2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
September 27 - October 1, 2021. Prague, Czech Republic

2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-6654-1714-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/IROS51168.2021.9636682

A Conformal Mapping-based Framework for Robot-to-Robot and Sim-to-Real Transfer Learning

Shijie Gao and Nicola Bezzo

***Abstract*— This paper presents a novel method for transferring motion planning and control policies between a teacher and a learner robot. With this work, we propose to reduce the sim-to-real gap, transfer knowledge designed for a specific system into a different robot, and compensate for system aging and failures. To solve this problem we introduce a Schwarz–Christoffel mapping-based method to geometrically stretch and fit the control inputs from the teacher into the learner command space. We also propose a method based on primitive motion generation to create motion plans and control inputs compatible with the learner's capabilities. Our approach is validated with simulations and experiments with different robotic systems navigating occluding environments.**

I. Introduction

Robotic applications are typically built considering specific systems in mind. For example, popular motion planning methods (e.g., artificial potential field [1], A* [2], probabilistic techniques [3]) and control methods (e.g., MPC, PID [4]) require fine tuning and knowledge about system model dynamics in order to be fully leveraged and obtain a desired performance on a selected platform. We also note that most technologies are developed through simulations which offer a practical and inexpensive mean to create and test the limits and performance of designed algorithms. Researchers usually spend considerable time and resources to create techniques for specific robotic systems and to adapt them on new systems, as well as to compensate for the simulation-reality gap during deployments on actual vehicles. Finally, even when a new technique is developed and deployed on a specific robot, it can still need to be adjusted or adapted over time due to mechanical aging, disturbances, and even failures that deprecate and modify the system's original model. In this paper we seek a general framework to transfer and adapt system's performance. As mentioned above the goal of the proposed work is to:

- Reduce the sim-to-real gap allowing a developer to quickly transfer motion planning and control methods onto a real platform.
- Transfer knowledge designed for a specific robot onto a different robot.
- Compensate for system deterioration/failures by learning quickly the limits and the proper input mapping to continue an operation.

All of the aforementioned problems can be simplified and cast as a *teacher* transferring knowledge to a *learner*.

Specifically, to address these problems, in this work we propose a novel method that leverages a variant of Schwarz–Christoffel mapping (SCM) [5] – a conformal transformation of a simple poly area onto the interior of a rectangle – to transfer a teacher vehicle's control input sequence to a learner vehicle, as depicted in Fig. 1. Our proposed method allows the teacher to understand the learner limitations, so that the transferred control input is compatible with the learner capabilities. Finally, once these limitations are extracted, we propose a mechanism to adapt also the teacher motion planning scheme to create paths compatible with the learner constraints. To deal with this problem, our scheme leverages an optimized finite horizon primitive motion generation.

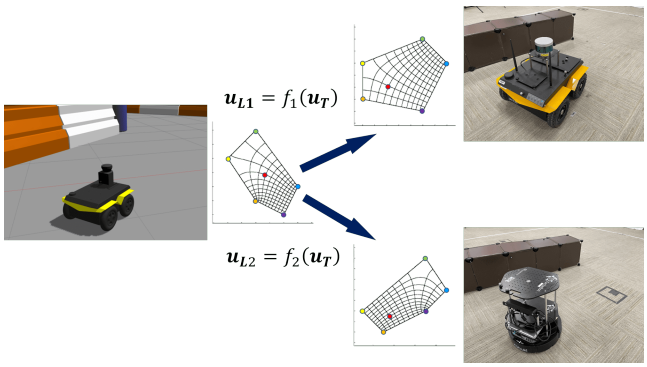

Fig. 1. Pictorial representation of the proposed work in which motion planning and control policies are transferred from a teacher simulated vehicle to two vehicles to create the same behavior designed in simulation.

The main contributions of this work are twofold: 1) a light-weight transfer framework that leverages SCM theory to directly transfers the control input from teacher to learner so that the learner can leverage the teacher's control policy while its own dynamics remain unknown; and 2) a method for adapting the source system's control and path planning policy to the learner. The method constrains the output of the source system's controller and of the path planner so that the transferred motion plan and control input is guaranteed to be compatible with the target system's dynamics.

The rest of the paper is organized as follows: in Section II we summarize the state-of-the-art approaches for solving sim-to-real problems in the current literature. We formally define the problem in Section III while the details of our SCM-based transfer learning framework are presented in Section IV. The proposed framework is validated with extensive simulations in Section V and experiments on real robots in Section VI. At last, we draw conclusions in Section VII.

II. Related Work

Transfer learning has been one of the most popular topics in robotics, especially since machine learning techniques have become widely exploited. The idea behind transfer

Shijie Gao, and Nicola Bezzo are with the Charles L. Brown Department of Electrical and Computer Engineering, and Link Lab, University of Virginia, Charlottesville, VA 22904, USA. Email:{sg9dn, nb6be}@virginia.edu

learning is to migrate the knowledge between similar problems to boost the training process [6], take advantage of existing knowledge [7], and reduce the risk of training [8], [9]. Although machine learning approaches have been massively explored, we cannot ignore that they typically require a large amount of data and a lot of effort in training the model.

