Learning Reachable Manifold and Inverse Mapping for a Redundant Robot manipulator

Seungsu Kim and Julien Perez

***Abstract*— Validating the kinematic feasibility of a planned robot motion and finding corresponding inverse solutions are time-consuming processes, especially for long-horizon manipulation tasks. Most existing approaches are based on solving iterative gradient-based optimization, so the processes are time-consuming and have a high risk of falling in local minima. In this work, we propose a unified framework to learn a kinematic feasibility model and a one-shot inverse mapping model for a redundant robot manipulator. Once they are trained, the models can compute the kinematic reachability of a target pose and its inverse solutions *without iterative process*. We validate our approach using a 7-DOF robot arm with an object grasping application.**

I. Introduction

One of the challenges for robot motion planning is validating the feasibility of planned robot motions within the robot's constraints (e.g., kinematic limits, geometric constraints, self-collision, etc.). One of the reasons is based on the time-consuming operations of solving optimization and the problem of falling on local minima. Such problems can be even harder for sequential robot manipulation planning, as kinematic and dynamic feasibility for each time step of the planned long-horizon trajectory should be validated. One of the approaches to tackle such an issue is modeling the kinematic feasibility manifold of a robot end-effector offline, and directly querying the feasibility from the model. In this context, a continuous reachable manifold learning approach [1], [2] is proposed to validate the feasibility of task space motions for a robot manipulator. Having such a reachable manifold in task space is very beneficial in several aspects. 1) A robot can plan its motion in task space without checking time-consuming iterative feasibility checks along a planned trajectory. Especially, for an action policy learning using reinforcement learning, discarding infeasible action space will reduce drastically the training iterations. 2) In object grasping tasks, this approach can provide an instant solution to discard infeasible grasping poses in terms of robot reachability with the respective robot's current base coordinate system. 3) Besides, it is a good indication to quantify the richness of solutions to place a robot end-effector to a target end-effector pose.

NAVER LABS Europe, 6 chemin de Maupertuis, Meylan, 38240, France seungsu.kim@naverlabs.com, julien.perez@naverlabs.com

Although such a reachable manifold provides a good indication of kinematic feasibility and richness of inverse mapping solutions for a given robot Cartesian space task, we have to compute configuration space values to control the robot. Indeed, such an inverse mapping problem is a classic problem in robotics. However, most of the adopted solutions are gradient (more specifically Jacobian) based iterative approaches in the velocity level. Also, the results are highly affected by initial joint configurations and null-space reference postures [3] for a redundant manipulator. Various learning approaches for this problem have been applied [4][5]. However, learning one-shot inverse mapping problem (to estimate joint configuration without iterative process for a target pose) for redundant manipulators are still challenging as is an one-to-many mapping. Indeed, there are infinite solutions (in joint configuration space) to put a robot end-effector to a specific pose for a redundant manipulator.

In this paper, we consider diversity of inverse solutions for a redundant manipulator to place its end-effector to a specific pose, i.e., multi-modal distribution. We propose a unified learning framework to validate the feasibility and to find diverse inverse mapping solutions. We train the models from a dataset generated from robot motor babbling in simulation which the robot explores its configuration space densely within its kinematic capability. As a result, the trained models can compute kinematic reachability of a target pose, and its inverse solutions *without iterative computation*. The contribution of the paper is in five folds:

- A unified learning framework to validate kinematic feasibility and to solve inverse mapping for a planned task space motion.
- It provides a reachability measure for a target pose in task space without an iterative process.
- It provides diverse and accurate inverse mapping solutions for a redundant manipulator.
- The approach is validated in a commercial robot platform in full position and orientation so that we can use the solution directly in various robotics tasks. Especially, we validated our approach in an object grasping task.
- As we train the models accurately using the samples which are generated by considering self-collision and the joint limit for a robot, the resultant models rarely violate self-collision and joint limit.

II. Related works

A. Reachable space estimation

Representing a space of a robot's feasible end-effector postures is essential to plan the robot's motion efficiently. The reachable (or kinematically feasible) space depends on the kinematic properties of the robot (e.g., degrees of freedom (DOF), joint limits and link lengths, etc.). Hence, the 6D reachable space spanned by the feasible positions and orientations of a robot is highly nonlinear and varies drastically from robot to robot.

Early investigations of this issue are based on discretized voxel [6], [7], [8], [9]. These approaches uniformly divide the complete reachable-space into 3D cells [6] or full 6D cells [7]. Each of the discretized cell contains reachability values. Indeed, such approaches provide an intuitive and accurate measure of reachability. However, for high resolution of the reachable space, high dimensional voxel grid is required.

