The State of Robot Motion Generation

Kostas E. Bekris, Joe Doerr, Patrick Meng, Sumanth Tangirala

Computer Science Dept., Rutgers University, New Brunswick NJ 08901, USA
kostas.bekris@cs.rutgers.edu

Abstract. This paper first reviews the large spectrum of methods for generating robot motion proposed over the 50 years of robotics research culminating to recent developments. It crosses the boundaries of methodologies, which are typically not surveyed together, from those that operate over explicit models to those that learn implicit ones. The paper concludes with a discussion of the current state-of-the-art and the properties of the varying methodologies highlighting opportunities for integration.

Keywords: robot motion generation, task and motion planning, control, imitation learning, reinforcement learning, foundation models

1 Introduction

The robotics community is grappling with a critical question. Will the emerging set of data-driven methods for generating robot motion supersede the traditional techniques as access to robot motion data increases?

In this context, this paper review methods for robot motion generation, which are classified into those that operate given an explicit model vs. those that implicitly learn one from data. Explicit models can correspond to analytical expressions for the world geometry and dynamics or an explainable, numerical approximation in the form of a simulator. Motion generation given explicit models is maturing and methods are being deployed on real systems, such as autonomous vehicles and industrial manipulators. At the same time, there is increasing excitement for data-driven methods, which have been demonstrated to perform complex tasks, such as dexterous manipulation and unstructured locomotion. These methods do not depend on explicit models. Instead, they learn implicit representations, which are stored in the internal parameters of machine learning models.

Given the challenge of comprehensively reviewing the vast amount of work in this area across disciplinary boundaries in a short manuscript, the focus is on breadth rather than diving deeply into specific methodologies. Similarly, the focus is on principles that are applicable across robotic platforms instead of techniques that are specific to certain robot hardware morphologies.

This paper concludes with a discussion regarding the properties of the various robot motion generation methods. It argues that integrative approaches and closer interactions between different sub-communities can help develop more robust, safe solutions that can be reliably deployed at reasonable costs and human engineering effort.

2 Motion Generation given Explicit Models

Figure 1 presents a classification of methods that operate over an explicit model. **Motion planning** methods aim to generate safe nomimal paths/trajectories

Manuscript 76 submitted to 2024 International Symposium on Robotics Research.
Received August 5, 2024.

to a goal given a fully-observable world model. **Task and motion planning** extends the same principle to generate motion for tasks that require sequencing multiple goals. Such solutions can be executed open-loop, if the underlying model is accurate. Robot models, however, are imperfect, resulting in failures upon execution. Given this challenge, **planning under uncertainty** methods aim to compute policies that are robust to model-able disturbances. Alternatively, **control and feedback-based planning** methods tightly integrate perception and motion generation, so that the robot dynamically reacts to any deviation from desired behavior given the latest observations.

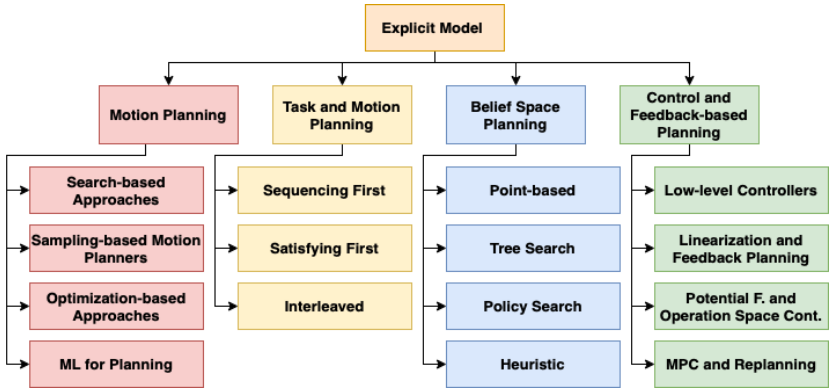

Fig. 1. Robot motion generation methods that operate over an explicit model.

2.1 Motion Planning

The typical objective of motion planning is to identify the shortest path or the fastest trajectory that brings a robot to a desirable goal state without undesirable collisions given a fully-observable world model.

Search-based Approaches: Uninformed search methods, such as Uniform Cost Search (UCS) or Dijkstra's algorithm [1] can be used to compute optimal paths over a discrete representation of the state space in the form of a grid or a graph given a cost function. Informed alternatives, such as A* [2], utilize heuristic cost-to-go estimates from each state to accelerate solution discovery and can still return the optimum solution over the discrete representation if the heuristic is admissible. Due to the comprehensive nature of search and the discrete representation, search methods suffer from the curse of dimensionality, i.e., the possible states to be explored grow exponentially with dimensions, rendering them computationally infeasible for maby robotics problem given naïve discretizations. There have been, however, many successful applications of search methods in robotics, such as planning for autonomous vehicles [3] and single or dual arm planning [4].

Sampling-based Motion Planners (SBMPs): Sampling has proven successful in providing graphical representations for searching the collision-free subset of a robot's state space in a more scalable manner than grids. The Probabilistic Roadmap Method (PRM) [5] samples collision-free configurations as nodes of a roadmap and collision-free local paths define the edges. The roadmap can then be used to solve multiple queries via search. For specific queries, the Rapidly

Manuscript 76 submitted to 2024 International Symposium on Robotics Research.
Received August 5, 2024.

Exploring Random Tree (RRT) [6] generates a tree data structure rooted on the robot's start state to quickly explore the free state space until it reaches the goal's vicinity. Asymptotically optimal variants (PRM*, RRT*) guarantee that as sampling progresses, the paths discovered converge to optimal ones [7]. RRT does not require a "steering function", i.e., the ability to perfectly connect two robot states, a primitive required by other methods, allowing it to deal with dynamical systems when a "steering function" is not available. RRT is provably suboptimal. Recent planners, however, achieve asymptotic optimality for kinodynamic problems ([8]). SBMPs have been applied for autonomous driving [9] and manipulation [10]. The Open Motion Planning Library (OMPL) provides software implementations for most SBMPs [11].