The problem of transferring from the simulation to the real world, also known as sim-to-real problem, has gained rising attention recently. The gap between the simulation and the real system exists mainly because either the model is not accurate or the environment factors do not appear in the simulation. The modeling gap can be closed by retraining the pre-trained model in real world [10]. Dynamics randomization is another popular solution which aims to cover reality with augmented simulation scenarios [10] [11]. Other approaches include reducing the costly errors by predicting the blind spots in real environments [12] and inflating safety critical regions to reduce the chance of collision [13]. Learning from demonstration is another sub-field of transfer learning in which reinforcement learning is usually getting involved. These types of works typically learn the policy from teacher's examples by approximating the state-action mapping [14], or by learning the system model [15]. Most of these problems turn into an optimization problem on tuning parameters. Although fewer training demos are desired, it can still take a large amount of data to address the problem. Thus, both the acquisition of data and the tuning process can be challenging when dealing with these types of problems.

To the best of our knowledge, the SCM method proposed in this paper is rarely used in the robotics field. In [16], the SCM is leveraged to map the planar motion to the continuous linear motion to solve a coverage control problem for wire-traversing robots. Comparing to the existing works, this paper proposes a light-weight transfer learning framework which does not rely on massive data collection. It is also the first work that exploits the conformal mapping method to directly transferring control inputs between two systems.

III. Problem Formulation

The problem behind this work can be cast as a teacher transferring knowledge to a learner vehicle. We assume that the teacher has more capabilities than the learner, meaning that it can achieve all the learner's maneuver but not vice versa. This assumption is suitable for our problem since we are primarily interested in transferring knowledge into a vehicle with degraded capabilities, and as it is easier to create a virtual simulated vehicle with more capabilities than a real vehicle in sim-to-real problems. The learner's dynamics are assumed a black-box model with only access to the inputs and output. The goal is to transition the behavior and control knowledge of the teacher into the learner including adapting the teacher motion planning framework to consider the limitations of the learner. Formally we can define two problems:

Problem 1. ***Teacher-Learner Control Transfer:*** *Given a teacher robot with dynamics $\boldsymbol{x_T}(t+1)=f_T(\boldsymbol{x_T}(t),\boldsymbol{u_T}(t))$ and control law $\boldsymbol{u_T}=g(\boldsymbol{x})$, where $\boldsymbol{x}$ is the state vector and $\boldsymbol{u}$ is the control input, find a policy to map $\boldsymbol{u_T}$ to a learner input $\boldsymbol{u_L}$ such that $\boldsymbol{x_L}(t+1)=f_L(\boldsymbol{x_L}(t),\boldsymbol{u_L}(t))=\boldsymbol{x_T}(t+1)$, with f_L unknown.*

Problem 2. ***Teacher-Learner Motion Planning Adaptation:*** *Consider a task to navigate from an initial location to a final goal G. Assume that the learner's input space $\boldsymbol{u_L} \in [\boldsymbol{u_L}_{min}, \boldsymbol{u_L}_{max}] \subset [\boldsymbol{u_T}_{min}, \boldsymbol{u_T}_{max}]$. Design a motion planning policy $\boldsymbol{\pi_T^L}$ for the teacher that considers the limitations of the learner and such that the computed desired trajectory τ can be tracked by the learner, i.e., such that $|\boldsymbol{x}_L - \boldsymbol{x}_\tau| \leq \epsilon$ where ϵ is a maximum allowable deviation threshold.*

IV. Methodology

Problem 1 is solved by leveraging SCM to comformally map between the teacher's and the learner's command domains. Problem 2 is addressed by constraining the teacher's control and planning policy in accordance with the learner's limitation. The block diagram in Fig. 2 shows the architecture of the whole process. The remainder of this section describes the details of the components of the proposed approach.

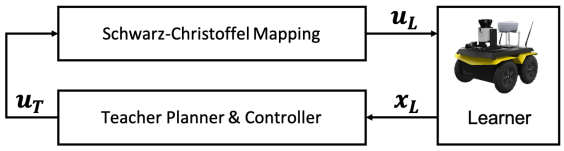

Fig. 2. The architecture of the proposed transfer learning process.

A. SCM-based Command Transferring

As we treat the dynamics of the learner as a black box, it is impossible to build a one-to-one command mapping without running inputs on the learner. In our work, we propose to use a limited number of teacher commands to characterize the learner's dynamics and then use SCM to find the mapping function between the region on the teacher's command domain and the corresponding region on the learner's side.

We use command pairs to characterize the learner's dynamics. The command pair $\boldsymbol{u_p}=\langle \boldsymbol{u_T}, \boldsymbol{u_L} \rangle$ is a pair of commands which makes the two vehicles produce the same motion (i.e., reach the same pose, speed). Since the dynamics of the teacher are known, by observing the states of the learner before and after executing $\boldsymbol{u_L}$, the equivalent teacher's command $\boldsymbol{u_T}$ can be retrieved. A group of these command pairs can capture the dynamics of the learner on the teacher command domain. At each control step, the learner uses the teacher's control policy to generate a control input which is the teacher's desired command as if the learner was the teacher. Given a desired teacher's command and several command pairs around it, the region whose vertices are from the command pairs and contains the desired command can be chosen on the teacher side. The corresponding region on the learner command domain is decided automatically by the learner's commands that come from the same command pairs as the teacher's vertices. An example is shown in Fig. 3.

