IEEE Robotics and Automation Letters (RAL) paper presented at the
2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Neural Manipulation Planning on Constraint Manifolds

Ahmed H. Qureshi, Jiangeng Dong, Austin Choe, and Michael C. Yip

***Abstract*— The presence of task constraints imposes a significant challenge to motion planning. Despite all recent advancements, existing algorithms are still computationally expensive for most planning problems. In this paper, we present Constrained Motion Planning Networks (CoMPNet), the first neural planner for multimodal kinematic constraints. Our approach comprises the following components: i) constraint and environment perception encoders; ii) neural robot configuration generator that outputs configurations on/near the constraint manifold(s), and iii) a bidirectional planning algorithm that takes the generated configurations to create a feasible robot motion trajectory. We show that CoMPNet solves practical motion planning tasks involving both unconstrained and constrained problems. Furthermore, it generalizes to new unseen locations of the objects, i.e., not seen during training, in the given environments with high success rates. When compared to the state-of-the-art constrained motion planning algorithms, CoMPNet outperforms by order of magnitude improvement in computational speed with a significantly lower variance.**

I. INTRODUCTION

Efficient and scalable manipulation planning is of paramount importance in robotics and automation to solve real-world tasks. However, in most cases, manipulation of objects imposes hard kinematic constraints on the robot that limit its allowable range of motion. Examples of such instances include moving an object from one place to another that might also contain orientation constraints [1], maintaining contact with the environment such as in robot surgery [2], and performing bi-manual manipulation [3]. In all of these cases, kinematic constraints form one or more low-dimensional manifolds of a set of robot configurations embedded in a high-dimensional ambient space, and are often known as manifold constraints [4].

Sampling-based Motion Planning (SMP) algorithms have emerged as a standard tool in robotics that find collision-free paths between the given states by randomly sampling the robot configuration space [5]. However, incorporating various kinematic constraints into the SMP algorithms is challenging as the underlying constraint manifold is usually of zero-measure. Therefore, the probability of generating samples on the constraint manifold by randomly sampling robot joint-values/configurations is not just low but zero [4]. Recently, SMP methods have been extended to plan under manifold constraints on various challenging problems [4]. However, despite these advancements, the existing techniques are computationally inefficient and therefore, frequently impractical for real-world manipulation tasks.

A. H. Qureshi, J. Dong, A. Choe, and M. C. Yip are affiliated with the Department of Electrical and Computer Engineering, University of California San Diego, USA. J. Dong and A. Choe contributed equally as second authors. {alqureshi, jid103, achoe, yip}@ucsd.edu.

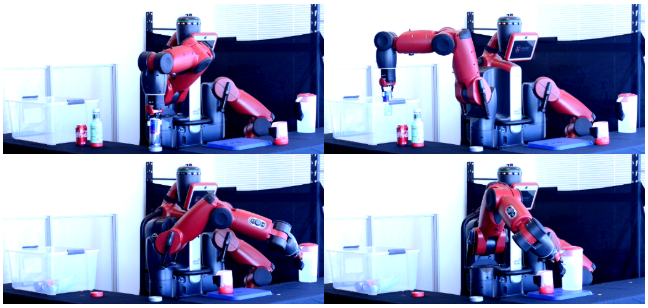


Fig. 1: CoMPNet applied to a real 7DOF Baxter robot manipulator in the bartender environment. The task is to place the bottles and cans to the trash, and carefully move (without tilting) the mug and pitcher to the tray. The top and bottom rows show the start and goal states at the beginning and end of this table cleaning task.

Recent developments also lead to imitation-based planners that learn to plan by imitating an oracle planner [6][7][8][9][10]. These planners are known for their extremely fast computational speed during online planning. Motion Planning Networks (MPNet) [10][11] is one of the learning-based planners that can generate end-to-end collision-free paths and is also combined with SMPs for worst-case theoretical guarantees. MPNet is a deep neural networks-based bidirectional iterative planning algorithm and it is shown to demonstrate consistently better performance than state-of-the-art SMP methods, e.g., [12], in challenging motion planning problems. However, MPNet, along with extensions to it, only consider unconstrained planning problems.

In this paper, we propose Constrained Motion Planning Networks (CoMPNet)[1] that extends MPNet to plan under multimodal kinematic and task-specific constraints. Our proposed framework is a full-stack, computationally-efficient planner that solves motion planning problems for both reach and manipulation tasks, i.e., reaching to grasp arbitrary object and manipulating the grasped objects under various kinematic constraints. To the best of our knowledge, CoMPNet is the first learning-based planning algorithm that finds feasible paths under multiple hard kinematic constraints. CoMPNet comprises the observation (environment perception) and task (constraint) encoders whose outputs are given to the neural planning network that, together with the bidirectional planning algorithm, generates a feasible path on the constraint manifolds between the given start and goal configurations.

[1]Supplementary material and video demonstrations are available at https://sites.google.com/view/constrainedmpnet/home

We evaluate our method on challenging tasks that include both simulations and a real-robot setup (Fig. 1). Our results show that CoMPNet outperforms existing methods in terms of computation speed and generalizes to new planning problems outside its training demonstrations with high success rates.

II. Related Work

In this section, we present a brief overview on existing non-sampling- and sampling-based approaches specifically addressing constraint motion planning (CMP), which represent a very challenging subset of planning problems with wide applications to robotics.