Optimization-based Approaches: The above motion planning methods are comprehensive in nature, which can lead to increased solution times. Moreover, their basic instances don't utilize gradient information for identifying high-quality solutions. The alternative is to locally optimize paths or trajectories for robotic systems given an objective function under physical or operational limits, such as collision avoidance and control bounds. Covariant Hamiltonian Optimization for Motion Planning (CHOMP) [12] demonstrated this principle by iteratively optimizing paths by reducing collision and trajectory costs. TrajOpt [13] utilizes sequential convex optimization and progressively ensures that each iteration produces feasible and collision-free trajectories. k-Order Markov Optimization (KOMO) [14] approaches trajectory optimization as a sparse non-linear program and tries to address high-dimensional problems by leveraging sparsity in dynamics and constraints. Factor graphs, a graphical optimization tool rooted in state estimation, and least-squares optimization can also be applied for trajectory optimization [15]. A recent approach, the Graph of Convex Sets [16], combines optimization and SBMPs. It builds a graphical structure where nodes are convex regions of the free space and optimization finds a trajectory over the set of nodes connecting the start with the goal. Overall, when optimization techniques work they tend to find high-quality solutions fast. They can suffer, however, from local minima, which arise from the highly non-linear and non-convex nature of robotics problems.

Machine Learning (ML) for Planning: ML can be used to improve the computational efficiency of planning given an explicit model [17]. Some approaches focus on components of planners, such as effective sampling [18], avoiding collisions [19] or providing distance metrics [20]. ML can also determine which combination of methods is best suited for a particular problem [21] or when a solution is not feasible [22]. *Neural Motion Planning (NMP)* [23] employ neural networks to approximate the operation of a planner. An encoder processes environmental data, like point clouds, to create a compact representation in a latent space. Then, a planning network predicts the robot's next configuration based on its current state, the goal state, and the encoded environment.

2.2 Task and Motion Planning (TAMP)

TAMP methods target long-horizon and multi-step robotic tasks, which may include moving through a sequence of goals or manipulating the environment [24].

Manuscript 76 submitted to 2024 International Symposium on Robotics Research.
Received August 5, 2024.

TAMP methods typically define low-level operators with motion constraints and high level logical relationships between the operators. Operators can contain hybrid discrete and continuous parameters, motion constraints, preconditions and effects, which make use of manually defined lifted variables. Planning is then performed via search across these lifted states. Possible state transitions are operators with satisfied preconditions, resulting in a sequence of operators called a plan skeleton. The hybrid parameters in the plan skeleton (e.g., start and goals) must be solved along with the low level trajectories of the operators. TAMP methodologies can be categorized by the order in which they sequence and satisfy the operators [24]: **Sequencing first** [25] defines a plan skeleton, i.e., a sequence of operators without satisfy their preconditions; this can cause frequent infeasible plans due to the lifted variables not being descriptive enough about underlying constraints. **Satisfying first** trajectories for operators, then planning with them, can solve this problem, but can spend time creating useless satisfied operators due to not having a plan skeleton to direct the process [26]. **Interleaved** sequencing and satisfying methods can provide a mix of both by checking low level information for feasibility during task planning [27]. TAMP critically relies on engineering effort to define preconditions and effects that properly characterize operator behavior and expressive lifted variables. Basic TAMP approaches also struggle under partial observability and uncertainty, which is the focus of recent efforts [28].

2.3 Belief Space Planning

The above planning methods assume a deterministic, perfect world model. Sensing noise and inaccurate execution introduce uncertainty, however, and the need to generate robust motions to such disturbances. If these sources of noise can be modeled in a probabilistic manner, (Partially Observable) Markov Decision Processes (PO)MDPs provide a problem formulations that reasons about uncertainty. (PO)MDPs employ belief distributions for the problem representation, i.e., probability distribution over states. They can be integrated with Bayesian state estimation (e.g., Kalman or particle filters) that return such beliefs. A key distinction is that solutions to (PO)MDPs are not nominal paths, but policies that map actions to beliefs. Due to the careful consideration of uncertainty, computing the exact optimal solution to a (PO)MDP problem is computationally intractable [29], especially for robotics problems that involve continuous state and action spaces.

This issue has motivated the development of approximate solutions for robotic (PO)MDPs. The Successive Approximations of the Reachable Space under Optimal Policies (SARSOP) algorithm [30], is an instance of **point-based** value iteration, and uses offline sampling to approximate (PO)MDPs solutions by focusing on representative beliefs and determining the best action to perform given only the sampled set. The Determinized Sparse Partially Observable Trees (DESPOT) [31] method employs **tree search** online to compute the optimal action at the initial state. It is also possible for an online algorithm to perform **policy search** by utilizing a controller in a limited search space increasing solution scalability at the cost of difficulty in bounding it [32]. **Heuristic** methods

Manuscript 76 submitted to 2024 International Symposium on Robotics Research.
Received August 5, 2024.

use rules or assumptions that reduce planning complexity by ignoring long-term consequences to focus on immediate gains or the most likely outcomes. For instance, Pre-image Back Chaining constructs a sequence of actions and states (pre-images) leading backwards from a goal state to the current state [33]. Generalized Belief Space (GBS) planning dynamically adapts to the ongoing discovery of an unknown environment [34].

2.4 Control and Feedback-based Planning

Instead of modeling uncertainty as in belief-space planning to address the model gap, control employs a tighter, closed-loop integration between state estimation and motion generation. Thus, it focuses on defining policies that are reactive to the different possible outcomes that may be observed during execution.