Once the regions of interest are determined on both teacher's and learner's command space, the transfer problem becomes a problem of finding the mapping function that transfers from an irregular polygon on the teacher's domain to the other polygon on the learner's domain. To solve this

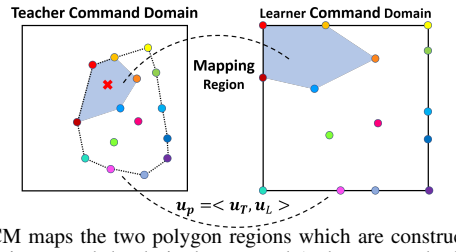

Fig. 3. SCM maps the two polygon regions which are constructed by the command pairs around the desired command (red cross on the left).

problem, first we use SCM to map the two polygons on each side of the command domain onto two rectangles with unique aspect ratios, which are decided by the shape of the mapping area. The reason why we map the two regions onto two different rectangles will appear as we walk through the mapping procedure. Then, we use a unit square to bridge the two rectangles so a teacher command can be mapped to the learner's domain. Fig. 4 shows the mapping flow.

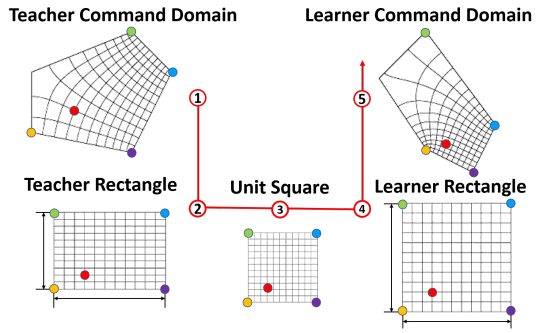

Fig. 4. The mapping flow of transferring the desired teacher command to the learner. A unit square is used as an intermediate plane to bridge between rectangle mapping of the two polygons.

Based on the user's preference, multiple command pairs can be selected to build the mapping areas Γ. For any of these irregular polygons, we can specify four of the vertices in the counterclockwise order to map to the rectangle's corners. These four vertices make Γ a generalized quadrilateral. Fig. 5 shows an example of this process, where we put the polygon from the teacher command domain onto the extended complex plane.

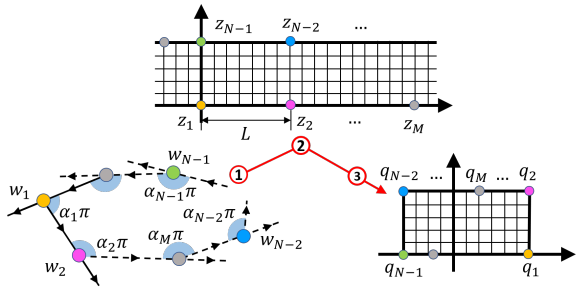

Fig. 5. The flow of conformal mapping that maps the polygon to the rectangle while using the bi-infinite strip as the intermediate plane.

As shown in Fig. 5, the vertices of the polygon $w_1, ..., w_N$, $(N \geq 4)$ are ordered in counterclockwise and the interior angles $\alpha_1\pi, ..., \alpha_n\pi$ at each of the vertex w_N is defined as the angle that sweeps from the outgoing edge to the incoming edge. The conformal mapping from the polygon Γ to the rectangle $\mathbb{Q}$ needs to borrow a bi-infinite strip $\mathbb{S}$ as an intermediate plane. The SCM function that maps the points on the boundary of the strip $\mathbb{S}$ to the vertices of the polygon is given by:

$$w = f_{\mathbb{S}}^{\Gamma}(z) = A\int_0^z \prod_{j=0}^{N} f_j(z)dz + C \tag{1}$$

where A and C are complex constants that rotate, translate and scale the polygon and are determined by its shape and location. Each factor f_j sends a point on the boundary of the strip to a corner of the polygon while preserving its interior angles. The factor f_j is a piecewise function which is defined by:

$$f_j(z)=\begin{cases} e^{\frac{1}{2}(\theta_+-\theta_-)z} & j=0, \\ \{-i\cdot\sinh[\frac{\pi}{2}(z-z_j)]\}^{\alpha_j} & 1\leq j\leq M, \\ \{-i\cdot\sinh[-\frac{\pi}{2}(z-z_j)]\}^{\alpha_j} & M+1\leq j\leq N, \end{cases} \tag{2}$$

where M is the number points on the bottom side of the strip. θ_+ and θ_- denote the desired divergence angles at $+\infty$ and $-\infty$, which are $\theta_+ = \theta_- = \pi$ in our case.

By leveraging the Jacobi elliptic of the first kind [17], the SCM mapping $f_{\mathbb{Q}}^{\mathbb{S}}$ from the rectangle $\mathbb{Q}$ to the bi-infinite strip $\mathbb{S}$ can be defined by:

$$z = f_{\mathbb{Q}}^{\mathbb{S}}(q) = \frac{1}{\pi}\cdot\ln(\sin(q|m)) \tag{3}$$

where q is the point on regular rectangle and m is the modulus of the Jacobi elliptic that is decided by q. The details of this conformal mapping can be found in [5]. With Eqs. (1) and (3), a mapping function from the generalized quadrilateral can be obtained. In order to explicitly solve (1), there are three parameters z_k that must be specified. For ease of computation, for example, we can fix $z_1 = 0$, $z_2 = L$, $z_{N-1} = i$, and $z_{N-2} = L+i$. The parameter L here is linked to the conformal modulus m.