In non-sampling-based planning methods, one of the prominent tools is Trajectory Optimization (TrajOpt) [13][14]. TrajOpt relaxes the hard constraints into soft constraints by defining them as penalty functions and optimizing them over the entire trajectory to find a motion plan. Due to this relaxation, TrajOpt methods weakly satisfy the given constraints and do poorly on long-horizon problems. Bonalli et al. [15] extends TrajOpt to implicitly-defined constraint manifolds. However, their performance in challenging robotics problems is yet to be explored and analyzed.

In sampling-based planning approaches, the multi-query Probabilistic Road Maps (PRMs) [16] and single-query Rapidly-exploring Random Trees (RRTs) [17] and their variants [18] are widely known. These methods were initially devised for unconstrained problems such as collision avoidance, and incorporating kinematic constraints into them is challenging due to zero probability of sampling constraint satisfying configurations [4]. To address this challenge, there exist projection- and continuation-based strategies that generate samples on the constraint manifolds for SMP methods.

The projection-based methods project the given configurations to the constraint manifolds using the constraint equations. Typically, projections are performed using Inverse Kinematics (IK)-based iterative solvers that employ Jacobian (pseudo-) inverses of the robot model. These projection-based approaches have been used for special cases such as closed-chain kinematic constraints [19] as well as for general end-effector constraints [20]. In a similar vein, Berenson. et al. [21] proposed CBiRRT, a bidirectional RRT planner that uses general constraint representation known as Task Space Regions (TSRs) together with a Jacobian pseudo-inverse projection operator for CMP.

The continuation-based methods compute a tangent space from a known constraint-satisfying configuration to locally parametrize the underlying constraint manifold. The new constraint-satisfying configurations and local motions are generated by projecting the configurations sampled from the computed tangent space. In [20], the idea of continuation has been used to sample the neighborhood of a constraint-satisfying configuration, and then project them to the constraint manifold. Recent advancements also led to bidirectional RRT-based algorithms called Atlas-RRT [22] and Tangent Bundle (TB)-RRT [23] that compose tangent spaces into an atlas instead of discarding them to represent the constraint manifold. Atlas-RRT computes half-spaces to separate tangent spaces into tangent polytypes for uniform coverage. In contrast, TB-RRT lazily evaluates configurations for constraint adherence and does not separate tangent spaces, leading to overlap and sometimes invalid states.

A very recent work called Implicit MAnifold Configuration Space (IMACS) [24] decouples constraint adherence methods like projection and continuation from the choice of underlying motion planning algorithms. Although their approach allows a broad range of SMP algorithms to operate under kinematic constraints, ultimately, a classic bidirectional RRT is preferred due to the poor computational speed of advance methods like RRT* [18] and its variants [12], [25], [26] in CMP.

In contrast to the methods mentioned above, CoMPNet leverages past planning experiences for learning a deep neural model and generates samples on the implicit manifolds during execution to quickly find a path solution for both constrained and unconstrained planning problems. Furthermore, our approach can also be combined with existing sampling-based CMP methods for worst-case theoretical guarantees while still retaining the computational benefits.

III. Problem Definition

Let $\mathcal{C} \in \mathbb{R}^d$ be a d-dimensional configuration space (C-space) where $c \in \mathcal{C}$ represents a robot's configuration such as joint angles. Since the robot surroundings usually contain obstacles, the C-space comprises obstacle ($\mathcal{C}_{obs}$) and obstacle-free ($\mathcal{C}_{free} = \mathcal{C} \backslash \mathcal{C}_{obs}$) spaces. In general, the aim of motion planning is to find a continuous path $\sigma \subset \mathcal{C}_{free}$ that lies in obstacle-free space and connects the given start (c_{init}) and goal (c_{goal}) configurations of the robot. In CMP, the challenge is not just to avoid collisions but also to find a path that satisfies the given constraint function $F(c : \mathcal{C} \mapsto \mathbb{R}^k)$, where k is the number constraints to be imposed. A configuration c is said to satisfy the given constraint function if $F(c) = 0$. Therefore, the constraint function defines a $(d-k)$-dimensional manifold within C-space, i.e., $\mathcal{M} = \{c \in \mathcal{C} | F(c) = 0\}$. Hence, the goal of CMP is to find a continuous feasible path $\sigma \subset \mathcal{M}_{free}$ connecting the given start and goal configurations, where $\mathcal{M}_{free} = \mathcal{M} \cap \mathcal{C}_{free}$.

IV. Neural Manipulation Planning

In this section, we formally present our novel constraint motion planning framework called CoMPNet that comprises the following components:

A. Task Encoder

The task encoder takes the task specification as the input for encoding. The task specification in our case is a text description, such as *"carefully move mug to the tray"*, *"open the cabinet"*, etc., that encapsulates the underlying constraints. For instance, the word *"carefully"* in the task specification *"carefully move mug to the tray"* implies both stability and collision-avoidance constraints. The output of the task encoder is a fixed size latent space encoding $Z_c \in$

