reference pair $(x_r; o_r)$ is found such that o_r has a similar appearance as the output image o_i; and 2) A parametric component, where given the same state value x_i and the reference pair $(x_r; o_r)$, the inverse flow field $\overrightarrow{h}$ is computed and used to warp the reference image to produce the output image o_i. The overall architecture is shown in Fig.2.

For the non-parametric component, we take a subset of the training set $\mathscr{F}_p \subseteq \mathscr{F}$, consisting of configuration-image pairs and store the data in a KD-tree. We can quickly find a reference pair by searching for the training data pair closest to the current joint configuration x_i in configuration space. Since the state-space is low-dimensional we can find this nearest neighbor very efficiently. To prevent over-fitting, we randomly sample one of the k-nearest neighbors in the training phase.

For the parametric component, we use the deep neural network architecture depicted in Figure 2. Given a current and reference state vector as input, the network is trained to produce a flow tensor to warp the reference image (similarly to [19]). The input is first mapped to a high dimensional space using six fully-connected layers. The resulting vector is reshaped into a tensor with size $256 \times 8 \times 8$. The spatial resolution is increased from 8×8 to $2^8 \times 2^8$ by using 5 deconvolution layers. Following the deconvolutions with convolutional layers was found to qualitatively improve the output images. Finally, we use the predicted flow field of size $2 \times 256 \times 256$ to warp the reference image. To allow end-to-end training, we used a differentiable image sampling method with bi-linear interpolation kernel in the warping layer [9]. Using L_2 prediction loss, the final optimization function is defined as follows:

$$\min_W L(W) = \sum_i ||o_i - \text{warp}(o_r; \overrightarrow{h}_W(x_i; x_r))||^2 + l\,||W||^2 \quad (2)$$

in which W contains the network parameters and l is the regularization coefficient. Training was conducted using the Caffe [10] library, which had been extended to support the warping layer. The ADAM optimization algorithm [11] was used for the solver method.

The idea of warping the nearest neighbor can be extended to warping an ensemble of k-nearest neighbor reference images, with each neighbor contributing separately to the prediction. In addition to the flow field, each network in the ensemble also predicts a 256×256 confidence map (as shown in Fig.2). We use these confidence maps to compute the weighted sum of different predictions in the ensemble and compute the final prediction.

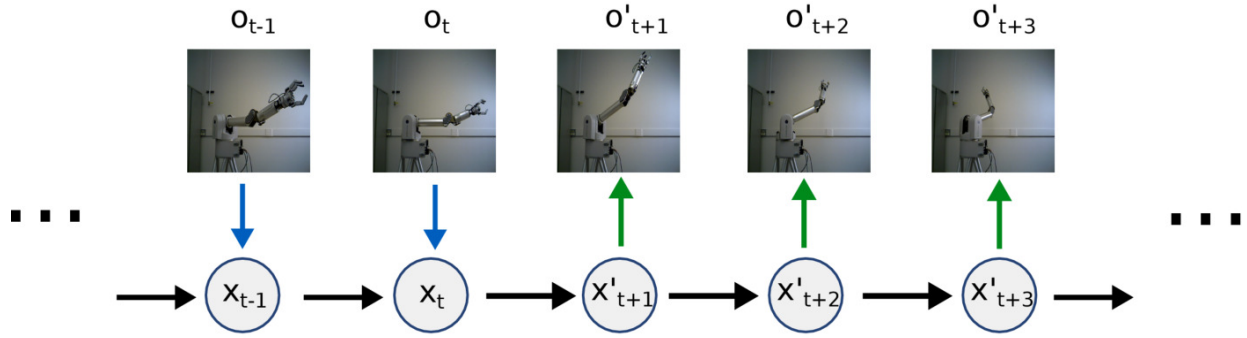

Fig. 1. The proposed framework makes use of an inverse sensor model (blue) for inference of a state x_i from observed frame o_i. A motion plan can be generated given a simulated model of the system. The resulting trajectory can be mapped into image space using a forward sensor model (green).

