Robotics and Autonomous Systems 57 (2009) 469–483

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

A survey of robot learning from demonstration

Brenna D. Argall [a,*], Sonia Chernova [b], Manuela Veloso [b], Brett Browning [a]

[a] Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA

[b] Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213, USA

ARTICLE INFO

Article history:
Received 23 May 2008
Received in revised form
15 October 2008
Accepted 25 October 2008
Available online 25 November 2008

Keywords:
Learning from demonstration
Robotics
Machine learning
Autonomous systems

ABSTRACT

We present a comprehensive survey of robot *Learning from Demonstration (LfD)*, a technique that develops policies from example state to action mappings. We introduce the LfD design choices in terms of demonstrator, problem space, policy derivation and performance, and contribute the foundations for a structure in which to categorize LfD research. Specifically, we analyze and categorize the multiple ways in which examples are gathered, ranging from teleoperation to imitation, as well as the various techniques for policy derivation, including matching functions, dynamics models and plans. To conclude we discuss LfD limitations and related promising areas for future research.

1. Introduction

The problem of learning a mapping between world state and actions lies at the heart of many robotics applications. This mapping, also called a *policy*, enables a robot to select an action based upon its current world state. The development of policies by hand is often very challenging and as a result machine learning techniques have been applied to policy development. In this survey, we examine a particular approach to policy learning, *Learning from Demonstration (LfD)*.

Within LfD, a policy is learned from *examples*, or demonstrations, provided by a teacher. We define examples as sequences of state–action pairs that are recorded during the teacher's demonstration of the desired robot behavior. LfD algorithms utilize this dataset of examples to derive a policy that reproduces the demonstrated behavior. This approach to obtaining a policy is in contrast to other techniques in which a policy is learned from *experience*, for example building a policy based on data acquired through exploration, as in Reinforcement Learning [1]. We note that a policy derived under LfD is necessarily defined only in those states encountered, and for those corresponding actions taken, during the example executions.

In this article, we present a survey of recent work within the LfD community, focusing specifically on robotic applications. We segment the LfD learning problem into two fundamental phases: *gathering* the examples, and *deriving* a policy from such examples. Based on our identification of the defining features of these techniques, we contribute a comprehensive survey and categorization of existing LfD approaches. Though LfD has been applied to a variety of robotics problems, to our knowledge there exists *no* established structure for concretely placing work within the larger community. In general, approaches are appropriately contrasted to similar or seminal research, but their relation to the remainder of the field lies largely unaddressed. Establishing these relations is further complicated by dealing with real world robotic platforms, for which the physical details between implementations may vary greatly and yet employ fundamentally identical learning techniques, or vice versa. A categorical structure therefore aids in comparative assessments among applications, as well as in identifying open areas for future research. In contributing our categorization of current approaches, we aim to lay the foundations for such a structure.

For the remainder of this section we motivate the application of LfD to robotics, and present a formal definition of the LfD problem. Section 2 presents the key design decisions for an LfD system. Methods for gathering demonstration examples are the focus of Section 3, where the various approaches to teacher demonstration and data recording are discussed. Section 4 examines the core techniques for policy derivation within LfD, followed in Section 5 by methods for improving robot performance beyond the capabilities of the teacher examples. To conclude, we identify and discuss open areas of research for future work in Section 6 and summarize the article with Section 7.

* Corresponding author. Tel.: +1 4122689923.
E-mail addresses: bargall@cs.cmu.edu (B.D. Argall), soniac@cs.cmu.edu (S. Chernova), mveloso@cs.cmu.edu (M. Veloso), brettb@cs.cmu.edu (B. Browning).

0921-8890/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.robot.2008.10.024

1.1. Support for demonstration learning

The presence of robots within society is becoming ever more prevalent. Whether an exploration rover in space, robot soccer or a recreational robot for the home, successful autonomous robot operation requires robust control algorithms. Non-robotics-experts may be increasingly presented with opportunities to interact with robots, and it is reasonable to expect that they have ideas about what a robot should do, and therefore what sort of behaviors these control algorithms should produce. A natural, and practical, extension of having this knowledge is to actually develop the desired control algorithm. Currently, however, policy development is a complex process restricted to experts within the field.

Traditional approaches to robot control model domain dynamics and derive mathematically-based policies. Though theoretically well-founded, these approaches depend heavily upon the accuracy of the world model. Not only does this model require considerable expertise to develop, but approximations such as linearization are often introduced for computational tractability, thereby degrading performance. Other approaches, such as Reinforcement Learning, guide policy learning by providing reward feedback about the desirability of visiting particular states. To define a function to provide the reward, however, is known to be difficult and requires considerable expertise to address. Furthermore, building the policy requires gathering information by visiting states to receive rewards, which is non-trivial for a robot learner executing actual actions in the real world.