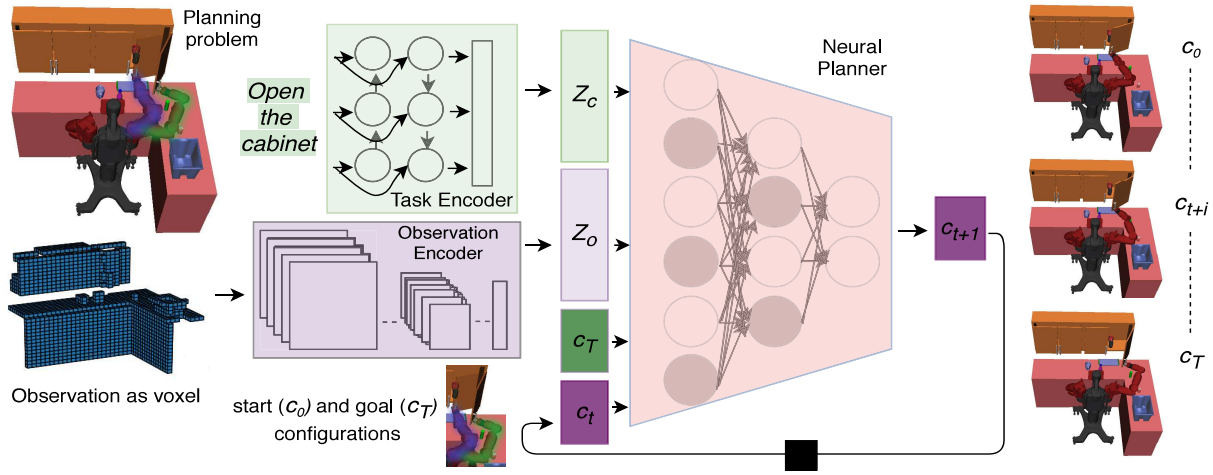


Fig. 2: CoMPNet generating configurations for the door opening task. Our neural planner takes task encoding Z_c, observation encoding Z_o, start configuration c_0 (purple) and goal c_T (green) configuration as input, and incrementally generates intermediate configurations c_{t+i} for the path planning.

Algorithm 1: CoMPNet($Z_c, Z_o, c_{init}, c_{goal}$)

$T_a \leftarrow c_{init}$; $c_t \leftarrow c_{init}$
$T_b \leftarrow c_{goal}$; $c_T \leftarrow c_{goal}$
for $i \leftarrow 0$ ***to*** n **do**
  $c^a_{t+1} \leftarrow \text{NeuralConfig}(Z_c, Z_o, c_t, c_T)$
  $c^a_{near} \leftarrow \text{NearestNode}(c^a_{t+1}, T_a)$
  $c^a_{new} \leftarrow \text{TransverseManifold}(c^a_{t+1}, c^a_{near}, T_a)$
  $c^b_{near} \leftarrow \text{NearestNode}(c^a_{new}, T_b)$
  $c^b_{new} \leftarrow \text{TransverseManifold}(c^a_{new}, c^b_{near}, T_b)$
  if $\text{Reached}(c^a_{new}, c^b_{new})$ **then**
    return $\text{ExtractPath}(T_a, c^a_{new}, c^b_{new}, T_b)$
  else
    $c_t \leftarrow c^a_{new}$; $c_T \leftarrow c^b_{new}$
    $\text{Swap}(T_a, T_b)$
    $\text{Swap}(c_t, c_T)$
return $\varnothing$

Algorithm 2: TransverseManifold(c_{target}, c_{near}, T)

$c_0 \leftarrow c_{near}$
$c_1 \leftarrow c_{near}$
$i = 0$
while $\|c_i - c_{target}\|_2 > \varepsilon$ **do**
  $c_{i+1} \leftarrow \text{Proj}(c_i + \Delta s(c_{target} - c_i))$
  if $\text{CollisionFree}(c_i, c_{i+1})$ **then**
    $T.\text{InsertNode}(c_{i+1})$
    $T.\text{InsertEdge}(c_i, c_{i+1})$
    $i = i + 1$
  else
    return c_i

$\mathbb{R}^{d_1}$ with dimensionality d_1. In our task encoder, we obtain the task specification representations using a pretrained recurrent-neural network-based model called InferSent [27]. We further process these representations by a feed-forward neural network, which is trained with other CoMPNet modules, to get an embedding of size d_1.

B. Observation Encoder

The observation encoder takes the environment perception information as an input to embed them into a latent space $Z_o \in \mathbb{R}^{d_2}$ of dimension d_2. In our settings, we obtain environment perception as a 3D point-cloud depth data, which is quantized as a voxel grid via voxelization. Ideally, 3D convolutional neural networks (CNNs) are used to process voxel-grids. However, in practice, 3D-CNNs are computationally expensive and impractical for large point-cloud data as their representations are inherently cubic and contain empty volume. Therefore, we convert the voxel grid of dimension $L \times W \times H \times C$, where C is a number of channels, to voxel image with voxel patches of dimension $L \times W \times (HC)$ and use 2D-CNNs to process them (for more details, refer to [28]).

C. Planning Network

It is a stochastic feed-forward neural network that drives its stochasticity from Dropout [29] during execution. The planning network (PNet) takes the observation encoding Z_o, task encoding Z_c, robot current c_t, and goal c_T configurations as input and learns to generate the next configuration c_{t+1} that will bring the robot closer to the goal configuration on the constraint manifold. Since PNet predicts one step at a time, the path is formed incrementally (Fig. 2).