The other body of approaches approximates the reachable-space of a robot with density functions. A data-driven probabilistic modeling approach has already been considered to compute the 6D reachable space of a robot hand [1], [2]. The approach uses a much more compact representation than the grid approach. Training samples of reachable space are acquired using motor babbling, i.e. randomly sampling joint configurations within the joint ranges of the robot. The resulting end-effector poses are computed from the known forward model. The density of the reachable space samples is modeled by the Gaussian mixture model (GMM). A specific end-effector pose is considered reachable if its reachability (measured as the probability density of the model) is larger than a threshold. High reachability correlates to many available inverse kinematics solutions and to a wide range of null-space (in case of a redundant manipulator). However, these approaches didn't consider self-collision nor redundant null space density.

B. Density estimation

One of the common approaches for density approximation from samples is Kernel Density Estimation, a non-parametric method to estimate probabilities for new points [10]. It has a very intuitive and simple structure to use, however, requires a large number of representative kernels to represent complex density distribution accurately.

Another category of the approach is Energy based model (EBM) [11][12][13]. It learns an energy function by assigning low energies to the observed data and high energies to the others. Thus, it can estimate multi-modal and complex density distributions of the data explicitly. The energy function is defined as a scalar function which is often parameterized by a neural network. The parameters of the energy function are generally learned via maximum likelihood estimation or minimizing the Kullback-Leibler (KL) divergence between the training samples and the trained model.

The other category of the approach is based on autoregressive flow (AF) [14][15][16][17]. AF uses autoregressive models as a form normalizing flow [18]. It transforms a standard normal distribution into a more complex distribution by the composition of invertible and differentiable functions [14]. The joint density $p(\xi)$ of a multivariate random variable ξ can be represented as a product of univariate conditional densities as $p(\xi) = p(\xi_1) \prod_{i=2} p(\xi_i|\xi_{1:i-1})$. Autoregressive density estimators model each conditionals $p(\xi_i|\xi_{1:i-1})$ as a parametric density. The model is trained to minimize the KL divergence of the model from the observed data [1].

C. Inverse mapping for redundant manipulator

Above density estimation approaches can be applied to estimate the density of feasible robot end-effector poses so that the density model provides an indication of feasibility and richness of possible solutions for a target end-effector pose. However, in order to make a robot to perform a task in Cartesian space, we need to find a set of values in joint configuration space (e.g., joint position, velocity, or torque) for the task space trajectory. Such an inverse mapping is not a trivial problem as a redundant manipulator admits an infinite number of solutions for the desired end-effector pose. In addition, joint limits, self-collisions, or more constraints (e.g., energy consumption, external obstacles, etc.) should be considered.

One of the most commonly used approaches is based on the robot Jacobian [19]. The inverse kinematics problem can be formulated as a gradient descent optimization problem using Jacobian. It intrinsically finds a solution in the closest distance from the initial joint configuration. For a redundant manipulator, null-space of the Jacobian [3] approach is generally used. It uses redundant DOFs for other user-defined requirements after satisfying the task constraints. This method can be used to avoid joint limits [3], for obstacle avoidance [20] and for prioritized tasks (the lower priority task only in the null space of the higher-priority task) [21]. These approaches are computationally very fast and effective when a robot is asked to do a manipulation task with additional constraints. However, the solutions are highly affected by the initial joint configuration. More importantly, these types of approaches have a high risk of falling into local minima.

One of the recent approaches in machine learning, [22] proposed an inverse mapping learning approach using deep neural networks. Unlike Jacobian based approach, it provides a direct mapping between end-effector pose $\mathbf{x}$ and joint configuration values $\mathbf{q}$. The aim is to approximate the inverse mapping $p(\mathbf{q}|\mathbf{x})$ of a dataset that are sampled from a known forward model $\mathbf{x} = f(\mathbf{q})$. For a redundant case, they employ a latent random variable $\mathbf{z}$, and approximate the inverse mapping as $\mathbf{q} = g(\mathbf{x}, \mathbf{z})$. The networks are trained to minimize 1) the forward prediction loss (i.e., end-effector pose estimation loss), 2) the backward prediction loss (i.e., joint configuration estimation loss) and 3) the latent variable loss ($\mathbf{z}$ to be normal distribution, and to be independent to $\mathbf{x}$ i.e., $p(\mathbf{z}|\mathbf{x}) = p(\mathbf{z})$ so that the network not encode twice the training information).

[1] Minimizing KL divergence is the same effect maximizing log-likelihood of observation [16]

In this work, we employ the work of [22] for learning the inverse mapping. We extend the work to full 6D Cartesian space and validate it on a commercial robot manipulator. Besides, we propose a unified learning framework for training the reachability density model and inverse mapping model jointly. Once they are learned, the networks directly provide a joint space solution for a given robot end-effector pose, without solving optimization problems recursively (e.g., Jacobian based inverse kinematics). It is highly beneficial for long-horizon path planning for robot manipulation. The detail will be followed in the next Section III-B

III. Method

In this paper, we propose an unified framework for feasibility validation of a target end-effector pose (in Section III-B) and for solving its inverse solutions (in Section III-A). We train the models from a dataset, $\{\mathbf{x}_i, \mathbf{q}_i\}_{i=1}^{N}$, generated using a motor babbling procedure in simulation which the robot explores its configuration space densely within its kinematic capability. We sample from a uniform distribution in joint configuration space. Where $\mathbf{x} \in SE(3)$ represents end-effector pose variable in Cartesian space [2]; $\mathbf{q} \in \mathbb{R}^D$ represents joint configuration space variable. We assume that the kinematic properties (kinematic chain, joint limits, robot collision shapes, etc.) of the target robot are available.