While the angles of the polygon are computed with (1) and (2), we need to find where the pre-vertices lie on the boundary of the strip to keep the length for each edge of polygon. This problem is known as the parameter problem in SCM [5]. Since we already fix $z_1 = 0$, in (1) the translation parameter is set to be $C = 0$. Hence, solving (1) is equal to solving:

$$w_k = A\int^{z_k} \prod_{j=0}^{N} f_j(z)dz, \quad k = 1, 2, 3, \ldots, N \tag{4}$$

In (4), the scalar A can be eliminated by the ratio of the adjacent sides length of the polygon:

$$\frac{w_{k+1}-w_k}{w_2-w_1} = \frac{\int_{z_k}^{z_{k+1}} \prod_{j=0}^{N} f_j(z)dz}{\int_{z_1}^{z_2} \prod_{j=0}^{N} f_j(z)dz}, \; k=2, 3, \ldots, N-2 \tag{5}$$

Let

$$I_k = \left|\int_{z_k}^{z_{k+1}} \prod_{j=0}^{N} f_j(z)dz\right|, \quad k = 1, 2, \ldots, N-2 \tag{6}$$

Then (5) can be rewritten as:

$$I_k = I_1\cdot\frac{w_{k+1}-w_k}{w_2-w_1}, \quad k = 2, 3, \ldots, N-1 \tag{7}$$

To this end, (7) leaves us $N-3$ conditions and the unknown parameters of (4) are $z_k\,(k = 1, 2, \ldots, N-3)$ which is exactly the number of the side length conditions given by (7) . We can get the complex constant A by:

$$A = \frac{w_2 - w_1}{\int_{z_1}^{z_2} \prod_{j=0}^{N} f_j(z) dz}. \tag{8}$$

As we get the conformal mapping function $f_{\mathbb{S}}^{\Gamma}$ from the strip to the generalized quadrilateral, we can compute $L = z_2 - z_1 = f_{\mathbb{S}}^{\Gamma\,-1}(w_2) - 0$. Considering (3) which maps the rectangle to the strip, the SCM function that maps the interior and the boundary of the generalized quadrilateral to the rectangle with an unique aspect ratio can be obtained by:

$$q = f_{SCM}(w) = f_{\mathbb{Q}}^{\mathbb{S}\,-1}(f_{\mathbb{S}}^{\Gamma\,-1}(w)). \tag{9}$$

As the shape of the rectangle $\mathbb{Q}$ depends on the parameter L, the aspect ratio of the rectangle is determined after L is computed. This explains why we map the two polygons from teacher and the learner command domains to two different rectangles. Since the dynamics of the teacher and learner are different, the shape of the polygons from the teacher and the learner cannot be identical, and neither are the mapped rectangles. A unit square is borrowed to bridge between the two mapped rectangles resulting in a complete mapping process from teacher to the learner, such that any teacher command that falls in the teacher's mapping area is connected to an image on the learner side.

There are a few points that are worth mentioning: 1) Although we use rectangle SCM and the number of the vertices for a polygon is at least 4 ($N \geq 4$), this mapping-based transferring framework still works for the triangle areas ($N = 3$) by leveraging a disk SCM function or an upper half-plane SCM function. 2) If the distance between the desired command and the existed closest command pair is smaller than a threshold ψ, it means that the desired motion is very similar to the motion produced by the closest pair. In this case, it is reasonable to skip the mapping procedure and directly use the learner's command from the closest pair. 3) If the command pairs that are used for constructing the mapping polygon are too far from the desired command, some local geometric features between the two domains may not be well captured during mapping. Thus, the number as well as the distribution of the command pairs can affect the mapping performance. More command pairs that cover the learner's command domain well are preferred.

B. Primitive Path Planning

As the vehicle learns the mapping function, it is also important to know the limitations of the learner so that the teacher's policy can generate the command to plan the motions that are compatible with the learner. This means that we want to find where the command boundary of the learner lies within the teacher command domain. This can be achieved by getting the command pair $\boldsymbol{u_p}=\langle \boldsymbol{u_T}(t), \boldsymbol{u_L}(t)\rangle$ when $\boldsymbol{u_L}(t)=\boldsymbol{u_{L}}_{max}$. As shown in Fig. 3, the teacher's control inputs from these command pairs can build a multi-dimensional convex hull that separates the interior of the convex hull from the rest of the command area. From the teacher's perspective, the boundary of the convex hull indicates the limitations of the learner. Any of teacher's commands from the interior of the convex hull can be matched with the learner's command, enabling the two vehicles to produce the similar motion with their own commands. However, as it is pointed out at the end of Section IV-A, to obtain better mapping performance, it is recommended to consider additional command pairs inside of the polygon.

We use a trajectory tracking case study to validate our approach. The teacher uses a search-based path planning method to compose a sequence of motion primitives that allows it to drive along the desired path P within a certain bounds. The teacher's input sequence associated to these primitives will be the desired commands for mapping.

A motion primitive results from feeding a known sequence of control inputs to the vehicle. To build one primitive $p=[\boldsymbol{x_T}_1, \boldsymbol{x_T}_2, \ldots, \boldsymbol{x_T}_t]$, we feed the teacher a sequence of the same control input for a certain amount of time and record its state sequence. Following the same procedure, a library of primitives can be built with different teacher's command. In Fig. 6, we show 5 different motion primitives that resulted from 5 different teacher's commands. The one-to-one primitives and the corresponding commands are color coded. The command pairs are shown as the gray points and the white region indicates the capability of the learner.