Considering these challenges, LfD has many attractive points for both learner and teacher. LfD formulations typically do not require expert knowledge of the domain dynamics, which removes performance brittleness resulting from model simplifications. The absence of this expert domain knowledge requirement also opens policy development to non-robotics-experts, satisfying a need that increases as robots become more commonplace. Furthermore, demonstration has the attractive feature of being an intuitive medium for communication from humans, who already use demonstration to teach other humans. Demonstration also has the practical feature of focusing the dataset to areas of the state–space actually encountered during task execution.

1.2. Problem statement

LfD can be seen as a subset of *Supervised Learning*. In Supervised Learning the agent is presented with labeled training data and learns an approximation to the function which produced the data. Within LfD, this training dataset is composed of example executions of the task by a demonstration teacher (Fig. 1, top).

We formally construct the LfD problem as follows. The world consists of states S and actions A, with the mapping between states by way of actions being defined by a probabilistic transition function $T(s'|s, a) : S \times A \times S \rightarrow [0, 1]$. We assume that the state is not fully observable. The learner instead has access to observed state Z, through the mapping $M : S \rightarrow Z$. A policy $\pi : Z \rightarrow A$ selects actions based on observations of the world state. A single cycle of policy execution at time t is shown in Fig. 1 (bottom).

The set A ranges from containing low-level motions to high-level behaviors. For some simulated world applications, state may be fully transparent, in which case $M = I$, the identity mapping. For all other applications state is not fully transparent and must be observed, for example through sensors in the real world. For succinctness, throughout the text we will use "state" interchangeably with "observed state." It should be assumed, however, that state is always the observed state, unless explicitly noted otherwise. This assumption will be reinforced by use of the Z notation throughout the text.

Throughout the teacher execution, states and selected actions are recorded. We represent a demonstration $d_j \in D$ formally as k_j pairs of observations and actions: $d_j = \{(z_j^i, a_j^i)\}$, $z_j^i \in Z$, $a_j^i \in A$, $i = 0 \cdots k_j$. These demonstrations set LfD apart from other learning approaches. The set D of the demonstrations is made available to the learner. The policy derived from this dataset enables the learner to select an action based on the current state.

1.3. Terminology and context

Before continuing, we pause to place the intents of this survey within the context of previous LfD literature. The aim of this survey is to review the broad topic of LfD, to provide a categorization that highlights differences between approaches, and to identify research areas within LfD that have not yet been explored.

We begin with a comment on terminology. Demonstration-based learning techniques are described by a variety of terms within the published literature, including Learning by Demonstration (LbD), Learning from Demonstration (LfD), Programming by Demonstration (PbD), Learning by Experienced Demonstrations, Assembly Plan from Observation, Learning by Showing, Learning by Watching, Learning from Observation, behavioral cloning, imitation and mimicry. While the definitions for some of these terms, such as imitation, have been loosely borrowed from other sciences, the overall use of these terms is often inconsistent or contradictory across articles.

Within this article, we refer to the general category of algorithms in which a policy is derived based on demonstrated data as *Learning from Demonstration (LfD)*. Within this category, we further distinguish between approaches by their various characteristics, as outlined in Section 2, such as the source of the demonstrations and the learning techniques applied. Subsequent sections introduce terms used to characterize algorithmic differences. Due to the already contradictory use of terms in the existing literature, our definitions will not always agree with those of other publications. Our intent, however, is not for others in the field to adopt the terminology presented here, but rather to provide a consistent set of definitions that highlight distinctions between techniques.

Regarding a categorization for approaches, we note that many legitimate criteria could be used to subdivide LfD research. For example, one proposed categorization considers the broad spectrum of *who, what, when and how to imitate*, or subsets thereof [2,3]. Our review aims to focus on the specifics of implementation. We therefore categorize approaches according to the computational formulations and techniques required to implement an LfD system.

To conclude, readers may also find useful other related surveys of the LfD research area. In particular, the book *Imitation in*

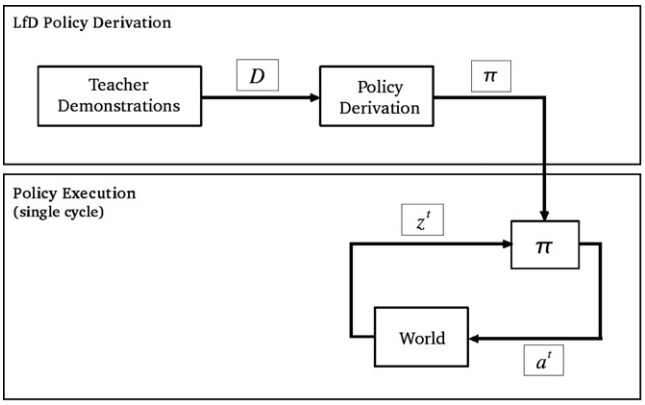

Fig. 1. Control policy derivation and execution.