A. Learning inverse mapping with latent representation

Since we consider redundant inverse mapping problem (i.e., one-to-many mapping or under-determined problem), there are infinite number of solutions in configuration space to place a robot end-effector to a feasible pose. Hence, we propose to employ a latent representation to encode the diversity of inverse solutions for a target end-effector pose. We aim to find a deterministic inverse mapping function $\mathbf{q} = \mathcal{I}(\mathbf{x}, \mathbf{z})$. Where $\mathbf{z} \in \mathbb{R}^K$ represents the latent space variable.

The architecture of the proposed networks is shown in Fig. 1. The forward network, $\{\mathbf{x}, \mathbf{z}\} = \mathcal{F}(\mathbf{q})$, estimates the end-effector pose $\mathbf{x}$ and latent representation $\mathbf{z}$ for a given joint configuration $\mathbf{q}$. To encode the diversity of the solutions in the latent space, the dimension of the latent variable K should be greater than or equal to the redundant degrees of freedom, $K \geq D - 6$.

The inverse net $\mathcal{I}$ estimates a set of joint configurations for a given end-effector pose $\mathbf{x}$ and a latent variable $\mathbf{z}$. The reachable density network $\mathcal{D}(\mathbf{x}, \mathbf{z})$ estimate joint density for given $\mathbf{x}$ and $\mathbf{z}$.

The first two networks, $\mathcal{I}$ and $\mathcal{F}$ are jointly trained by minimizing below three losses, with fixed density network $\mathcal{D}$.

- $\mathcal{L}_q(\mathbf{q}, \hat{\mathbf{q}})$: Joint configuration reconstruction loss
- $\mathcal{L}_c([\mathbf{x}, \hat{\mathbf{z}}], [\bar{\mathbf{x}}, \bar{\mathbf{z}}])$: End-effector pose and latent variable reconstruction loss

[2]The selection of orientation representation affects to the training result drastically because of the discontinous issue of orientation represenataion. We use first two axes of rotation matrix used in [1] to represent orientation.

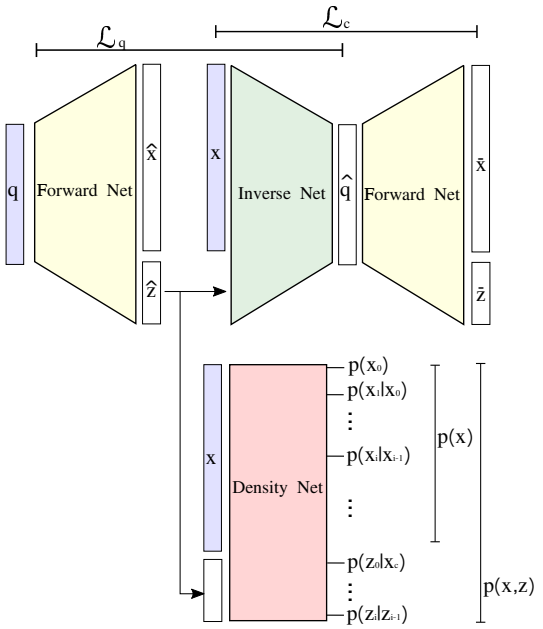


Fig. 1: The architecture of the proposed network models. The details of the forward/inverse networks and the density network are described in Section III-A and III-B respectively.

- $\mathcal{L}_z = KL(p(\mathbf{z})|\mathcal{N}(0, \Sigma)) + KL(p(\mathbf{x}, \mathbf{z})|p(\mathbf{x})p(\mathbf{z}))$: The latent loss is consisted with two losses. The first one is that the density of the latent variable $\mathbf{z}$ to be shaped as a normal distribution. The other is that the mutual information between $\mathbf{x}$ and $\mathbf{z}$ to be zero. This loss forces $\mathbf{x}$ and $\mathbf{z}$ to be independent so that the forward model does not encode same information twice in $\mathbf{x}$ and $\mathbf{z}$ [22]. For the mutual information loss, we also use KL divergence. Where $p(\mathbf{x}, \mathbf{z})$ and $p(\mathbf{x})$ are computed from the density model $\mathcal{D}$ that is presented in the next section III-B.