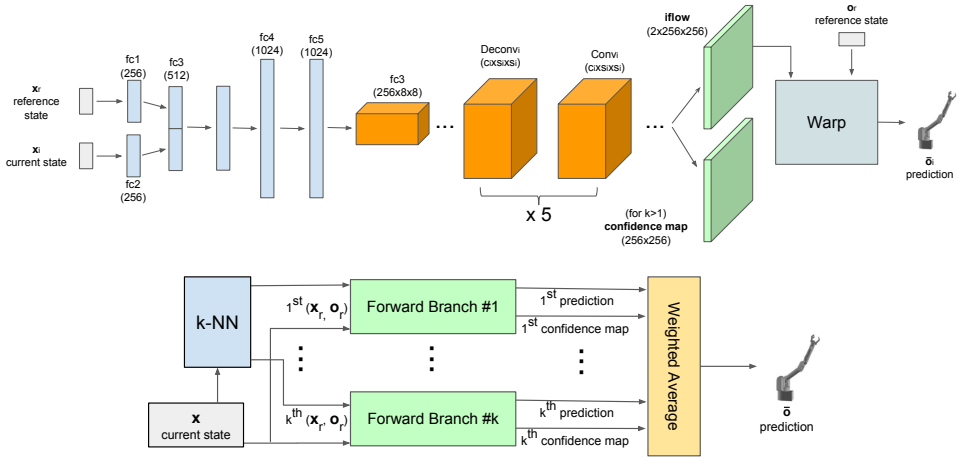

Fig. 2. Forward sensor model architecture. **Top**: Forward parametric branch example. **Bottom**: Complete architecture, containing a Nearest Neighbor (NN) module producing (image, state)-pair outputs. These are used by individual Forward branches to produce a warped reference image.

B. Tracking using the Forward Model

A common approach to state estimation is to use a Bayesian filter with a generative sensor model to provide a correction conditioned on measurement uncertainty. For instance, the forward model described in Section II-A could be used to provide such a state update, and potentially allow for a single model to be used for both tracking and prediction. We therefore consider this approach as a suitable comparison to the inverse-forward framework being proposed.

In order to track a belief-state distribution, we can derive an Extended Kalman Filter from the forward sensor model. This is a straightforward process, as network models are inherently amenable to linearization. The correction step is performed by computing the Jacobian J as the product of the layer-wise Jacobians:

$$J = J^{(L)} \times J^{(L-1)} \times \cdots \times J^{(0)}; \tag{3}$$

where L is the total number of layers in the network. The dimensionality of the observations makes it impractical to compute the Kalman Gain $\left(K = \mathsf{S}J^T(J\mathsf{S}J^T + R)^{-1}\right)$ directly. Instead, we use a low-rank approximation of the sensor-noise covariance matrix R, and perform the inversion in projected space:

$$(J^T\mathsf{S}J + R)^{-1} \approx U(U^T J^T \mathsf{S}JU + S_R)^{-1}U^T \tag{4}$$

where S_R contains the top-most singular values of R. For simplicity, the state prediction covariance Q was approximated as $Q = gI_{n\times n}$, where $g = 10^{-6}$. A first-order transition model is used, where next-state joint-value estimates are defined as $x'_{t+1} = x_t + (x_t - x_{t-1})\mathsf{D}t$ for a fixed time-step $\mathsf{D}t$. We provide an initial prior over the state values with identical covariance, and an arbitrary mean-offset from ground-truth.

C. Tracking using an Inverse Sensor Model

In order to infer the latent joint states x_i from observed images o_i, we can also define a discriminative model g_{fwd}. Given the capacity of convolutional neural network models to perform regression on high-dimensional input data, we used a modified version of the VGG CNN-S network [1] containing

5 convolutional layers and 4 fully-connected layers (shown in Fig.3). The model was trained on $256 \times 256 \times 3$ input images and corresponding joint labels, optimizing an L_2 loss on joint values and regularized with dropout layers.

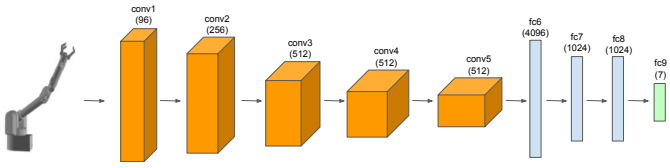

Fig. 3. Inverse sensor model architecture.