D. Training Objective

We train the task encoder's feed-forward neural network, observation encoder network, and planning network end-to-end through supervised imitation learning. An oracle planner is used to generate demonstration trajectories for a given set of constrained planning problems. A demonstration trajectory comprises waypoints $\sigma = \{c_0^*, \cdots, c_T^*\}$, where c_0^* and c_T^* correspond to given start and goal configurations, respectively. Our training objective is to optimize the mean-square error between the predicted configurations and the actual configurations from the training data, i.e.,

$$\frac{1}{N_B} \sum_{i=0}^{N} \sum_{j=0}^{T_i - 1} ||c_{i,j+1} - c_{i,j+1}^*||^2, \tag{1}$$

where T_i is the length of each given path, $N \in \mathbb{N}$ is the number of paths in the training batch, and N_B is an averaging term. To train all modules together, we backpropagate the gradient of the loss function (Eqn. 1) from the planning network to the task and observation encoders.

E. Bidirectional Neural Planning Algorithm

Algorithm 1 outlines our bidirectional neural planning algorithm, whereas its essential components and overall execution are described as follows.

1) NeuralConfig: The NeuralConfig function uses the planning network to generate neural configuration on/near the manifold by taking the current c_t and desired goal c_T configurations together with the task Z_c and observation Z_o encodings as the input (Fig. 2), i.e.,

$$c_{t+1} \leftarrow \text{PNet}(Z_c, Z_o, c_t, c_T)$$

In NeuralConfig, the given configurations are normalized to $[-1, 1]$ before passing to the neural network and the generated neural configurations are unnormalized to robot joint-space for other path planning routines.

2) Transverse Manifold: The TransverseManifold function (Algorithm 2), also known as geodesic interpolation, extends the given T from the node c_{near} towards the target node c_{target} on the constraint manifold in small adaptive steps $\Delta s \in \mathbb{R}$. The procedure takes a linear Δs step towards the given target and project it to the constraint manifold using the projection operator (Algorithm 3). A projection operator (Proj) takes the configuration c and project it to the constraint manifold via gradient descent using inverses or pseudo-inverses of the Jacobian $J(c)$ of the differentiable constraint function F. We define F using TSRs [21] that returns displacement Δx in task space. The iterative projection finds, if one exists in a given loop limit, a constraint-satisfying configuration c such that $F(c) < \varepsilon$. The geodesic extension using Proj continues until it reaches c_{target} or if the collision happens.

Algorithm 3: Proj(c)

```
for i ← 0 to n do
    Δx ← F(c)
    if Δx < ε then
        return c
    else
        J ← GetJacobian(c)
        J⁺ ← GetPseudoInverse(J)
        Δc ← J⁺Δx
        c ← (c − Δc)
```

3) Algorithm summary: Let T_a and T_b contain the waypoints and their connecting edges from start to the goal configuration and from goal to the start configuration, respectively. The planning (Algorithm 1) begins by generating a neural configuration c_{t+1}^a towards the goal c_T followed by finding its nearest node c_{near}^a within T_a. The transverse manifold extends T_a from c_{near}^a towards c_{t+1}^a on the manifold and stops before collision leading to c_{new}^a. The algorithm then proceeds by finding the nearest node c_{near}^b to c_{new}^a within T_b. The transverse manifold function extends T_b from c_{near}^b towards c_{new}^a and terminates before collision occurs by producing c_{new}^b. If c_{new}^a and c_{new}^b reach each other, the algorithm extracts the end-to-end collision-free path connecting given start and goal configurations on the constraint manifold. The extracted path is further processed via smoothing to remove any redundant states before returning it as a feasible path solution. In case the c_{new}^a and c_{new}^b do not meet, the procedure continues by updating c_t with c_{new}^a and c_T with c_{new}^b followed by swapping the roles of (T_a, c_t) with (T_b, c_T) to solicit bidirectional path generation.

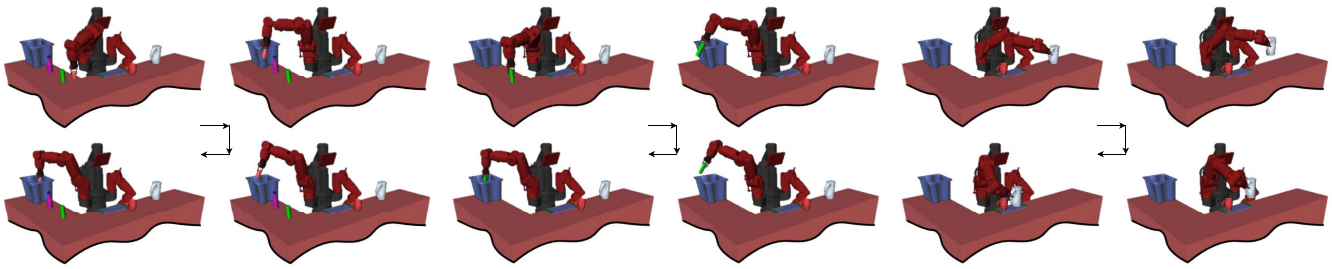
Note that our framework intelligently uses the trained planning network to generate configurations bidirectionally, i.e., from given start to goal configuration and from given goal to start configuration. It also computes nearest neighbors of the newly generated samples from each path to further solicit bidirectional geodesic extension on the constraint manifolds during each planning iteration. This is in contrast to MPNet algorithm that extends a path from c_t to c_{t+1} rather than from its nearest neighbors and rely on re-planning, making it less suitable for geodesic interpolation due to non-euclidean geometry of implicit manifolds.