In this paper, mean squared error (MSE) is used to compute $\mathcal{L}_q$ and $\mathcal{L}_c$. The total loss is computed by weighted sum of the individual losses, $\mathcal{L} = \Sigma_{l=1}^{L} \lambda_l \mathcal{L}_l$, where λ are corresponding weight for each of individual losses. All the parameters that are used for training these networks are shown in Section IV.

B. Learning reachability manifold

In this section, we present our density estimation approach for a robot end-effector pose and latent space variable. Among the approaches reviewed in Section II-B, we use Block Neural Autoregressive Flows (B-NAFs) [16], as it provides a compact and universal density approximation model as well as outstanding performance on the density approximation. Unlike neural autoregressive flow [23] use conditioner network, B-NAFs train the network directly [16].

The density network $\mathcal{D}$ estimates the joint density for a random variable $\boldsymbol{\xi} = [\mathbf{x}, \mathbf{z}]$.

$$p(\boldsymbol{\xi}) = p(\xi_1) \prod_{i=2} p(\xi_i | \xi_{1:i-1}) \tag{1}$$

Each of the conditional densities is modeled individually using dense networks. By employing a single masked autoregressive network [24], all the conditional density functions are efficiently computed in parallel. The joint probabilities, $p(\mathbf{x})$ and $p(\mathbf{x}, \mathbf{z})$ are computed by the chain rule of probability, from the density network $\mathcal{D}$ as seen in Equation 1.

The model is trained by minimizing the loss function $\mathcal{L}_d = KL(p_s(\mathbf{x}, \mathbf{z})|p(\mathbf{x}, \mathbf{z}))$, KL-divergence between source distribution $p_s(\mathbf{x}, \mathbf{z})$ of training samples and the target distribution $p(\mathbf{x}, \mathbf{z})$ of the density network $\mathcal{D}$. Once the density model is trained, a given pose $\mathbf{x}$ or $[\mathbf{x}, \mathbf{z}]$ are said to be feasible when the density $p(\mathbf{x})$ or $p(\mathbf{x}, \mathbf{z})$ exceeds the reachable density thresholds, ρ_x and ρ_c respectively. In addition, high density indicates the richness of its mappings to joint configuration space for the end-effector pose.

The overview training procedure are described in below algorithm.

Algorithm 1: Training procedure

Data: $\{\mathbf{x}_i, \mathbf{q}_i\}_{i=1}^N$ from motor babbling
while *numer of epoch* **do**
  // train forward and inverse networks
  $(\_, \hat{\mathbf{z}}) = \mathcal{F}(\mathbf{q})$
  $\hat{\mathbf{q}} = \mathcal{I}(\mathbf{x}, \hat{\mathbf{z}})$
  $(\bar{\mathbf{x}}, \bar{\mathbf{z}}) = \mathcal{F}(\hat{\mathbf{q}})$
  compute $p(\mathbf{x})$, $p(\mathbf{x}, \hat{\mathbf{z}})$ from $\mathcal{D}$
  compute losses $\mathcal{L}_q$, $\mathcal{L}_c$ and $\mathcal{L}_z$
  optimizer step for $\mathcal{F}$ and $\mathcal{I}$
  // train density network
  $(\_, \hat{\mathbf{z}}) = \mathcal{F}(\mathbf{q})$
  compute loss $\mathcal{L}_d$
  optimizer step for $\mathcal{D}$
end

IV. Experimental validation

We first demonstrate the capability of the proposed framework on a commercial robot manipulator, Panda from Franka Emika company[3], in Section IV-A. In addition, we demonstrate a typical application of object-grasping task, selecting a kinematically feasible grasp pose among diverse candidates (in Section IV-B).

A. Reachable-space and Inverse mapping of a robot manipulator

a) Training and validation dataset generation: Training samples $\{\mathbf{x}_i, \mathbf{q}_i\}_{i=0}^N$ are acquired using motor babbling by randomly sampling in a uniform distribution of joint configuration space within the joint ranges of the robot. The resulting end-effector poses $\mathbf{x}$ are computed from the known forward kinematics. In addition to that, we also check self-collision [25] and discard the self-collided samples from the data-set. Total $1e8$ samples are generated and randomly selected 80% is used for training and others are used for testing.