III. Experimental Results

A. Datasets

The experiments were conducted using a Barrett WAM manipulator, a cable-actuated robotic arm. Data was captured from raw joint-encoder traces and a statically-positioned camera, collecting 640x480 RGB frames at a rate of 30 fps. Due to non-linear effects arising from joint flexibility and cable stretch, large joint velocities and accelerations induce discrepancies between recorded joint values and actual positions seen in the images. In order to mitigate aliasing, the joint velocities were kept under 10°=s. This and other practical constraints imposed limitations on obtaining an adequate sampling density of the four-dimensional joint space. As such, the training data was collected while executing randomly generated linear trajectories using only the first four joints (trajectories were made linear for simplicity). A total of 225,000 camera frames and corresponding joint values were captured, with 50,000 reserved for a nearest-neighbor data-set, and the remaining for training data. Test data was collected for arbitrary joint trajectories with varying velocities and accelerations (without concern for joint flexibility and stretch).

B. Forward Sensor Model Evaluation

We first examine the predicted observations generated by the proposed forward sensor model (kNN-FLOW). A deconvolutional network (DECONV) similar to that used in [5] is chosen as a baseline, with the absence of a branch for predicting segmentation masks (as these are not readily available from RGB data). Qualitative comparisons between the ground-truth, forward-sensor predictions, and DECONV outputs are depicted in Fig.6 for a sequence of state-input values at various times on a pair of test trajectories. Detailed texture and features have been preserved in the kNN-FLOW predictions, and the generated robot poses closely match the ground truth images. The DECONV outputs suffer from blurred reconstructions, as expected, and do not manage to render certain components of the robot arm (such as the end-effector). Quantitative results are shown in Table I, where it is apparent that the model outperforms the DECONV baseline in both mean L_1 and RMS pixel error.

RGB Error	Mean L_1	RMS
kNN-FLOW (ours)	**0.01109**	**0.023097**
DECONV (baseline)	0.03066	0.061374

TABLE I
Forward model baseline comparison.

C. Tracking Task Evaluation

Given a sequence of observations $\{o_0; o_1; \ldots; o_{t-1}; o_t\}$, we wish to infer the corresponding sequences of state values $\{x_0; x_1; \ldots; x_{t-1}; x_t\}$ which includes the current state. This tracking task can be accomplished with a discriminative (inverse) sensor model, which provides deterministic subspace values independently of previously observed frames. An example of tracking accuracy, using the model described in Section II-C, is shown in Fig.4, which demonstrates frame-by-frame tracking capability within a margin of 3-degrees from ground-truth data.

We evaluate the accuracy of the EKF tracker on the same test trajectories as the inverse model. A single-trajectory example is shown in Figure 4 for 5 and 10-degrees mean-offsets at t = 0, where the ground-truth joint values are identical to those shown for the inverse model. The results demonstrate the ability of the tracker to converge the state estimate to the true trajectory over time, given an initial prior.

The stability of the tracking is highly dependent on the quality of the generated images. Discontinuities arise from the non-parametric component of the observation model, as selected nearest-neighbors may abruptly change during a tracked trajectory. Although this effect is mitigated by using a softmax mask for weighting of nearest-neighbor-flow outputs, these sudden jumps in the value of the Jacobian and residual terms can lead to aberrations in the tracking performance.

The comparatively high robustness of the inverse model in tracking is further demonstrated by estimating the state of an arbitrary nonlinear test trajectory shown in Figure 5. No latent dynamics model is assumed here, and state estimates are produced independently given a currently observed frame.

IV. Conclusions

A framework for tracking and prediction has been proposed, which consists of separate models for purposes of state estimation and generation of future observations. Beginning with the task of frame prediction, a non-parametric approach is taken to warping reference frames to generate novel instances of the scene. In both a quantitative and qualitative sense, this generative network produces improved results over the deconv-net baseline.

For state-estimation and tracking, it is shown that the generative observation model can be used in an EKF-based framework to perform probabilistic inference on the underlying latent state, and track the manipulator state over a simple trajectory.