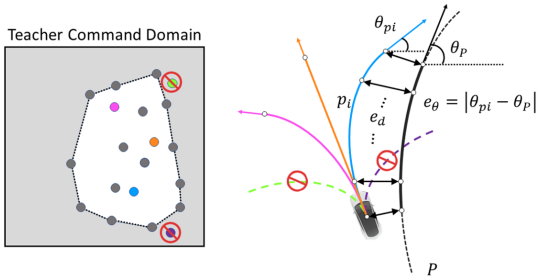

Fig. 6. The teacher commands and the corresponding motion primitives are shown on the left while a path planning scenario is shown on the right.

We want to point out that: 1) To better adapt to the capability of the learner, only the command which falls inside of the convex hull should be considered. 2) The learner can leverage the teacher's motion planner as soon as the convex hull is built. 3) The convex hull does not need to capture the entire command domain of the learner, it just provides a boundary that make sure the learner is operating within the known capability.

As the path planner searches primitives from the library to use, it evaluates the difference between each of the primitive and the corresponding segment on the desired path. As shown in (10) and in Fig. 6, the difference is measured by considering both the dynamic time warping (DTW) distance e_d and the heading difference e_θ at the end of the primitive:

$$\begin{aligned} \delta_i &= k_d \cdot e_d + k_\theta \cdot e_\theta \\ &= k_d \cdot DTW(P, p_i) + k_\theta \cdot |(\theta_P - \theta_{p_i})|, \\ p_i^* &= \min_{p_1,\ldots,p_i} \delta_i. \end{aligned} \tag{10}$$

The two types of differences are weighted by two user-defined gains ($k_d \geq 0$, $k_\theta \geq 0$). A large k_d will force the vehicle to remain close to the trajectory while a large k_t will give the primitives that are parallel to the trajectory a better chance to be chosen. Using this metrics, the planner searches through all the primitives in the library and selects the one with the least difference as the optimal local path plan p_i^*. The teacher's control input $\boldsymbol{u_T^*}$, which is associated to p_i^*, is the command that will be mapped to the learner.

After a command sequence is executed, the learner will evaluate the situation and use the planner to generate a new local path and corresponding command sequence. The learner will continue to repeat this planning procedure until it arrives to the destination.

Since the learner has differing dynamics from the teacher, as the learner executes the command sequence to follow the composed path, it may deviate from it. When the learner is in an open area, such deviation is not critical because the command sequence only lasts a short period of time and it can always be corrected by the planner at the next planning step. However, such deviation can compromise the safety of the learner when it maneuvers in a cluttered environment. To provide safety guarantees to the system, we introduce an event triggered mechanism to monitor the learner at runtime. The runtime monitor measures the distance between the learner and the planned path $d_{\hat{e}}$. The re-planning procedure is triggered when $d_{\hat{e}} > \epsilon$. The smaller that the threshold ϵ is, the more conservative the learner behaves. As we discussed, the learner does not need to constantly re-plan if the deviation happens in an open area. Thus, the threshold ϵ should be dynamically changed to reflect how crowded the surroundings are. In our work, the threshold is defined as:

$$\epsilon = \begin{cases} \eta * \min(||p - o_i||) & i = 1, 2, \ldots, N_o, \\ \infty & i = \varnothing, \end{cases} \tag{11}$$

where N_o is the number of obstacles in the learner's field of view, o_i is the position of obstacle i, and η is a constant.

V. Simulations

For the simulations, we created a general case study which, we believe, is rich enough to represent the problems we are dealing with. With the following case studies we demonstrate how, thanks to our approach, a robot can quickly adapt to downgraded dynamics due for example to a failure or system's aging. In this case, the teacher is a vehicle with full capabilities while the learner is the same vehicle whose dynamics are compromised. For ease of implementation, we consider that both the teacher and the learner have small inertia thus the acceleration period can be neglected (e.g., an electric vehicle). The kinematics for both the teacher and the learner are given by the following bicycle model:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} (v \cdot v_{max}) \cdot \cos\theta \\ (v \cdot v_{max}) \cdot \sin\theta \\ \gamma \cdot \gamma_{max} \end{bmatrix}, \quad \boldsymbol{u} = \begin{bmatrix} v \\ \gamma \end{bmatrix}, \tag{12}$$

where v_{max} and γ_{max} denote the maximum capability on velocity and steering angle of the vehicle. The learner's model is treated as a black box which takes in a control input and produces the updated state of the learner. A Gaussian noise of $G \sim \mathcal{N}(0, 0.1)$ is added to the learner's position to simulate measurement errors. Since the teacher and the learner are the same vehicle, the range of the control inputs for both of the vehicles are same which are $\boldsymbol{u}=\{v, \gamma \mid v \in [0, 1], \gamma \in [-1, 1]\}$. However, the learner is downgraded so that it can not achieve the same level of performance as the teacher when it is given the same command. In this case study, the maximum velocity v_{max} of the learner is downgraded from 3 m/s to 1 m/s while the maximum steering angle γ_{max} is downgraded from $\pi/3$ rad/s to $\pi/8$ rad/s. For example, the same control input v=1 drives the teacher at 3 m/s while the learner can only drive at 1 m/s. The learner is asked to follow a "S"-shaped trajectory while navigating through a cluttered environment.