Low-level Controllers: *Proportional Integral Derivative (PID)* control and related tools are simple feedback strategies that are robust once properly tuned. This makes them ubiquitous for tracking desirable robot controls provided by higher-level motion generation processes. *Path Tracking Controllers* dynamically select controls given the latest state estimates to minimize deviations from a desired path. These controllers, however, are myopic and do not reason about the desired goal.

Linearization and Feedback-based Planning: Principles of linear control can be applied for robot motion generation. The *Linear Quadratic Regulator (LQR)* provides a an optimal solution for linear time-invariant systems given a quadratic cost function as a control law of the form $u(t) = -K \cdot x(t)$. Most robots, however, are non-linear and time-varying, while problems involve complex, non-quadratic objectives. *Feedback linearization* locally transforms nonlinear systems into equivalent linear ones, so that the linear control law can be applied. It can be used for track states x_d along a desired trajectory, where the control input is designed to minimize the error $e = x(t) - x_d(t)$. Given the linearization, these solutions tend to work only in the vicinity of the desired goal/trajectory. In order to expand the set of initial conditions from which the goal can be reached, *sequential composition* of such feedback policies [35] gives rise to hybrid controllers that sequentially switch between them. *LQR-Trees* [36] apply sequential composition by combining linear control and SBMPs. They uses control verification tools to evaluate the region of attraction (RoA) of local LQR controllers. They expand a tree backwards from the goal and sample controllers that bring the robot within the RoAs of controllers that already lead to the goal. Recent advancements are considering dynamic environments [37] and systems with dynamics [38].

Potential Functions and Operational Space Control: *Potential functions* [39] define attractive fields towards the goal and repulsive ones that push away from obstacles. Moving along the negative gradient of the sum of these fields provides the motion vector. Some engineering effort is needed to tune the parameters of the potentials. In complex environments, the corresponding potential may have multiple minima and goal convergence is not guaranteed. *Navigation functions* [40] are smooth and ensure a single minimum at the goal but they are more complicated to design and can only be constructed for specific environments (i.e., sphere and star worlds). Such control policies do not have to

Manuscript 76 submitted to 2024 International Symposium on Robotics Research.
Received August 5, 2024.

be defined directly in the robot's state space. *Operational Space Control* [41] defines such control laws in lower-dim. task spaces. For instance, if the task involves the robot's end-effector, a control law is defined to move the end-effector towards a desired goal unencumbered by other robot constraints. Additional control laws can be defined so that links avoid collisions. The corresponding motion vectors for the individual links are then mapped and integrated to a state space motion for the robot via the pseudo-inverses of the robot's Jacobian matrices. This allows for *multi-level hierarchical control* [42], where a hierarchy can be imposed over constraints, operational tasks, and soft objectives, so that lower priority objectives are solved in the null space of higher priority ones once projected to the state space. The above strategies rely on precise robot models and high-quality sensing to provide accurate feedback on robot state.

Model-Predictive Control (MPC) and Replanning: MPC is a simple but powerful feedback strategy that aims to solve fast and repetitively finite horizon optimization problems given the latest state observations [43]. MPC executes the initial portion of the motion and then recomputes given the latest observation. During each step, MPC uses the system's model to predict future behavior and makes control decisions that minimize a cost function at the end of the finite horizon, while adhering to constraints, such as collision-avoidance. Initial MPC approaches used analytical methods to predict future states and Linear MPC utilizes linear analytical models or approximations. Shooting MPC employs numerical methods, such as guessing an initial set of controls and then locally optimizing to minimize the cost function [44]. It is a general method applicable across robotic systems that helps to bridge the model gap. At the same time, shooting MPC involves numerous parameters that require tuning to achieve good performance [45]. It is also possible to replan using longer horizon motion planners, where it is important to minimize computational cost so as to achieve tight feedback. For instance, D* Lite [46], a replanning variant of A*, updates the path it initially creates to adjust to changes in the workspace. Similarly, replanning versions of SBMPs reuse prior computations to ensure the robot reacts swiftly to changes while maintaining safety [47], including under dynamics constraints [48].

3 Data-driven Motion Generation with Implicit Models

Figure 2 presents a classification of data-driven robot motion generation methods that implicitly learn a model. **Learning from demonstrations** methods employ supervision, where a robot is tasked to best replicate the behavior shown in provided demonstrations. Alternatively, **reinforcement learning** methods learn to make decisions by experiencing rewards or penalties after interaction with the environment. **Cross-task learning** transfer knowledge from an existing solution to a new novel task or adapt it dynamically. Finally, the emerging area of **large models** given access to significant data allows for pretrained ML models that are can then be finetuned and deployed in diverse domains.

3.1 Learning from Demonstrations

Imitation learning mimics the decision-making process given demonstration data.

Manuscript 76 submitted to 2024 International Symposium on Robotics Research.
Received August 5, 2024.

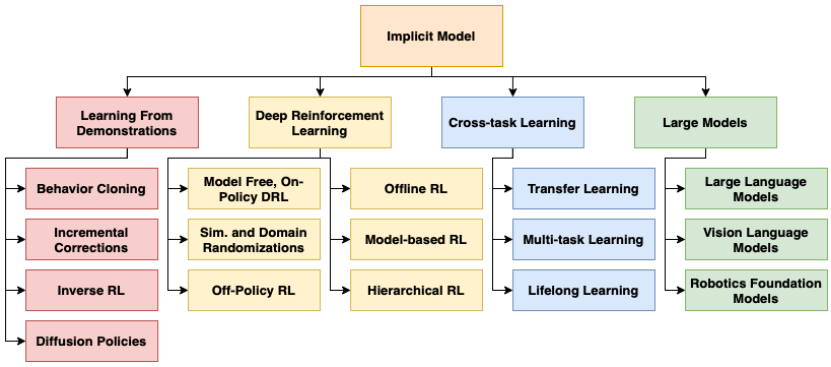

Fig. 2. Robot motion generation methods that operate over an explicit model.