[3]https://www.franka.de/

For the validation dataset, we generate kinematically infeasible samples as well. Generating infeasible samples requires a more complicated process than the feasible sample generation, as the samples should be validated analytically or probabilistically. A pose in SE(3) is randomly selected within the limited position envelope (1.5<x, y, and z <1.5 unit: m); an orientation is randomly selected but without any restrictions. Note that sampling space should include the whole reachable space of the robot with some margin. Each pose is tested by solving the inverse kinematics to validate its feasibility. A general Jacobian-based IK solver based on Newton-Raphson iterations [26] is used. As the initial choice of joint angles affects the solution, for each target pose, 200 different initial joint angles are chosen randomly within the joint ranges of the robot. We set the maximum iterations for the IK at 200. If all IK attempts are failed within the iteration limit, the pose is considered as negative, otherwise positive. Total $1e5$ samples were generated, and 6.8% were fell into feasible and the rest 93.2% were fell into infeasible. As we consider full range of orientation for the end-effector, the feasible sample ratio in SE(3) is very small.

b) Network architecture details and training: For the forward $\mathcal{F}$ and inverse $\mathcal{I}$ models, we use a fully connected neural network with 6 and 9 layers respectively. The dimensions of all the hidden layers are set to 500. The density model $\mathcal{D}$ consists of 4 layers with 66 hidden dimensions. The models are trained with Adam optimizer using learning rates of 0.0001 for $\mathcal{F}$ and $\mathcal{I}$ networks, and 0.005 for $\mathcal{D}$. The parameters have been validated through cross-validation.

For the joint configuration reconstruction loss $\mathcal{L}_q(\mathbf{q}, \hat{\mathbf{q}})$ and the end-effector pose and latent variable reconstruction loss $\mathcal{L}_c([\mathbf{x}, \hat{\mathbf{z}}], [\bar{\mathbf{x}}, \bar{\mathbf{z}}])$, we simply use MSE. Batch size is set as $1e5$.

c) Result: Figure 3 shows loss curves which are computed from the testing and validation sets. The reachable density threshold ρ_x is selected 99% of training samples to be above the threshold at every epoch. The threshold is used at the next epoch to compute the true positive rate (TPR) and true negative rate (TNR). After the training is finished, we got TPR and TNR as 0.99 and 0.95 respectively. As the reachable density model is too high dimensional to display the resultant density in 2D or 3D, we show a few examples in 2D (x-y plane) by fixing the other values arbitrary in Figure 4. We see that different orientation constraints highly affect the reachable space in the x-y position plane.

Figure 2 shows the diverse solutions for a given target end-effector pose. The middle of the figure shows the probability density $p(\mathbf{x}, \mathbf{z})$ in latent space for a given end-effector pose. Seven arbitrary latent values (which the reachability score is higher than the threshold) are selected, and their joint configurations are directly computed from the inverse net $\mathcal{I}$. We see that the inverse net with latent representation provides diverse inverse solutions.

To check if the resultant inverse mapping solutions from $\mathcal{I}$ contain self-colliding samples, we evaluate the inverse solutions of the testing set using the FCL library [25]. As a result, 99.7 % were safe from the self-collision.

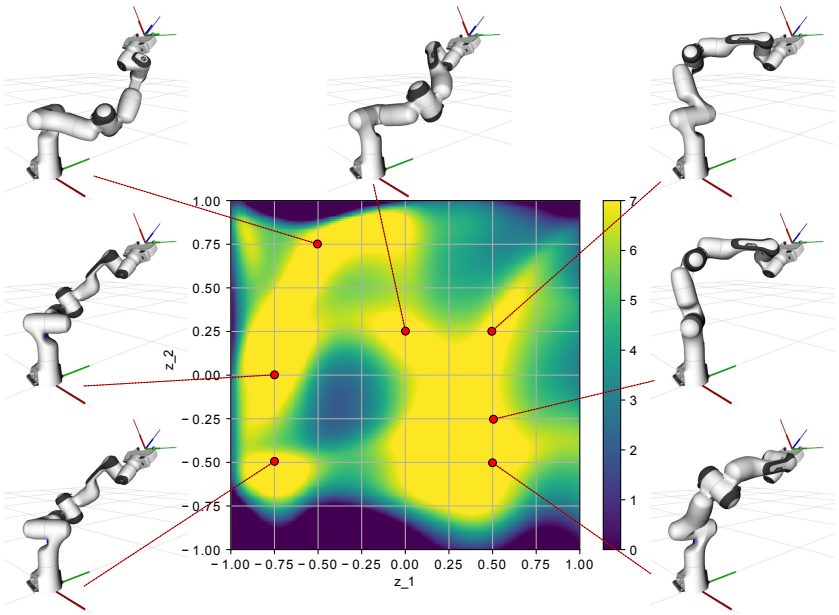


Fig. 2: Probability density $p(\mathbf{x}, \mathbf{z})$ plot on latent space in log scale. An end-effector pose in the testing set is set as the target pose: position $(0.200, 0.58, 0.77)$ and orientation $(-0.30, -0.30, -0.63, 0.65)$ in scalar-last format of unit Quaternion. Seven latent variables are tested with the trained inverse net. The resultant joint configurations are displayed with the Panda robot. The inverse network generates diverse solutions in joint configuration space for the target end-effector pose.