V. Implementation details

We train CoMPNet neural models using PyTorch and port them to C++ via TorchScript for planning algorithm. The environments were set up in OpenRave, and for benchmark algorithms, we use their standard C++ implementations. The rest of the section provides details on the data generation, scene setup, and constraint representations. For more details, please refer to our supplementary material.

A. Data generation

In this section, we describe the data generation pipeline from setting scenarios to obtaining training data. Tasks are designed to involve grasping objects and controlling orientations on a series of sequences. Both grasped objects and forced orientation impose constraint manifolds in the robot configuration space that the planners must satisfy in addition to reaching its sequence of goals.



(a) Move soda can to the trash. (b) Move green can to the trash. (c) Carefully move kettle to the tray.

Fig. 3: Bartender setup: It requires the robot to clean the tables by placing the cans and bottle to the trash, and carefully moving the mug and kettle to the tray. Figs. (a-c) show some of the example subtasks.

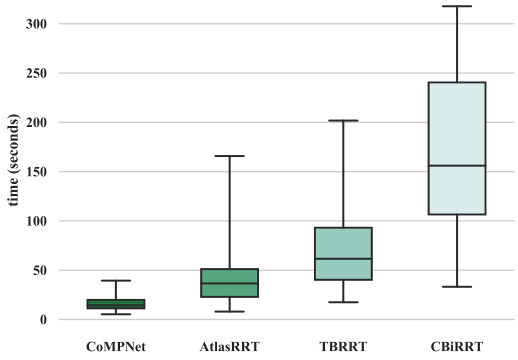


Fig. 4: The plot shows the interquartile range, minimum and maximum of total computation time of presented methods to solve all manipulation problems in the bartender task. Note that, CoMPNet's computation time is consistent and significantly faster than other benchmark methods.

1) Scene setup: We set up two practical scenarios, (i) bartender task (Fig. 3) and (ii) kitchen task (Fig. 5), each of which includes multiple sub-tasks.

The bartender task involves five manipulatable objects comprising the soda can (red), juice can (green), fuze bottle (purple), red mug, and kettle. The task is to place the soda can, juice can, and fuze bottle to the trash bin, and carefully transfer (without tilting) the red mug and kettle to the tray. In this task, we created 30 unique environments by random placement of trash bin and tray, reachable by the the robot's right arm. For each environment, we further create 60-110 unique scenarios by placing the five manipulatable objects randomly on the table at the reacjable locations.

The kitchen task involves seven manipulatable objects that included a soda can, juice can, fuze bottle, cabinet door, black mug, red mug, and pitcher. The task is to place the cans and bottle to the trash bin, open the cabinet to a given angle, carefully move (without tilting) the black and red mugs to the tray, and then move the pitcher into the cabinet. We create 60 unique environments by randomly placing a trash bin at the robot's right hand's reachable locations on the table and also by randomly setting the cabinet's door starting angle between 0 to $\pi/3$. For each environment, we further create about 30 unique scenarios through the random placement of the soda can, juice can, fuze bottle, tray, and pitcher at the robot's (right arm) reachable poses on the tables. Note that the starting pose of the black and red mugs are fixed in all cases, i.e., inside the cabinet, whereas their goal poses are on the randomly placed tray.

2) Observation data: From all generated scenarios, we get the point-cloud depth data using multiple Kinect sensors before solving each of the given sub-tasks in the order provided by the task planner. The point-cloud data from various sensors are stacked and converted to voxels via voxelization. The voxel dimensions for bartender and kitchen tasks/subtasks were $33 \times 33 \times 33$ and $32 \times 32 \times 32$, respectively.

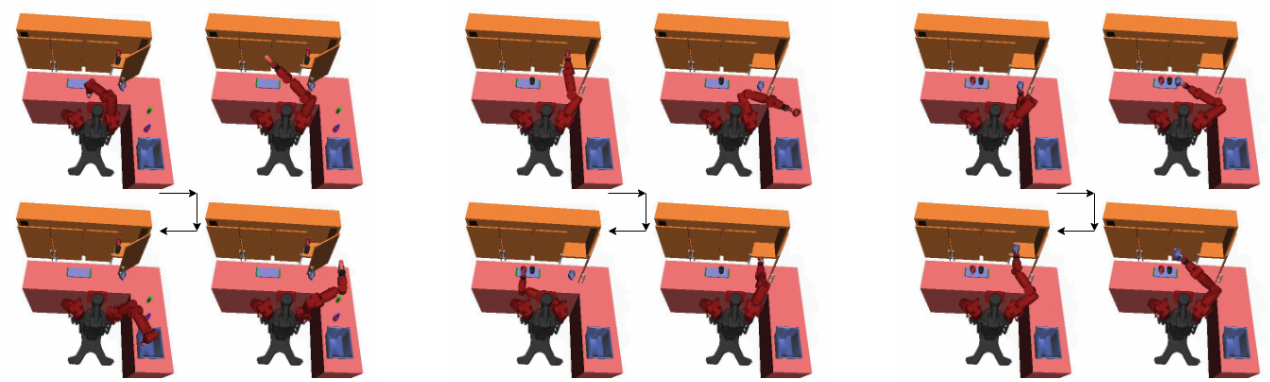
3) Planning trajectories (training & testing): We pull out more than 10% of the randomly generated scenarios for testing. For the remaining scenes, we get demonstration trajectories using CBiRRT [21] and use them for training the CoMPNet's neural models. In the real robot setup (Fig. 1), we replicate one of the bartender scenarios and execute CoMPNet's planned motion to demonstrate the transferability of our simulation experiments to the real-robot settings.