Behavior Cloning (BC) trains a model in a supervised manner from a dataset of recorded demonstrations to provide a motion policy. There have been successful demonstrations of BC enabling robots to learn complex behaviors [49, 50]. The process starts with data collection, which includes capturing sensory inputs and corresponding control actions for solving a target task. An appropriate ML model architecture, such as Convolutional Neural Networks (CNNs) for visual data or Recurrent Neural Networks (RNNs) for sequential data, is then trained using supervision to minimize a loss function such as mean squared error. The model is optimized using algorithms like Stochastic Gradient Descent (SGD) or Adam. BC is limited by the gap between the demonstration and the execution setups. Noise and stochasticity in demonstrations, which may cover only a small subset of the state space, can also cause compounding distributional shift between training and testing [51]. Furthermore, BC is sensitive to engineering decisions, such as observation space, hyperparameters, and recurrent information [52]. These shortcomings motivate more sophisticated approaches to imitate expert data.

Incremental Corrections: The DAgger algorithm [53] focuses on decision making in out-of-distribution (OOD) states given the available demonstrations. When a BC policy performs a rollout and encounters OOD states, an expert, such as a human or a planner, is queried to provide the proper actions at these states. These new datapoints are aggregated into the dataset and the policy is retrained. This is an iterative process as new OOD states will be encountered after each retraining. DAgger heavily relies on an expert to provide the corrections. SafeDAgger provided a more query-efficient extension [54]. It first uses a safety policy to predict the error of the learned policy. This safety policy is used to select only a small subset of training examples to be collected.

Inverse Reinforcement Learning (IRL) focuses on extracting the problem's underlying dense reward function given demonstrations so as to achieve improved generalization relative to BC [55]. This reward function can then be used to extract the policy given an MDP problem formulation. Initial works [56] modeled the reward as a linear combination of input features, and the weights of this linear combination are obtained by solving an optimization problem. A

Manuscript 76 submitted to 2024 International Symposium on Robotics Research.
Received August 5, 2024.

significant challenge of IRL, however, is the difficulty in scaling to high-dim systems, which recent approaches attempt to mitigate [57].

Diffusion Policies: Demonstrations commonly include multi-modal behaviors, which may be difficult to fit into a multi-layer perceptron (MLP) trained via mean squared error (MSE), which averages the demonstrated actions resulting in erroneous choices [58]. In computer vision, diffusion models have become popular for generating images by expressing multi-modal distributions [59]. This motivates the use of diffusion processes as the implicit models for BC, where Diffusion Policy [60] has outperformed alternatives, such as LSTM-GMM in learning from demonstrations. Inference speed of diffusion policies is a critical challenge for robot motion generations [61] and is an active area of research.

3.2 Deep Reinforcement Learning (DRL)

In robotics the focus is on Deep Reinforcement Learning (DRL), where Deep Neural Networks (DNN) predict the Q function and the corresponding policy, which has been applied across tasks, including grasping [62], locomotion [63], and assembly [64].

Model-free, On-Policy DRL: Standard DRL gathers data directly from interaction with the environment, where Policy Gradient (PG) methods generate a batch of full trajectory data, compute the gradient of each state transition, and scale each gradient by the respective discounted return. PGs update the policy and repeat the process towards maximizing the expected cumulative reward. Proximal Policy Optimization (PPO) [65] is a PG method that leverages a value function, which is the average discounted return from a given state. These are on-policy methods, i.e., they only use data collected from the most recent policy. Exploration can be achieved through the periodic execution of random actions (epsilon greedy), injecting Gaussian noise to policy outputs [66], or using a stochastic policy that aims for entropy maximization [65].

DRL faces *multiple challenges* in robotics: (i) sample inefficiency: Policies need a lot of data to train and take significant training time. Data on real robots can be extremely slow, expensive, and unsafe to collect [67]; (ii) instability: Policies can be inconsistent across training sessions due to poorly designed rewards, exploration strategies, learning rate parameters or learning error from DNNs; (iii) reward engineering: The simplest rewards are sparse, i.e., they assign a 0 reward everywhere except at the goal. While desirable, such sparse rewards often do not find a successful behavior in reasonable time [67]. This motivates dense rewards to guide exploration, which involves manual engineering. While they can improve training time and stability, they can also lead to wrong behaviors if they do not match the true task objective; (iv) long horizon tasks: Basic RL struggles with long horizon tasks as the state space to be explored is even larger and is difficult to propagate rewards across subproblems. The above challenges have motivated multiple DRL variations in robotics as described below.

Simulation and Domain Randomization: Simulators, which are explicit world models, are useful for addressing sample inefficiency by generating data without real-world experiments. They suffer, however, from the model gap issue. Even so, it may still be possible to create good policies via *domain randomization*

Manuscript 76 submitted to 2024 International Symposium on Robotics Research.
Received August 5, 2024.

(DR) [68], which can partially compensate for imperfect models during sim-to-real transfer. DR results in behaviors that have the highest expected reward across a variety of underlying dynamics and perception errors.