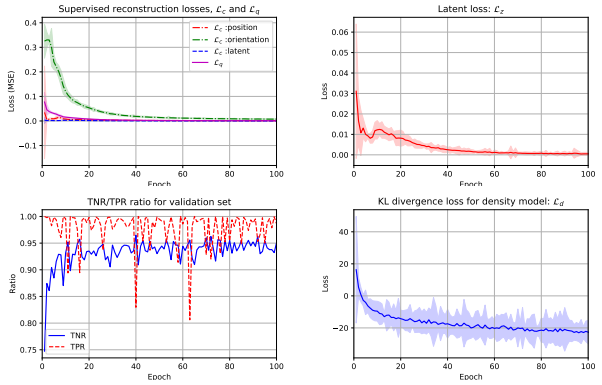


Fig. 3: The loss curves during the training. The supervised losses $\mathcal{L}_q$ and $\mathcal{L}_c$ (left top), latent loss $\mathcal{L}_z$ (right top) and density loss $\mathcal{L}_d$ (right bottom) are computed from the testing set for each epoch. The figure in left bottom shows the density model accuracy curve (TPR and TNR) which is computed from the validation set.

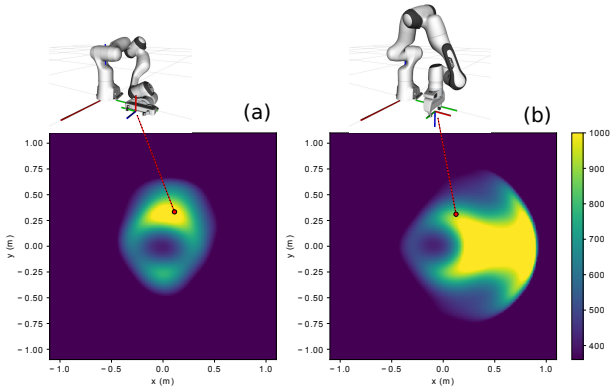


Fig. 4: Reachable density $p(\mathbf{x})$ on x-y plane with different orientations, (a) $[0.707, 0.0, 0.707, 0.0]$ and (b) $[0.707, 0.707, 0.0, 0.0]$ in scalar last format of unit quaternion. Z is fixed as $z = 0.01$. Two examples of inverse solution are displayed with colored axis (red: x axis, green: y axis and blue: z axis of rotation matrix).

Indeed, the inverse mapping solutions still include some errors in Cartesian space [4] although they are small. For a task that requires high-resolution, we add a refinement process on the resultant inverse mapping solution. The refinement process includes one or two iterations of Jacobian based IK approach. Initial and rest joint configurations are set with the output of the inverse network. As we see in Figure 5, the refinement process helps to reduce the remaining end-effector pose error.

The online computation time for an inverse mapping of the method is 0.51 ± 0.17 ms for a single query and 2.30 ± 0.50 ms for a batch of 512 samples [5]. Indeed, it is not directly comparable with other approaches (e.g. TRAC-IK [27]), as ours get benefits from GPU, but most of the other approaches are implemented to use CPU. However, obviously ours is highly beneficial for robot-learning applications that are implemented to use GPUs.

[4]The Cartesisan space pose for the resultant inverse solution is computed by using the true forward kinematics, and it is compared with the one in the testing set.

[5]The computation is performed on Nvidia Quadro P1000

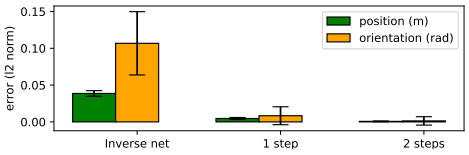


Fig. 5: Refinement process result. One and two iterations of Jacobian based IK is applied, by setting the initial and rest joint configurations as the output of trained inverse network.

B. Grasping pose estimation

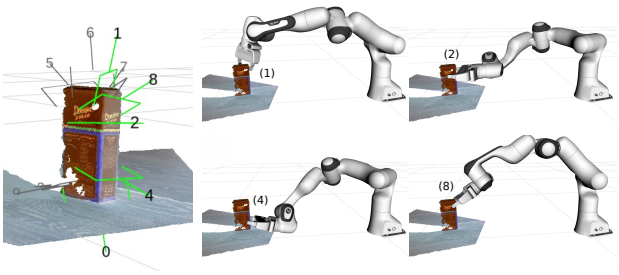
Generating a successful grasping pose is an essential but crucial component for robot object manipulation tasks. Numerous approaches have been proposed in this domain [28][29][30]. However, most of them are focused on finding diverse robot hand poses with respect to the *object coordinate system*. Deciding to select an optimal grasping configuration among them highly depends on the target task and kinematic feasibility for a target robot. Indeed, the kinematic feasibility validation is an essential process for successful grasping, and a conventional way such as diverse trial of IK for different initial configurations (that are used for validation dataset generation in Section IV-A) might be an option. However, as we discussed in Section III-A, it is a time-consuming process and has a risk of falling on a local optima. In contrast, our presented approach directly provides the kinematic feasibility and corresponding joint configurations together without iterative process.