B. Constraint & Task Representations

We represent the constraint function F using Task Space Regions (TSRs) [21] and our task encoder takes the constraint information as text encoding using InferSent [27]. We train our CoMPNet for both reach and manipulation tasks, and their text description as follows:

i) **Reach tasks:** The text description is *"Reach the object name"*. The *object name* includes *"soda can"*, *"juice can"*, *"fuze bottle"*, *"red mug"*, *"black mug"*, *"kettle"*, *"pitcher"*, and *"cabinet"* depending on the given setting (bartender or kitchen) and their task plan.

ii) **Manipulation tasks:** In the bartender scenarios, the text descriptions are *"move the object name A to the trash"* and *"carefully move the object name B to the tray"*. In the kitchen scenarios, the text description are *"move the object name A to the trash"*, *"open the cabinet"*, *"carefully move the object name C to the tray"*, and *"carefully move the pitcher to the cabinet"*. The *object name A* includes *"soda can"*, *"juice can"*, and *"purple bottle"*. The *object name B* gets *"red mug"*, and *"kettle"*. The *object name C* corresponds to *"red mug"*, and *"black mug"*.



(a) Move soda can to the trash. (b) Carefully move red mug to the tray. (c) Carefully move pitcher to the cabinet.

Fig. 5: Kitchen setup: Figs. (a-c) and Fig. 2 show some instances of the CoMPNet's path execution in these scenarios.

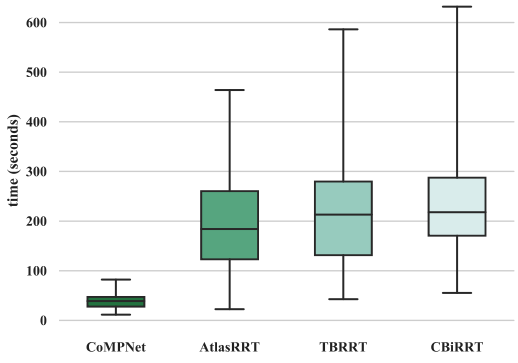


Fig. 6: The plot exhibits the interquartile range, minimum and maximum of total computation time of CoMPNet, Atlas-RRT, TB-RRT, and CBiRRT to solve all manipulation tasks in the kitchen setup. Note that, CoMPNet's has significantly narrower and lower ranges of computational time than other benchmark algorithms.

VI. Results

In this section, we present the computation time comparison of CoMPNet and state-of-the-art CMP methods named CBiRRT [21], Atlas-RRT [22], and TB-RRT [23] on several bartender and kitchen tasks. All experiments were performed on a system with 32GB RAM, GeForce GTX 1080 GPU, and 3.40GHz×8 Intel Core i7 processor.

A. Comparative Studies

In the unconstrained planning problems, i.e., reaching-to-the-target-object, we validate that CoMPNet's mean computation time is about 1-2 seconds, which is similar to results with MPNet [11], showing that performance improvements over gold-standard unconstrained SMPs are retained.

In the constrained manipulation planning problems of the bartender (Fig. 1 & Fig. 3) and kitchen (Fig. 5) environments, the mean success rates of all the presented methods were around 90%. Fig. 4 and Fig. 6 compare algorithms in both environments using the box plots of mean accumulated computation times in solving all the manipulation tasks. Furthermore, Table I also provides the mean computation time with standard deviations of manipulating each object, grouped by their constraint types, in each setting.

Note that all test environments were randomly generated and were not seen by the CoMPNet during training. These environments are challenging, representing practical scenarios, and often requiring a planner to find convoluted long-horizon paths through narrow passages. For instance, Fig. 5 (b) shows a CoMPNet path solution for manipulating a red mug in the kitchen setup. It is a non-trivial, long-horizon plan that transverses narrow passages formed by the door and other objects on the table.

Despite challenging planning problems, it can be seen that CoMPNet compared to other methods exhibits i) higher/similar success rates, ii) lower inter-quartile computational time ranges, iii) lower minimum and maximum computation times, and iv) lower mean computation times with a narrower standard-deviations. Although CoMPNet uses constraint adherence methods like classical CMP algorithms for traversing manifolds, its lower computation times indicate that the generated samples are mostly on the constraint manifold and does not rely on the constraint adherence operator significantly. We also observed that continuation-based operators are highly sensitive to their parameters, and lazy evaluation heuristic of TB-RRT often leads to invalid states causing poor performance than other methods.

B. Ablative Studies

We present an ablation study to highlight the significance of the following components added to MPNet that led to CoMPNet for scalable CMP: 1) A projection operator for constraint adherence and steering on the manifold. 2) A planning algorithm with bidirectional constrained extensions to newly generated sample c_{t+1} from its nearest neighbors $\{c^a_{near}, c^b_{near}\}$ rather than its previous configuration c_t. 3) A task specification for scalability and multimodality which could be one-hot or text-based encodings. Hence, our first model is MPNet without re-planning phase and with a projection operator for steering. Second is the proposed CoMPNet framework without task encodings. Third and fourth are the proposed CoMPNet models with one-hot and text-based task specifications, respectively.