Animals and Artifacts [4] provides an interdisciplinary overview of research in imitation learning, presenting leading work from neuroscience, psychology and linguistics as well as computer science. A narrower focus is presented in the chapter "Robot Programming by Demonstration" [2] within the book *Handbook of Robotics*. This work particularly highlights techniques which may augment or combine with traditional LfD, such as giving the teacher an active role during learning. By contrast, our focus is to provide a categorical structure for LfD approaches, in addition to presenting the specifics of implementation. We do refer the reader to this chapter for a more comprehensive historical overview of LfD, as the scope of our survey is restricted to recently published literature. Additional reviews that cover specific sub-areas of LfD research in detail are highlighted throughout the article.

2. Design choices

There are certain aspects of LfD which are common among all applications to date. One is the fact that a teacher demonstrates execution of a desired behavior. Another is that the learner is provided with a set of these demonstrations, and from them derives a policy able to reproduce the demonstrated behavior.

However, the developer still faces many design choices when developing a new LfD system. Some of these decisions, such as the choice of a discrete or continuous action representation, may be determined by the domain. Other design choices may be up to the preference of the developer. As we discuss in the later sections, these design decisions strongly influence how the learning problem is structured and solved. In this section we highlight those decisions that are the most significant to make.

To illustrate these choices, we pair the presentation with a running *pick and place* example in which a robot must move a box from a table to a chair. To do so, the object must be (1) picked up, (2) relocated and (3) put down. We present alternate representations and/or learning methods for this task, to illustrate how these particular choices influence task formalization and learning.

2.1. Demonstration approach

Within the context of gathering teacher demonstrations, two key decisions must be made: the choice of *demonstrator*, and the choice of *demonstration technique*. We discuss various choices for each of these decisions below. Note that these decisions are at times affected by factors such as the complexity of the robot and task. For example, teleoperation is rarely used with high degree of freedom humanoids, since their complex motions are typically difficult to control via joystick.

2.1.1. The choice of demonstrator

Most LfD work to date has made use of human demonstrators, although some techniques have also examined the use of robotic teachers, hand-written control policies and simulated planners. The choice of demonstrator further breaks down into the subcategories of (i) who *controls* the demonstration and (ii) who *executes* the demonstration.

For example, consider a robot learning to move a box, as described above. One demonstration approach could have a robotic teacher pick up and relocate the box using its own body. In this case a *robot* teacher controls the demonstration, and its *teacher* body executes the demonstration. An alternate approach could have a human teacher teleoperate the robot learner through the task of picking up and relocating the box. In this case a *human* teacher controls the demonstration, and the *learner* body executes the demonstration. The choice of demonstrator has a significant impact on the type of learning algorithms that can be applied. As we discuss in Section 3, the similarity between the state and action spaces of the teacher and learner determines the kinds of algorithms that may be required to process the data.

2.1.2. Demonstration technique

The choice of demonstration technique refers to the *strategy* for providing data to the learner. One option is to perform *batch learning*, in which case the policy is learned only once all data has been gathered. Alternatively, *interactive approaches* allow the policy to be updated incrementally as training data becomes available, possibly provided in response to current policy performance. Examples of both approaches are highlighted throughout the article.

2.2. Problem space continuity

The question of continuity plays a prominent role within the context of state and action representation, and many valid representations frequently exist for the same domain. Within our robot box moving example, one option could be to discretize state, such that the environment is represented by Boolean features such as *box on table* and *box held by robot*. Alternatively, a continuous state representation could be used in which the state is represented by the 3D positions of the robot's end effector and the box. Similar discrete or continuous representations could be chosen for the robot's actions.

In designing a domain, the continuity of the problem space may be determined by many factors, such as the desired learned behavior, the set of available actions and whether the world is simulated or real. As discussed in Section 4, the question of continuity heavily influences how suitable the various policy derivation techniques are for addressing a given problem.

Additionally, we note that LfD can be applied at a variety of action control levels, depending on the problem formulation. We roughly group actions into three control levels: low-level actions for motion control, basic high-level actions (often called action primitives) and complex behavioral actions for high-level control. Note that this is a somewhat different consideration to action–space continuity. For example, a low-level motion could be formulated as discrete or continuous, and so this action level can map to either space. As a general technique, LfD can be applied at any of these action levels. Most important in the context of policy derivation, however, is whether actions are continuous or discrete, and not their control level. For the remainder of the article, we distinguish between representations based on continuity only.