Off-policy RL Sample inefficiency motivates the reuse of data. Off-policy RL stores the expected discounted future reward of any in-distribution state-action tuple on a Deep Q function, which is used to train the policy. To update the parameters θ of the Deep Q function for a given state transition (s, a, r, s'), the immediate reward r and future rewards $Q(s', \pi_\phi(s'))$ are used to define the loss function: $L(\theta) = (Q_\theta(s, a) - r + \gamma \cdot Q_\theta(s', \pi_\phi(s')))^2$. This allows off-policy RL algorithms, such as TD3 [69] and SAC [70], to be more sample efficient than online RL. Off-policy RL, however, is complicated by the instability of DNN learning, especially because errors compound while learning Q functions [71]. A way to mitigate instability is to use target Q functions to query for future rewards [69] [70] [71]. The target Q functions average their weights slowly with the learned Q function's weights. Given that the policy is defined as $\pi^*(s) = \arg\max_a Q(s, a)$, errors in Q function learning, which overvalue actions, are propagated rapidly. This is called an overestimation bias [69]. DRL techniques use a clipped, double Q function, an ensemble of two Q networks coupled with target Q networks, that take the lower value of the two when queried for future rewards to underestimate Q values when uncertain [69] [70]. Hindsight Experience Replay (HER) [72] aims for sample-efficiency in multi-goal problems given only sparse rewards. It relabels the end or intermediate states of executed trajectories as the desired goal allowing to train over all available experiences.

Offline RL Offline RL is a subcase of off-policy RL, which doesn't interact online with the environment to modulate the replay buffer. Instead it uses offline data, focusing on sample efficiency. Offline RL aims to improve the optimality of suboptimal demonstrations by patching together data towards the most optimal solution rather than imitating the data [73]. Additionally, it can handle multi-modal demonstrations as it can solve for the single most optimal action for each state. The policy π used for updating the Q function is parameterized via a function approximator. This can cause π to choose actions that are out-of-distribution (OOD). The Q function queried with the OOD state-action pair will output an untrained value, which can be an erroneous large reward that can propagate and degrade performance [74]. In online RL, when an OOD state-action pair is overvalued, π will favor the action and gather data on the OOD state-action pair, which will remedy the error. IQL [75], which is an offline RL method, mitigates distributional shift by using only state-action pairs from the dataset. Instead, CQL [76] employs a conservative estimate on Q values.

Model-based RL learns a model of the environment, which is then used to roll out the policy to generate training data offline. Relative to a simulator, the learned model can operate over a more compact state representation, run faster, and can be queried for any given state. The major drawback is that RL may exploit errors in the learned model. MOPO [77] aims to mitigate these effects by detecting when the query to the model is out of distribution.

Manuscript 76 submitted to 2024 International Symposium on Robotics Research.
Received August 5, 2024.

Hierarchical RL (HRL) focuses on long horizon tasks, similar to TAMP (Section 2.2). It temporally abstracts high-level actions that correspond to low-level learned policy. The high-level actions can target exploration to better cover the space and more easily assign rewards [78]. Using random actions to search is not effective in long horizon tasks and guidance for their generation is critical. Long horizon tasks remain rather challenging for RL approaches.

3.3 Cross-task Learning

Demonstrations from related tasks or earlier instantiatios of a task can be useful in bootstrapping or guiding RL.

Transfer Learning can bootstrap RL by using an adaptively-weighted auxiliary term in the loss function of PPO to increase the similarity of the actions of a learned policy to those of a demonstrated one from a task with similar MDP [79]. It is also possible to pre-train multiple tasks with offline RL given a single task-conditioned policy [80]. During online training, the approach fine-tunes both the conditioning parameter and policy to automatically reset the tasks for autonomous real-world training. For transferring information across heterogeneous MDPs, researchers have tried mapping states and actions across MDPs [81] and transferring useful representations across domains [82].

Multi-Task Learning trains multiple tasks in parallel while sharing information across tasks to accelerate training and improve generalization [83]. Methods focus on how to select parameters to sharing so as to effectively transfer learned representations between tasks [84].

Lifelong Learning emphasizes learning a new task in a sequence by transferring information from previously learned ones without forgetting how to solve them. Retaining previous task information can be done by mixing previous data with new ones during retraining [85]. Functionally composed modular lightweight networks have been proposed to learn a large variety of combinatorially related tasks to solve novel combination tasks in a zero-shot manner [86].

3.4 Large Models

The emergence of **Large Language Models (LLMs)** pretrained on internet-scale data has opened new avenues for robot motion generation. LLMs are capable of semantic reasoning and planning, making them candidates for extracting task specifications and creating action models for task planning [87]. . LLMs have also been used to improve an existing action model to handle failure cases [88]. These methods notably do not generate constraints for low level motion in the operators as seen in TAMP. Furthermore, they have been used in conjunction with evolutionary optimization as code generators for defining rewards, which can then be used to acquire complex skills via reinforcement learning [89].

Progress has also delivered **Vision Language Models** (VLMs), which can integrate multi-modal information about visual input and language. Saycan [90] integrates a visual affordance model, which evaluates possible actions the robot can take, with a language model that interprets the user's commands and generates high-level action plans. VLMs have also been used for autonomous demonstration data generation over diverse sets of tasks [91].

Manuscript 76 submitted to 2024 International Symposium on Robotics Research.
Received August 5, 2024.

This progress leads towards vision-language-action models (VLAs), which could constitute **Robotics Foundation Models**, i.e., large pretrained models that associate actions to sensing data and language task specifications, which can be fine-tuned for specific tasks to provide improvements in training time and generalization over training a model from scratch. OpenVLA [92] is such a model trained on a large robotic manipulation dataset [93] and utilizes large pretrained vision encoders and language models. It has been argued that it performs well on in-distribution tasks and robotic embodiments, while it can be fine-tuned for novel tasks and robotic embodiments.

4 Perspectives on Robot Motion Generation Research

Promises and Pitfalls of the Explicit Model Approach: These methods have a long history and they can reliably solve a variety of challenges today. For instance, planning collision-free motions for industrial arms among obstacles is rather reliably addressed today at relatively high speeds. Integration with state estimation also allows mobile robots to reliably execute navigation in semi-structured domains. They face challenges, however, in setups where it is difficult to achieve accurate models or accurate state estimation. Examples of such setups often involve the presence of complex contacts and partial observability, such as the manipulation of previously unknown objects in cluttered domains, locomotion over uneven terrains or navigation at high speeds or in dynamic, unstructured environments.