Setup	Objects	Algorithms			
		CoMPNet	CBiRRT	Atlas-RRT	TB-RRT
Bartender	J/F/S	$\mathbf{4.92 \pm 2.42}$	54.81 ± 25.82	14.24 ± 9.331	23.48 ± 14.84
	R/K	$\mathbf{1.17 \pm 0.93}$	1.940 ± 1.380	1.276 ± 0.361	1.574 ± 5.641
Kitchen	J/F/S	$\mathbf{4.28 \pm 2.86}$	32.65 ± 22.40	24.87 ± 19.82	27.55 ± 21.47
	C	$\mathbf{0.03 \pm 0.02}$	00.05 ± 00.04	00.04 ± 00.03	00.05 ± 00.04
	R/B/P	$\mathbf{9.16 \pm 3.04}$	49.79 ± 22.95	41.28 ± 24.02	46.61 ± 26.07

TABLE I: The mean computation times with standard deviations of solving manipulation problem of each object, grouped by their constraint types, in both bartender and kitchen environments. The objects are denoted by their first letter. It can be seen that CoMPNet computation times are lower and more consistent across different problems than other methods.

Table II presents the total mean computation times with standard deviations and success rates of all models mentioned above in the bartender and kitchen environments. Although MPNet performs well in unconstrained planning problems, it can be seen that MPNet with only a projection operator and no re-plannings performs poorly in terms of success rates than other models in CMP. Similarly, the task encoding also leads to improved success rates validating that it is a crucial component of CoMPNet. Furthermore, task specification such as one-hot or text-based representation gives similar performances. However, using a sequential embedding such as text-based constraint specification is vital as they allow scalability to an arbitrary number of constraint types. In contrast, other methods such as one-hot representations would scale poorly and become very limited in practice with a growing set of multi-task and multimodal constraints.

VII. Discussion

In this section, we present a brief discussion on CoMPNet's ability to solve multimodal constraint motion planning problems and exhibit probabilistic completeness on manifold coverage if merged with uniform C-space sampling methods.

A. Multimodal Constraints

To the best of authors' knowledge, CoMPNet is the first planning algorithm capable of handling multimodal constraints and exploring various constraint manifolds simultaneously. In the presented kitchen and bartender setups, the problems require: i) reaching to the target end-effector pose to grab the given object; ii) manipulating the grabbed object under various constraints such as stability and collision-avoidance; iii) opening the cabinet door, which also imposes constraints inculcated by the allowable rotation of the door's hinge. In all of these problems, CoMPNet's high success rate and low computation time validate that its planning network, conditioned on the observation and text-based task encodings, can implicitly transition between different constraint manifolds and can produce the constraint-adhering configurations efficiently.

B. Stochasticity & Manifold Coverage

In this section, we highlight that CoMPNet, coupled with an exploration-based C-space sampling strategy, covers the constraint manifold leading to probabilistic completeness guarantees. The probabilistic completeness guarantees are that the planner will output a path solution, if one exists, with a probability of one, if it is allowed to run for a large number of iterations approaching infinity.

In CoMPNet, we apply Dropout [29] with probability of 0.5 to almost every layer of the planning network during offline training and online planning. The Dropout randomly skips the output of some of the neurons from its preceding neural network's layer according to the given probability $p \in [0, 1]$. In [30], Yarin and Zubin use Dropout for uncertainty modeling in the neural networks. In our method, we use Dropout-based stochasticity in the planning network to generate configuration samples on/near the constraint manifold. The generated configuration samples that are slightly off are projected to the constraint manifold using the gradient-descent-based projection operator. In [21][24], it is proved that the projection operator (Algorithm 3) combined with uniform C-space sampling fully explores the underlying constraint manifold with the running time approaching infinity. CoMPNet can also be combined with uniform C-space sampling techniques leading to the exploration-exploitation approach. The exploitation phase will be to leverage CoMPNet's planning network for a fixed number of iterations to generates samples on/near the constraint manifold, potentially leading to a path solution. The exploration phase

Tasks	Algorithms			
	MPNet (with Proj)	CoMPNet (w/o task encoder)	CoMPNet (with one-hot)	CoMPNet
Bartender	18.80 ± 8.90 (75.1%)	18.84 ± 8.96 (82.1%)	17.00 ± 8.31 (87.8%)	16.52 ± 7.67 (88.3%)
Kitchen	44.01 ± 9.53 (68.6%)	41.21 ± 12.13 (86.2%)	41.93 ± 13.97 (89.7%)	38.6 ± 13.23 (91.1%)

TABLE II: Ablation study: The total mean computation times with standard deviations and mean success rates are presented for CoMPNet and it's ablated models in solving all manipulation tasks in the bartender and kitchen environments.

will be to do uniform sampling after exploitation. Hence, the exploration-exploitation based sampling approach paired with a projection operator will cover the constraint manifold with probabilistic completeness, and the proof can be derived in the same way as in [21][24].