Among the different approaches for generating diverse grasping candidates w.r.t. the object coordinate system, we use 6-DOF Graspnet [28], as it provides diverse grasping gripper candidates, and its performance (success rate for picking up objects) is with state-of-the-art performances. In this experiment, we use one of the objects (see Figure 6), pre-trained model and implementation from [28][31]. As investigating and reviewing diverse approaches for grasping pose estimation is out of scope of this paper, we invite the reader to refer to the mentioned papers for exhaustive details about the approach.

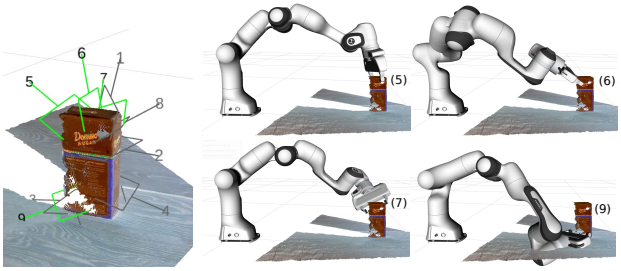
Figure 6 shows the kinematic feasibility validation and direct inverse mapping result of our presented approach. Among the diverse grasping poses for the object produced by 6-DOF Graspnet, we select 10 best grasping candidates only for good visualization. Our density model $\mathcal{D}$ evaluates the candidates in terms of kinematic feasibility, and the inverse model $\mathcal{I}$ generates robot joint configurations for the gripper poses (latent variable are set as $[0, 0]$). As we see in the figure, the density net $\mathcal{D}$ directly discards the grasp configurations which are kinematically infeasible for the robot manipulator. Also, our approach directly generates joint configuration for the gripper pose.

V. Discussion and Conclusion

In this paper, we presented a unified learning framework to validate the feasibility of a desired end-effector pose and to compute its diverse inverse mapping solutions, which are



(a) Object position : (0.8, 0.0, 0.0)



(b) Object position : (-0.8, 0.0, 0.0)

Fig. 6: Depend on the pose of the object w.r.t. the robot base coordinate system, the feasibilities for the grasping candidates are varied drastically (green: feasible, gray: invisible). The presented approach accurately and efficiently filters out the kinematically infeasible candidates (left), and provides joint configurations without iterative process (right). The numbers near the gripper represent the ids of the grasping candidates.

essential elements for robot arm path planning. Emphasis is placed on the presented approach directly providing the *kinematic feasibility* and *diverse inverse solutions* for a given task space target *without iterative process* in the execution stage.

Compared to the iterative gradient based approach for solving inverse kinematics, the presented approach provides a kinematic feasibility and its inverse solutions without iterative optimization. Hence the presented approach is computationally very efficient and it can reduce the risk of falling in a local minima. In addition, our models are accurately trained using the samples which are generated by considering self-collision. We show that the resultant models rarely violate self-collision.

The presented latent representation is conceptually similar to the null-space motion of Jacobian in the classic Inverse kinematics in robotics. However, the presented approach has minimum representation in the latent space, and directly provides a joint position space solution rather than the local velocity space.

As we see in Figure 5, the inverse mapping solutions still include some errors although they are very small. For a task that requires high accuracy, an additional refinement process (in Section IV-A) can be applied optionally to remove the remaining end-effector pose error.

In the future work, we are going to apply the presented framework to high-level policy learning and reinforcement learning tasks to reduce the search space for an action policy by efficiently discarding the infeasible action space in advance. Finally, we plan to extend the latent representation to encode useful information for robot manipulation, such as manipulability [32].

REFERENCES

[1] S. Kim, A. Shukla, and A. Billard, "Catching Objects in Flight," *IEEE Transactions on Robotics*, vol. 30, no. 5, pp. 1049–1065, 2014.

[2] S. Kim, R. Haschke, and H. Ritter, "Gaussian Mixture Model for 3-DoF orientations," *Robotics and Autonomous Systems*, vol. 87, pp. 28–37, Jan. 2017.

[3] A. Liegeois, "Automatic Supervisory Control of the Configuration and Behavior of Multibody Mechanisms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 7, no. 12, pp. 868–871, 1977.

[4] A. D'Souza, S. Vijayakumar, and S. Schaal, "Learning inverse kinematics," in *IROS*, vol. 1. IEEE, 2001, pp. 298–303.

[5] M. Rolf, J. J. Steil, and M. Gienger, "Goal babbling permits direct learning of inverse kinematics," *IEEE Transactions on Autonomous Mental Development*, vol. 2, no. 3, pp. 216–229, 2010.

[6] F. Zacharias, C. Borst, and G. Hirzinger, "Object-Specific Grasp Maps for Use in Planning Manipulation Actions," in *Advances in Robotics Research*. Springer Berlin Heidelberg, 2009, pp. 203–213.