For this given dataset, the object is essentially fully-observed, with limited instances of self-occlusion. As such, it is shown that using a straight-forward discriminative model

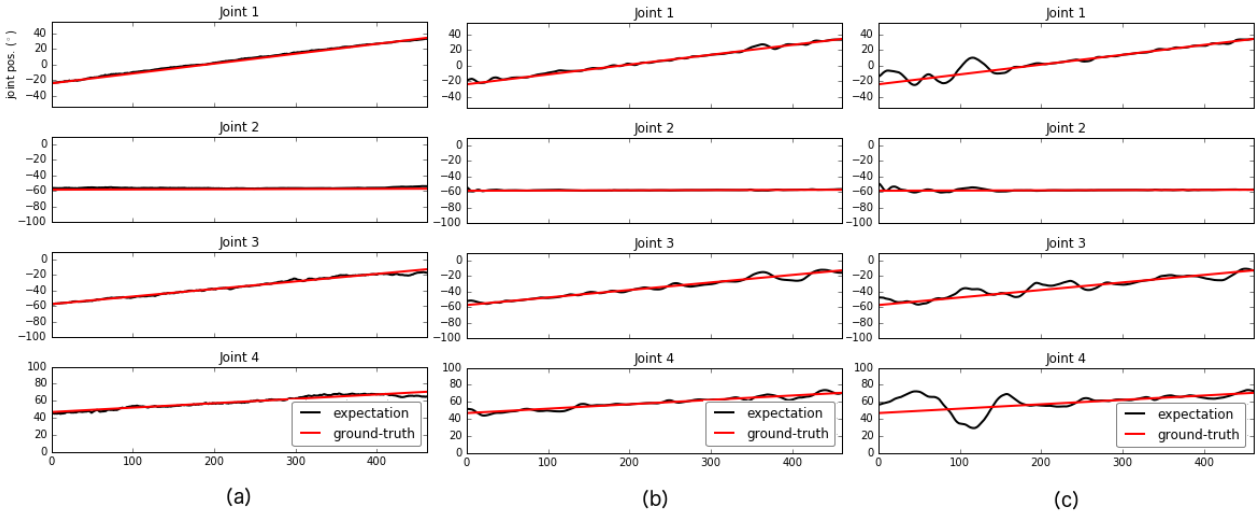

Fig. 4. Inferred joint state values using (a) the inverse model, and the forward sensor model in an EKF update for a (b) 5-degree and (c) 10-degree offset from ground-truth at $t = 0$.

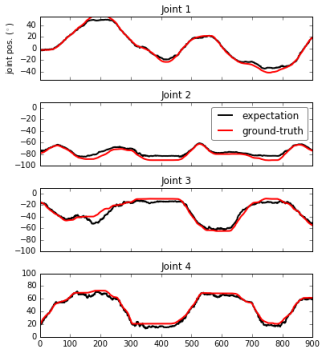

Fig. 5. Inferred joint state values from a sequence of RGB images using inverse sensor model for an arbitrary trajectory.

for tracking outperforms the filtered approach. However, in a scene containing many instances of partial or full object occlusion, an approach which models latent-state dynamics may be favorable over the use of a discriminative, inverse model.

V. Future Work

In training our models on state-observation pairs, we have assumed that ground-truth state measurements are reliably obtained from joint encoders through low-velocity operation. In future work, training data will be collected using a high-fidelity motion-capture system to collect ground-truth joint displacement values. New data sets will also be comprised of scenes containing partial or total occlusion of the manipulator, to further demonstrate the tracking performance of the framework.

References

[1] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*, 2014.

[2] Bruno Damas and José Santos-Victor. An online algorithm for simultaneously learning forward and inverse kinematics. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 1499–1506. IEEE, 2012.

[3] Anthony Dearden and Yiannis Demiris. Learning forward models for robots. In *IJCAI*, volume 5, page 1440, 2005.

[4] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2758–2766, 2015.

[5] Alexey Dosovitskiy, Jost Springenberg, Maxim Tatarchenko, and Thomas Brox. Learning to generate chairs, tables and cars with convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 2016.

[6] Aaron D'Souza, Sethu Vijayakumar, and Stefan Schaal. Learning inverse kinematics. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, volume 1, pages 298–303. IEEE, 2001.

[7] Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. In *Advances In Neural Information Processing Systems*, pages 64–72, 2016.

[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.