Fig. 7 shows two snapshots within the time frame of the entire simulation. As the result shows, the learner is able to closely follow the desired trajectory. The learner behaves more conservatively when the obstacles are within the field of view (FOV). In order to obtain the results in Fig. 7, a sequence of 5×5 grid commands were fed to the learner. Based on the change of the states before and after executing the command, an equivalent teacher command is retrieved and paired with learner's input. All the command pairs are shown in Fig. 8. The boundary of the commands on teacher's command space marks the limitation of the learner. The learner can map the teacher's command which falls in the boundary to get the learner's control input, and the mapped control input will produce a similar maneuver as the teacher.

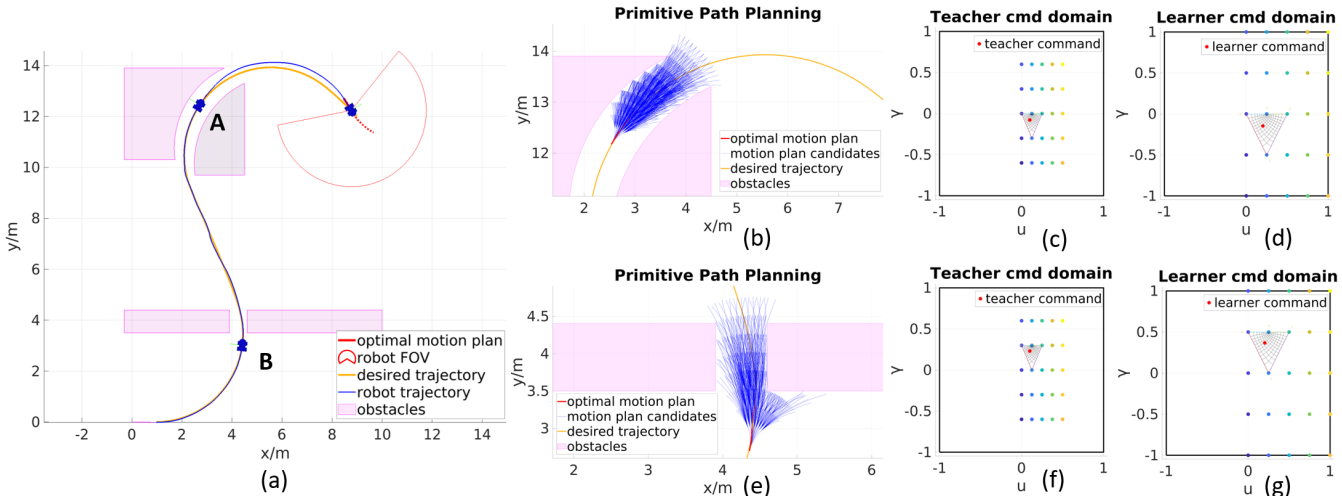

Fig. 7. The path following result of the entire simulation is depicted in (a). The local path planning of the SCM mapping results for the robot at position '**A**' are shown in (b), (c), (d), and the results at position '**B**' are shown in (e), (f), and (g).

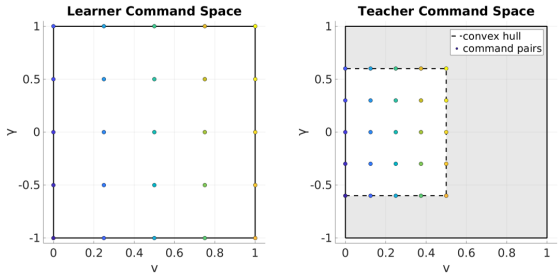

Fig. 8. The command pairs are one-to-one color coded across the two command domains.

Fig. 9 shows all the teacher's motion primitives and the corresponding commands. Each of the primitives are constructed by driving the teacher with a certain control input for 1 second. The command pairs on the boundary of the convex hull are used to identify if the command for building the motion primitive is within the learner's capability. Among all the 121 motion primitives, 35 of them are preserved after the motion degradation and used for path planning.

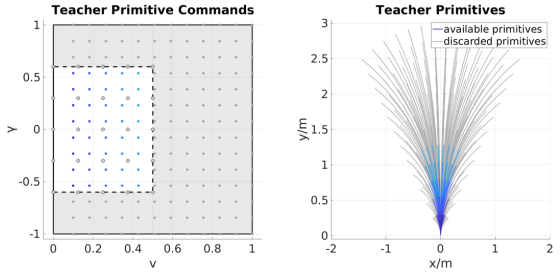

Fig. 9. The primitives associated with the small gray commands in shaded area are beyond the limitation of the learner and thus are discarded. The available motion primitives and the associated commands are color coded.

For the path planner, we set the planning horizon to s=2 and the threshold to trigger re-planning as η=0.5. In Fig. 10, we show the result of the learner driving directly with the teacher's commands without using our proposed approach. As expected, the learner failed because it used commands not adapted to its new dynamics.