VIII. Conclusions & Future Works

We proposed Constrained Motion Planning Networks (CoMPNet), a neural network-based bidirectional planning algorithm that is shown to solve complex planning problems under multimodal kinematic constraints in seconds and with significantly less variability than state-of-art methods that take minutes and have high variability. CoMPNet also encapsulates MPNet [10] as it not only solves constrained manipulation problems but also unconstrained planning problems, i.e., reach to the given robot end-effector poses to grasp the objects. Furthermore, we also show that, similar to MPNet [10], CoMPNet generalizes to unseen tasks that were not in the training examples with a high success rate and probabilistic completeness.

In our future studies, we plan to extend CoMPNet to integrated task and motion planning by leveraging its fast computational speed for almost real-time motion reasoning. In addition, we also aim to incorporate dynamical constraints to allow a computationally efficient kinodynamic path planning for practical, real-world problems from the high-dimensional observation data.

Acknowledgments

We thank Dmitry Berenson for the insightful discussions on the proposed work and Frank Park for sharing their TBRRT implementations.

References

[1] C. Ott, O. Eiberger, W. Friedl, B. Bauml, U. Hillenbrand, C. Borst, A. Albu-Schaffer, B. Brunner, H. Hirschmuller, S. Kielhofer, *et al.*, "A humanoid two-arm system for dexterous manipulation," in *2006 6th IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2006, pp. 276–283.

[2] G. H. Ballantyne and F. Moll, "The da vinci telerobotic surgical system: the virtual operative field and telepresence surgery," *Surgical Clinics*, vol. 83, no. 6, pp. 1293–1304, 2003.

[3] L.-W. Tsai, *Robot analysis: the mechanics of serial and parallel manipulators*. John Wiley & Sons, 1999.

[4] Z. Kingston, M. Moll, and L. E. Kavraki, "Sampling-based methods for motion planning with constraints," *Annual review of control, robotics, and autonomous systems*, vol. 1, pp. 159–185, 2018.

[5] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[6] B. Ichter, J. Harrison, and M. Pavone, "Learning sampling distributions for robot motion planning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 7087–7094.

[7] A. H. Qureshi and M. C. Yip, "Deeply informed neural sampling for robot motion planning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 6582–6588.

[8] M. J. Bency, A. H. Qureshi, and M. C. Yip, "Neural path planning: Fixed time, near-optimal path generation via oracle imitation," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 3965–3972.

[9] B. Ichter and M. Pavone, "Robot motion planning in learned latent spaces," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2407–2414, 2019.

[10] A. H. Qureshi, A. Simeonov, M. J. Bency, and M. C. Yip, "Motion planning networks," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 2118–2124.

[11] A. H. Qureshi, Y. Miao, A. Simeonov, and M. C. Yip, "Motion planning networks: Bridging the gap between learning-based and classical motion planners," *IEEE Transactions on Robotics*, 2020.

[12] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Batch informed trees (bit*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs," in *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2015, pp. 3067–3074.

[13] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 489–494.

[14] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

[15] R. Bonalli, A. Cauligi, A. Bylard, T. Lew, and M. Pavone, "Trajectory optimization on manifolds: A theoretically-guaranteed embedded sequential convex programming approach," in *Proceedings of Robotics: Science and Systems*, FreiburgimBreisgau, Germany, June 2019.

[16] L. E. Kavraki and J.-C. Latombe, "Probabilistic roadmaps for robot path planning," *Pratical motion planning in robotics: current approaches and future challenges*, pp. 33–53, 1998.

[17] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[18] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[19] D. Xie and N. M. Amato, "A kinematics-based probabilistic roadmap method for high dof closed chain systems," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, vol. 1. IEEE, 2004, pp. 473–478.

[20] M. Stilman, "Global manipulation planning in robot joint space with task constraints," *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 576–584, 2010.

[21] D. Berenson, S. Srinivasa, and J. Kuffner, "Task space regions: A framework for pose-constrained manipulation planning," *The International Journal of Robotics Research*, vol. 30, no. 12, pp. 1435–1460, 2011.

[22] L. Jaillet and J. M. Porta, "Path planning with loop closure constraints using an atlas-based rrt," in *Robotics Research*. Springer, 2017, pp. 345–362.

[23] B. Kim, T. T. Um, C. Suh, and F. C. Park, "Tangent bundle rrt: A randomized algorithm for constrained motion planning," *Robotica*, vol. 34, no. 1, pp. 202–225, 2016.

[24] Z. Kingston, M. Moll, and L. E. Kavraki, "Exploring implicit spaces for constrained sampling-based planning," *The International Journal of Robotics Research*, vol. 38, no. 10-11, pp. 1151–1178, 2019.

[25] A. H. Qureshi and Y. Ayaz, "Intelligent bidirectional rapidly-exploring random trees for optimal motion planning in complex cluttered environments," *Robotics and Autonomous Systems*, vol. 68, pp. 1–11, 2015.

[26] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed rrt*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 2997–3004.

[27] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes, "Supervised learning of universal sentence representations from natural language inference data," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, September 2017, pp. 670–680.

[28] C. Zhang, W. Luo, and R. Urtasun, "Efficient convolutions for real-time semantic segmentation of 3d point clouds," in *2018 International Conference on 3D Vision (3DV)*. IEEE, 2018, pp. 399–408.

[29] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

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