[7] F. Zacharias, C. Borst, S. Wolf, and G. Hirzinger, "The Capability Map: A Tool to Analyze Robot Arm Workspaces," *International Journal of Humanoid Robotics*, vol. 10, no. 04, p. 1350031, Dec. 2013.

[8] O. Porges, R. Lampariello, J. Artigas, A. Wedler, C. Borst, and M. A. Roa, "Reachability and Dexterity: Analysis and Applications for Space Robotics," in *Workshop on Advanced Space Technologies for Robotics and Automation (ASTRA)*, 2015, p. 7.

[9] N. Vahrenkamp, H. Arnst, M. Wachter, D. Schiebener, P. Sotiropoulos, M. Kowalik, and T. Asfour, "Workspace analysis for planning human-robot interaction tasks," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. Cancun, Mexico: IEEE, 2016, pp. 1298–1303.

[10] S. Kim, A. Coninx, and S. Doncieux, "From exploration to control: learning object manipulation skills through novelty search and local adaptation," *Robotics and Autonomous Systems*, vol. 136, 2021.

[11] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. J. Huang, "A Tutorial on Energy-Based Learning," in *Predicting Structured Data*, 2006.

[12] J. Xie, Y. Lu, S.-C. Zhu, and Y. N. Wu, "A Theory of Generative ConvNet," in *ICML*, 2016.

[13] J. Zhao, M. Mathieu, and Y. LeCun, "Energy-based Generative Adversarial Network," in *ICLR*, 2017.

[14] I. Kobyzev, S. J. D. Prince, and M. A. Brubaker, "Normalizing Flows: An Introduction and Review of Current Methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[15] G. Papamakarios, "Neural Density Estimation and Likelihood-free Inference," *arXiv:1910.13233 [cs, stat]*, Oct. 2019.

[16] N. De Cao, I. Titov, and W. Aziz, "Block Neural Autoregressive Flow," in *Uncertainty in Artificial Intelligence*, 2019.

[17] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling, "Improved Variational Inference with Inverse Autoregressive Flow," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, Inc., 2016, pp. 4743–4751.

[18] D. J. Rezende and S. Mohamed, "Variational Inference with Normalizing Flows," in *ICML*, 2016.

[19] K. M. Lynch and F. C. Park, *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press, 2017.

[20] N. Yoshihiko, *Advanced Robotics: Redundancy and Optimization*. Addison-Wesley Publishing Company, 1991.

[21] N. Yoshihiko, H. Hideo, and Y. Tsuneo, "Task-Priority Based Redundancy Control of Robot Manipulators," *The International Journal of Robotics Research*, vol. 6, no. 2, pp. 3–15, 1987.

[22] L. Ardizzone, J. Kruse, S. Wirkert, D. Rahner, E. W. Pellegrini, R. S. Klessen, L. Maier-Hein, C. Rother, and U. Köthe, "Analyzing Inverse Problems with Invertible Neural Networks," in *ICLR*, 2019.

[23] C.-W. Huang, D. Krueger, A. Lacoste, and A. Courville, "Neural Autoregressive Flows," in *International Conference on Machine Learning*, 2018.

[24] M. Germain, K. Gregor, I. Murray, and H. Larochelle, "MADE: Masked Autoencoder for Distribution Estimation," in *ICML*, 2015, pp. 881–889.

[25] J. Pan, S. Chitta, and D. Manocha, "FCL: A general purpose library for collision and proximity queries," in *ICRA*, 2012, pp. 3859–3866.

[26] A. Goldenberg, B. Benhabib, and R. Fenton, "A complete generalized solution to the inverse kinematics of robots," *IEEE Journal on Robotics and Automation*, vol. 1, no. 1, pp. 14–20, 1985.

[27] P. Beeson and B. Ames, "TRAC-IK: An open-source library for improved solving of generic inverse kinematics," in *IEEE-RAS International Conference on Humanoid Robots*. Seoul, South Korea: IEEE, Nov. 2015, pp. 928–935.

[28] A. Mousavian, C. Eppner, and D. Fox, "6-DOF GraspNet: Variational Grasp Generation for Object Manipulation," in *ICCV*, 2019.

[29] H.-S. Fang, C. Wang, M. Gou, and C. Lu, "GraspNet-1Billion: A Large-Scale Benchmark for General Object Grasping," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Seattle, WA, USA: IEEE, June 2020, pp. 11 441–11 450.

[30] L. Pinto and A. Gupta, "Supersizing Self-supervision: Learning to Grasp from 50K Tries and 700 Robot Hours," *ICRA*, 2016.

[31] J. Lundell, "6-DOF GraspNet Pytorch," *https://github.com/jsll/pytorch_6dof-graspnet*, 2020.

[32] N. Jaquier, L. Rozo, D. G. Caldwell, and S. Calinon, "Geometry-aware manipulability learning, tracking, and transfer," *The International Journal of Robotics Research*, Aug. 2020.