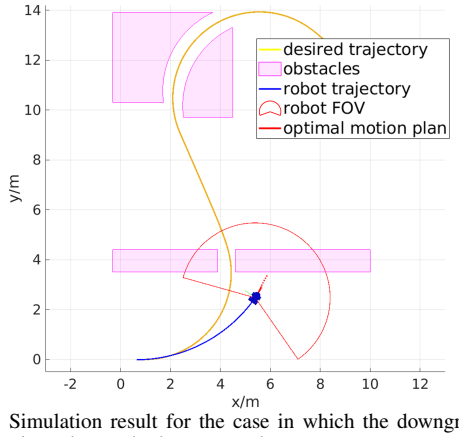

Fig. 10. Simulation result for the case in which the downgraded learner is directly given the teacher's commands.

VI. Experiments

Our proposed transfer learning approach was validated by a set of experiments in which we transferred the planning and control knowledge of a simulated teacher into two real learner vehicles. The video of all experiments are available in the provided supplemental material. In each of the experiments, we used the same simulated teacher vehicle. The vehicle dynamic model can be approximated to the one showed in the simulation experiments. The maximum velocity v_{max} and the maximum steering angle γ_{max} of the teacher were set to be 1.6 m/s and $\pm$1.2 rad/s respectively. The proposed method was implemented in MATLAB and we used the MATLAB ROS Toolbox together with Robot Operating System (ROS) to control the vehicles. We used MATLAB Schwarz-Christoffel toolbox [18] for computing the mapping function. The experiments were conducted in the indoor environment and the state of the vehicles are captured by a VICON motion capture system.

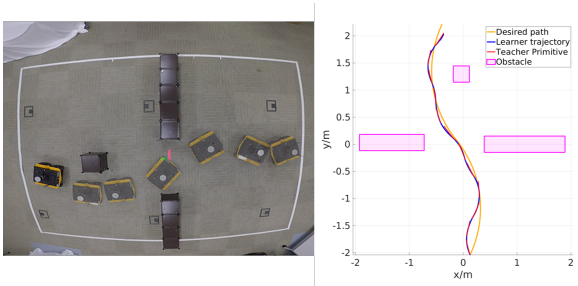

Fig. 11. Jackal experiment with SCM.

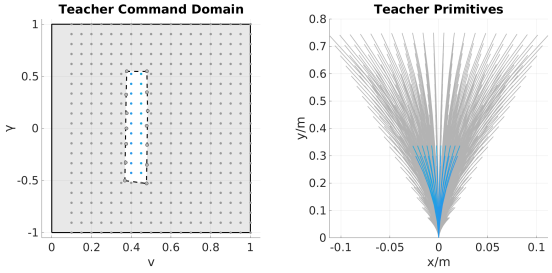

Fig. 12. The Jackal's capability is indicated within the white area. The gray points on the dashed boundary are the commands that were tested on the Jackal for extracting the limitations. The blue colored commands on the left create the primitives on the right and are used for mapping to the real UGV.

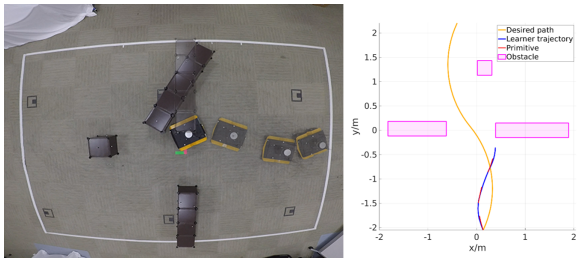

Fig. 13. Jackal experiment by directly feeding teacher's command.

For the first experiment, we asked the learner vehicle to follow an S-shaped path with the initial heading of $\frac{\pi}{4}$ from the desired orientation. As shown in Fig. 11, a narrow gate and an obstacle was set along the path. Using a Clearpath Jackal UGV as the learner vehicle, we tested its capability by sending certain commands over a period of 1 second, and based on the change to the state, we retrieved the equivalent teacher commands. The command pairs and the teacher's primitives that were used to plan the learner's path are demonstrated in Fig. 12. During the tracking mission, the maximum distance between the desired path and the actual trajectory was recorded as 0.1905 m and the maximum deviation between the actual trajectory and the local motion

plan was 0.0293 m. Considering the vehicle's initial heading is not aligned with the desired path and the size of the vehicle is approximately 0.5 m$\times 0.43$ m$\times 0.25$ m, the maximum deviation was negligible. For comparison, the same experiment without the SCM component was performed. As expected and as shown in Fig. 13, the learner vehicle collided with the gate and could not continue its task. Additionally, it can be clearly seen that there was a mismatch between the learner's trajectory and the primitive which was given by the path planner. This is also due to the fact that the teacher's control input was not mapped to the learner.

To show the generalizability of our proposed framework, similar to the experiment with the Jackal UGV, we performed another experiment with the same settings but this time using a Turtlebot2 as learner. The command pairs and the primitives which were used for learner path planning are shown in Fig. 14. The result shows that with our proposed approach, the Turtlebot2 could adapt the teacher controller and path planner to track the desired path with the maximum deviation of 0.1381 m. The tracking error between the vehicle's trajectory and the local planned primitive was small within 0.0978 m as can be noted in the figure in which the blue and the red segments are nearly overlapping throughout the whole process.