For long-horizon tasks, TAMP methods require an engineer to encode possible pathways and concepts before symbolic reasoning can ensue, which often involves expensive combinatorial reasoning. These methods are limited to the universe of variables defined and cannot easily anticipate changes from what has been programmed. While belief-space planning targets robustness to noise and partial observability, it require significant computation and access to accurate models of uncertainty, which is an even higher information requirement to achieve. At the same time, feedback-based solutions do find applicability in real world domains, either via MPC or hierarchical control strategies and bring the promise of also being able to describe the conditions under which a controller can solve a challenge.

Promises and Pitfalls of the Implicit Model Approach: The progress in ML promises effective data-driven motion generation methods that can access prior experience and which do not require an engineered model or even accurate state estimation. They have unblocked perception tasks, such as object detection and estimation, which are often prerequisites for robust motion generation. Learning from demonstration methods, especially those integrated with diffusion processes that allow to express multi-modal distributions, can achieve impressive results exactly in tasks that the traditional, explicit model, approaches are facing challenges, such as dexterous manipulation and locomotion. This is true, however, as long as the setup upon execution resembles the demonstration setup, thus limiting generalization. The various versions of reinforcement learning bring the promise of broader generalization and there have been many

Manuscript 76 submitted to 2024 International Symposium on Robotics Research.
Received August 5, 2024.

successful demonstrations of learning skills for robotic tasks via RL though sample inefficiency still remains a bottleneck for achieving highly accurate solutions across a wide set of initial conditions.

These limitations have motivated the robotics community to pursue the direction of collecting more data for robotics problems in lab environments, which may be diverse across embodiments and tasks [93], towards the objective of mimicking the success of foundation models in language and vision challenges. While this direction should be pursued, it is not clear that it is possible to collect internet-scale demonstration data that will allow learning robust enough policies that cover the space of possible tasks that robots need to solve in novel, unstructured and human environments. Furthermore, it is challenging to predict when the resulting solutions will be successful or not, which is a significant concern, as failures in robotics can cause physical harm.

Possible Integrative Directions: There is promise in the potential integration of robot motion generation solutions. For instance, training data-driven methods can also benefit from simulation setups, where the explicit model approaches act as the expert demonstrators under full observability and the data-driven methods learn to map sensing data to the actions demonstrated by the expert planners. This requires accurate enough models for the corresponding policies to be transferable to the real systems as well as training strategies that make these policies robust to the variety of conditions they can experience.

Upon deployment, the emerging data-driven methods must also be part of larger architectures that provide with safety and verification. Towards these objectives, they can benefit from integrating with components of explicit model methods. For instance, being able to describe the conditions under which a learned controller is successful, similar to control verification tools can allow the learned solutions' safe deployment. It can also assist in the composition of such controllers for addressing long-horizon tasks, where high-level symbolic reasoning can still be beneficial. Such task planning needs to be adaptive and allow a robot to dynamically define pre/postconditions, and discover new skills in the context of its task. It should also be accompanied by failure explanation to identify the reasons for why a problem is not solvable, which can help guide data collection or the type of reasoning that will assist in addressing similar challenges into the future. For such falsification as well as explainability purposes, maintaining an internal world model or a simulation, such as "cognitive physical engine", that is learned from data as well as from first principles can be useful.

A gap towards exploring and evaluating such integrative solutions is the lack of common interfaces, software components and benchmarks that allow to easily switch and experiment with components from different methodologies in the same context. Most of the existing instances of software infrastructure either support one set of methods or the other, requiring the ad hoc integration of tools for a solution that needs to borrow components from both families of methods.

Manuscript 76 submitted to 2024 International Symposium on Robotics Research.
Received August 5, 2024.

References

[1] E. W. Dijkstra, "A note on two problems in connexion with graphs," in *Numerische Mathematik*, 1959.

[2] R. Dechter and J. Pearl, "Generalized best-first search strategies and the optimality of A*," *Journal of the ACM*, 1985.

[3] M. Likhachev and D. Ferguson, "Planning long dynamically feasible maneuvers for autonomous vehicles," *IJRR*, 2009.

[4] B. Cohen, S. Chitta, and M. Likhachev, "Single- and dual-arm motion planning with heuristic search," *IJRR*, 2014.

[5] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *TRO*, 1996.

[6] S. LaValle and J. Kuffner, "Randomized kinodynamic planning," in *ICRA*, 1999.

[7] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *IJRR*, 2011.

[8] K. Hauser and Y. Zhou, "Asymptotically optimal planning by feasible kinodynamic planning in a state–cost space," *TRO*, 2016.

[9] J. hwan Jeon, R. V. Cowlagi, S. C. Peters, *et al.*, "Optimal motion planning with the half-car dynamical model for autonomous high-speed driving," in *ACC*, 2013.

[10] A. Perez, S. Karaman, A. Shkolnik, E. Frazzoli, S. Teller, and M. R. Walter, "Asymptotically-optimal path planning for manipulation using incremental sampling-based algorithms," in *IROS*, 2011.

[11] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *RAM*, 2012.

[12] M. Zucker, N. Ratliff, A. Dragan, *et al.*, "CHOMP: Covariant hamiltonian optimization for motion planning," *IJRR*, 2013.

[13] J. Schulman, J. Ho, A. X. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization," in *RSS*, 2013.

[14] M. Toussaint, "Newton methods for k-order Markov Constrained Motion Problems," arXiv:1407.0414, Tech. Rep., 2014.

[15] M. Mukadam, J. Dong, F. Dellaert, and B. Boots, "Steap: Simultaneous trajectory estimation and planning," *Autonomous Robots*, 2019.

[16] T. Marcucci, J. Umenberger, P. Parrilo, and R. Tedrake, "Shortest paths in graphs of convex sets," *SIOPT*, 2024.

[17] T. McMahon, A. Sivaramakrishnan, E. Granados, and K. E. Bekris, "A survey on the integration of machine learning with sampling-based motion planning," *FTR*, 2022.

[18] O. Arslan and P. Tsiotras, "Machine learning guided exploration for sampling-based motion planning algorithms," in *IROS*, 2015.

[19] B. Burns and O. Brock, "Sampling-based motion planning using predictive models," in *ICRA*, 2005.

[20] H.-T. L. Chiang, A. Faust, S. Sugaya, and L. Tapia, "Fast swept volume estimation with deep learning," in *WAFR*, 2020.

[21] M. Morales, L. Tapia, R. Pearce, S. Rodriguez, and N. M. Amato, "A machine learning approach for feature-sensitive motion planning," in *WAFR*. 2005.

[22] S. Li and N. Dantam, "Learning proofs of motion planning infeasibility," in *RSS*, 2021.

[23] A. H. Qureshi, A. Simeonov, M. J. Bency, and M. C. Yip, "Motion planning networks," in *ICRA*, 2019.

Manuscript 76 submitted to 2024 International Symposium on Robotics Research.
Received August 5, 2024.

[24] C. R. Garrett, R. Chitnis, R. Holladay, *et al.*, "Integrated task and motion planning," *Annual Review of Control, Robotics, & Autonomous Systems*, 2021.
[25] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel, "Combined task and motion planning through an extensible planner-independent interface layer," in *ICRA*, 2014.
[26] K. Hauser and V. Ng-Thow-Hing, "Randomized multi-modal motion planning for a humanoid robot manipulation task," *IJRR*, 2011.
[27] C. Dornhege, P. Eyerich, T. Keller, S. Trüg, M. Brenner, and B. Nebel, "Semantic attachments for domain-independent planning systems," *Towards Service Robots for Everyday Environments*, 2012.
[28] A. Curtis, G. Matheos, N. Gothoskar, *et al.*, "Partially observable task and motion planning with uncertainty and risk awareness," *RSS*, 2024.
[29] H. Kurniawati, "Partially observable markov decision processes (pomdps) and robotics," *CoRR*, 2021.
[30] J. Pineau, G. Gordon, and S. Thrun, "Point-based value iteration: An anytime algorithm for pomdps," in *IJCAI*, 2003.
[31] N. Ye, A. Somani, D. Hsu, and W. S. Lee, "Despot: Online pomdp planning with regularization," *JAIR*, 2017.
[32] H. Bai, D. Hsu, W. Lee, and N. Vien, "Monte carlo value iteration for continuous-state pomdps," in *WAFR*, 2010.
[33] L. P. Kaelbling and T. Lozano-Pérez, "Pre-image backchaining in belief space for mobile manipulation," in *ISRR*. 2017.
[34] V. Indelman, L. Carlone, and F. Dellaert, "Planning in the continuous domain: A generalized belief space approach for autonomous navigation in unknown environments," *IJRR*, 2015.
[35] R. R. Burridge, A. A. Rizzi, and D. E. Koditschek, "Sequential composition of dynamically dexterous robot behaviors," *IJRR*, 1999.
[36] R. Tedrake, I. R. Manchester, M. Tobenkin, and J. W. Roberts, "Lqr-trees: Feedback motion planning via sums-of-squares verification," *IJRR*, 2010.
[37] M. K. M. Jaffar and M. Otte, "Pip-x: Funnel-based online feedback motion planning/replanning in dynamic environments," in *WAFR*, 2022.
[38] C. K. Verginis, D. V. Dimarogonas, and L. E. Kavraki, "Kdf: Kinodynamic motion planning via geometric sampling-based algorithms and funnel control," *TRO*, 2022.
[39] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *ICRA*, 1985.
[40] E. Rimon and D. Koditschek, "Exact Robot Navigation Using Artificial Potential Functions," in *TRO*, 1992.
[41] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *JRA*, 1987.
[42] L. Sentis and O. Khatib, "Synthesis of whole-body behaviors through hierarchical control of behavioral primitives," in *IJHR*, 2005.
[43] E. Camacho and C. Alba, *Model Predictive Control*. 2013.
[44] A. Richards and J. P. How, "Real-Time Model Predictive Control: A Framework for Optimal Guidance and Control in Aerospace Systems," *JGCD*, 2004.
[45] W. Edwards, G. Tang, G. Mamakoukas, T. Murphey, and K. Hauser, "Automatic tuning for data-driven model predictive control," in *ICRA*, 2021.
[46] S. Koenig and M. Likhachev, "Improved fast replanning for robot navigation in unknown terrain," in *ICRA*, 2002.

Manuscript 76 submitted to 2024 International Symposium on Robotics Research.
Received August 5, 2024.