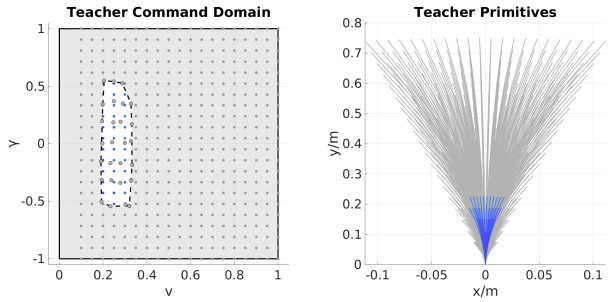

Fig. 14. Similar to the Jackal experiment, the turtlebot experiment command pairs and primitives are shown in the figure.

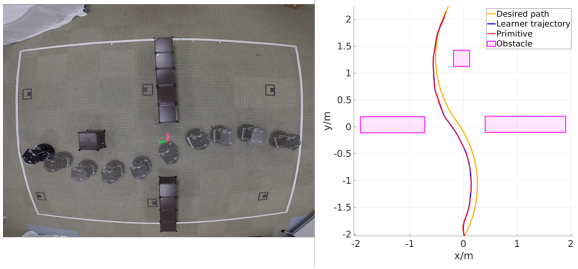

Fig. 15. Turtlebot experiment with SCM.

VII. Conclusion and Future Work

In this work, we proposed a novel light-weight transfer learning framework based on conformal mapping. We use SCM to directly map the control input from the teacher to the learner without knowing the dynamical model of the learner. The framework transfers not only the control policy but also adapts the teacher's motion planning policy to make it compatible with the learner. The proposed method is validated with both simulations and actual experiments. The results show that the learner can safely adapt the control and motion planning policy to suit its own dynamics.

In our future work, we are looking into leveraging multi-dimensional conformal mapping to transfer from a higher-order system to a lower-order system, such as from an aerial vehicle to a ground vehicle. We plan also to extend our framework to deal with learners that have more capabilities than the teacher.

VIII. Acknowledgements

This work is based on research sponsored by DARPA under Contract No. FA8750-18-C-0090.

References

[1] H. Chiang, N. Malone, K. Lesser, M. Oishi, and L. Tapia, "Path-guided artificial potential fields with stochastic reachable sets for motion planning in highly dynamic environments," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 2347–2354.

[2] F. Duchoň, A. Babinec, M. Kajan, P. Beňo, M. Florek, T. Fico, and L. Jurišica, "Path planning with modified a star algorithm for a mobile robot," *Procedia Engineering*, vol. 96, pp. 59–69, 2014.

[3] J. D. Marble and K. E. Bekris, "Asymptotically near-optimal planning with probabilistic roadmap spanners," *IEEE Transactions on Robotics*, vol. 29, no. 2, pp. 432–444, 2013.

[4] L. Pacheco and N. Luo, "Testing pid and mpc performance for mobile robot local path-following," *International Journal of Advanced Robotic Systems*, vol. 12, no. 11, p. 155, 2015.

[5] T. A. Driscoll and L. N. Trefethen, *Schwarz-christoffel mapping*. Cambridge University Press, 2002, vol. 8.

[6] S. James, P. Wohlhart, M. Kalakrishnan, D. Kalashnikov, A. Irpan, J. Ibarz, S. Levine, R. Hadsell, and K. Bousmalis, "Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 627–12 637.

[7] C. Devin, A. Gupta, T. Darrell, P. Abbeel, and S. Levine, "Learning modular neural network policies for multi-task and multi-robot transfer," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 2169–2176.

[8] D. J. Fremont, E. Kim, Y. V. Pant, S. A. Seshia, A. Acharya, X. Bruso, P. Wells, S. Lemke, Q. Lu, and S. Mehta, "Formal scenario-based testing of autonomous vehicles: From simulation to the real world," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2020, pp. 1–8.

[9] J. Zhang, B. Cheung, C. Finn, S. Levine, and D. Jayaraman, "Cautious adaptation for reinforcement learning in safety-critical settings," in *International Conference on Machine Learning*. PMLR, 2020, pp. 11 055–11 065.

[10] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 3803–3810.

[11] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-real: Learning agile locomotion for quadruped robots," *arXiv preprint arXiv:1804.10332*, 2018.

[12] R. Ramakrishnan, E. Kamar, D. Dey, E. Horvitz, and J. Shah, "Blind spot detection for safe sim-to-real transfer," *Journal of Artificial Intelligence Research*, vol. 67, pp. 191–234, 2020.

[13] S. Ghosh, S. Bansal, A. Sangiovanni-Vincentelli, S. A. Seshia, and C. Tomlin, "A new simulation metric to determine safe environments and controllers for systems with unknown dynamics," in *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, 2019, pp. 185–196.

[14] S. R. Branavan, H. Chen, L. S. Zettlemoyer, and R. Barzilay, "Reinforcement learning for mapping instructions to actions." Association for Computational Linguistics, 2009.

[15] K. Hwang, W. Jiang, and Y. Chen, "Adaptive model learning method for reinforcement learning," in *2012 Proceedings of SICE Annual Conference (SICE)*, 2012, pp. 1277–1280.

[16] G. Notomista and M. Egerstedt, "Coverage control for wire-traversing robots," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 5042–5047.

[17] P. F. Byrd and M. D. Friedman, *Handbook of elliptic integrals for engineers and scientists*. Springer-Verlag, 1971.

[18] T. A. Driscoll, "Algorithm 843: improvements to the schwarz-christoffel toolbox for matlab," *ACM Transactions on Mathematical Software (TOMS)*, vol. 31, no. 2, pp. 239–251, 2005.