[47] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *IJRR*, 2002.
[48] K. E. Bekris and L. E. Kavraki, "Greedy but safe replanning under kinodynamic constraints," in *ICRA*, 2007.
[49] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser, "Tossingbot: Learning to throw arbitrary objects with residual physics," *TRO*, 2020.
[50] T. Zhang, Z. McCarthy, O. Jow, *et al.*, "Deep imitation learning for complex manipulation tasks from virtual reality teleoperation," in *ICRA*, 2018.
[51] M. Laskey, J. Lee, R. Fox, A. Dragan, and K. Goldberg, "Dart: Noise injection for robust imitation learning," in *CoRL*, 2017.
[52] A. Mandlekar, D. Xu, J. Wong, *et al.*, "What matters in learning from offline human demonstrations for robot manipulation," *CoRL*, 2021.
[53] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *AISTATS*, 2011.
[54] J. Zhang and K. Cho, "Query-efficient imitation learning for end-to-end simulated driving," in *AAAI*, 2017.
[55] S. Arora and P. Doshi, "A survey of inverse reinforcement learning: Challenges, methods and progress," *AIJ*,
[56] A. Y. Ng and S. J. Russell, "Algorithms for inverse reinforcement learning," in *ICML*, 2000.
[57] Y. Xu, W. Gao, and D. Hsu, "Receding horizon inverse reinforcement learning," in *NeurIPS*, 2022.
[58] T. Pearce, T. Rashid, A. Kanervisto, *et al.*, "Imitating human behaviour with diffusion models," *ICLR*, 2023.
[59] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *NeurIPS*, 2020.
[60] C. Chi, Z. Xu, S. Feng, *et al.*, "Diffusion policy: Visuomotor policy learning via action diffusion," in *RSS*, 2023.
[61] B. Kang, X. Ma, C. Du, T. Pang, and S. Yan, "Efficient diffusion policies for offline reinforcement learning," in *NIPS*, 2024.
[62] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *IJRR*, 2018.
[63] X. B. Peng, G. Berseth, K. Yin, and M. Van De Panne, "Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning," *TOG*, 2017.
[64] T. Inoue, G. De Magistris, A. Munawar, T. Yokoya, and R. Tachibana, "Deep reinforcement learning for high precision assembly tasks," in *IROS*, 2017.
[65] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv:1707.06347*, 2017.
[66] T. P. Lillicrap, J. J. Hunt, A. Pritzel, *et al.*, "Continuous control with deep reinforcement learning," *CoRR*, 2015.
[67] J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, and S. Levine, "How to train your robot with deep reinforcement learning: Lessons we have learned," *IJRR*, 2021.
[68] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *IROS*, 2017.
[69] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *ICML*, 2018.

Manuscript 76 submitted to 2024 International Symposium on Robotics Research.
Received August 5, 2024.

[70] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *ICML*, 2018.
[71] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, 2015.
[72] M. Andrychowicz, F. Wolski, A. Ray, *et al.*, "Hindsight experience replay," *NeurIPS*, 2017.
[73] A. Kumar, J. Hong, A. Singh, and S. Levine, "When should we prefer offline reinforcement learning over behavioral cloning?" *ICLR*, 2022.
[74] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," *arXiv:2005.01643*, 2020.
[75] I. Kostrikov, A. Nair, and S. Levine, "Offline reinforcement learning with implicit q-learning," *arXiv:2110.06169*, 2021.
[76] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative q-learning for offline reinforcement learning," *NeurIPS*, 2020.
[77] T. Yu, G. Thomas, L. Yu, *et al.*, "Mopo: Model-based offline policy optimization," *NeurIPS*, 2020.
[78] S. Pateria, B. Subagdja, A.-h. Tan, and C. Quek, "Hierarchical reinforcement learning: A comprehensive survey," *CSUR*, 2021.
[79] S. Schmitt, J. J. Hudson, A. Zidek, *et al.*, "Kickstarting deep reinforcement learning," *arXiv:1803.03835*, 2018.
[80] H. R. Walke, J. H. Yang, A. Yu, *et al.*, "Don't start from scratch: Leveraging prior data to automate robotic reinforcement learning," in *CoRL*, 2023.
[81] A. Gupta, C. Devin, Y. Liu, P. Abbeel, and S. Levine, "Learning invariant feature spaces to transfer skills with reinforcement learning," *ICLR*, 2017.
[82] J. Yang, C. Glossop, A. Bhorkar, *et al.*, "Pushing the limits of cross-embodiment learning for manipulation and navigation," *RSS*, 2024.
[83] Y. Zhang and Q. Yang, "A survey on multi-task learning," *TKDE*, 2021.
[84] X. Sun, R. Panda, R. Feris, and K. Saenko, "Adashare: Learning what to share for efficient deep multi-task learning," *NeurIPS*, 2020.
[85] D. Rolnick, A. Ahuja, J. Schwarz, T. P. Lillicrap, and G. Wayne, "Experience replay for continual learning," *NeurIPS*, 2019.
[86] J. A. Mendez, H. van Seijen, and E. Eaton, "Modular lifelong reinforcement learning via neural composition," *arXiv:2207.00429*, 2022.
[87] M. Han, Y. Zhu, S.-C. Zhu, Y. N. Wu, and Y. Zhu, "Interpret: Interactive predicate learning from language feedback for task planning," *RSS*, 2024.
[88] G. Chen, L. Yang, R. Jia, *et al.*, "Language-augmented symbolic planner for open-world task planning," *RSS*, 2024.
[89] Y. J. Ma, W. Liang, G. Wang, *et al.*, "Eureka: Human-Level Reward Design via Coding Large Language Models," *arxiv-2310.12931*, 2023.
[90] A. Brohan, Y. Chebotar, C. Finn, *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," in *CoRL*, 2023.
[91] J. Duan, W. Yuan, W. Pumacay, *et al.*, "Manipulate-anything: Automating real-world robots using vision-language models," *arXiv:2406.18915*, 2024.
[92] M. J. Kim, K. Pertsch, S. Karamcheti, *et al.*, "OpenVLA: An Open-Source Vision-Language-Action Model," *arXiv:2406.09246*, 2024.
[93] A. Padalkar, A. Pooley, A. Jain, *et al.*, "Open X-embodiment: Robotic learning datasets and RT-X models," *arXiv preprint arXiv:2310.08864*, 2023.

Manuscript 76 submitted to 2024 International Symposium on Robotics Research.
Received August 5, 2024.