2.3. Policy derivation and performance

In selecting an algorithm for generating a policy, we consider two key decisions: the general technique used to *derive the policy*, and whether performance can *improve beyond the teacher's demonstrations*. These decisions are influenced by action-continuity, as described in the previous section, which is in turn determined both by task and robot capabilities.

2.3.1. Policy derivation technique

As summarized in Fig. 2, research within LfD has seen the development of three core approaches to policy derivation from demonstration data, which we define as *mapping function*, *system model*, and *plans*:

- *Mapping function* (Section 4.1): Demonstration data is used to directly approximate the underlying function mapping from the robot's state observations to actions ($f() : Z \rightarrow A$).
- *System model* (Section 4.2): Demonstration data is used to determine a model of the world dynamics ($T(s'|s, a)$), and possibly a reward function ($R(s)$). A policy is then derived using this information.

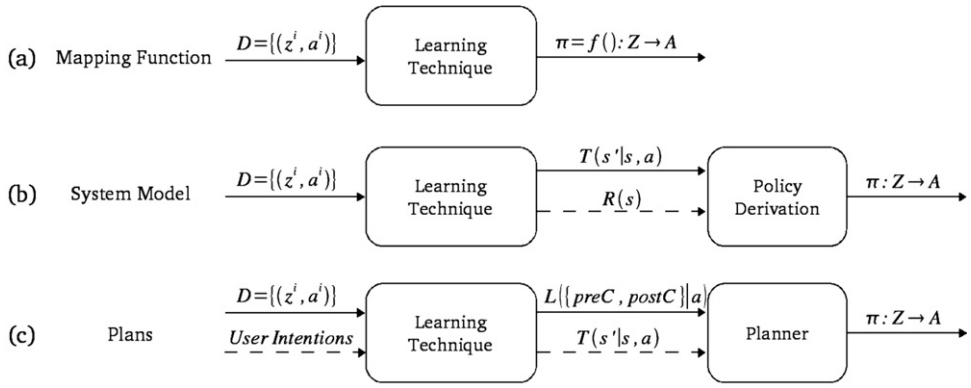

Fig. 2. Policy derivation using the generalization approach of determining (a) an approximation to the state→ action mapping function, (b) a dynamics model of the system and (c) a plan of sequenced actions.

- *Plans* (Section 4.3) : Demonstration data, and often additional user intention information, is used to learn rules that associate a set of pre- and post-conditions with each action ($L(\{preC, postC\}|a)$), and possibly a sparsified state dynamics model ($T(s'|s, a)$). A sequence of actions is then planned using this information.

Returning again to our example, suppose a mapping function approach is used to derive the policy. A function $f() : Z \rightarrow A$ is learned that maps the observed state of the world, for example the 3D location of the robot's end effector, to an action which guides the learner towards the goal state, for example the desired end effector motor speed. Consider instead using a system model approach. Here a state transition model $T(s'|s, a)$ is learned, for example that taking the *pick up* action when in state *box on table* results in state *box held by robot*. Using this model, the derived policy indicates the best action to take when in a given state, to guide the robot towards the goal state. Finally, consider using a planning approach. The pre- and post-conditions of executing an action $L(\{preC, postC\}|a)$ are learned from the demonstrations. For example, the *pick up* action requires the *box on table* pre-condition, and results in the *box held by robot* post-condition. A planner uses this learned information to produce a sequence of actions that ends with the robot in the goal state. Each of the three approaches are discussed in detail within Section 4.

2.3.2. Dataset limitations

Training examples obtained from demonstration are inherently limited by the performance of the teacher. In many domains, it is possible that this teacher performance is suboptimal when compared with the abilities of the learner. For example, a human teacher may not be physically able to execute actions as quickly or accurately as a robot. Since the learner derives its policy from these examples, the performance of this policy is therefore also limited by the teacher's abilities. Many LfD learning systems, however, have been augmented to enable learner performance to improve beyond what was provided in the demonstration dataset. Examples include the incorporation of teacher advice or Reinforcement Learning techniques. These approaches are discussed in depth within Section 5.

3. Gathering examples: How the dataset is built

In this section, we discuss various techniques for executing and recording demonstrations. The LfD dataset is composed of state–action pairs recorded during teacher executions of the desired behavior. Exactly *how* they are recorded, and *what* the teacher uses as a platform for the execution, varies greatly across approaches. Examples range from sensors on the robot learner recording its own actions as it is passively teleoperated by the teacher, to a camera recording a human teacher as she executes the behavior with her own body.

For LfD to be successful, the states and actions in the learning dataset must be usable by the student. In the most straightforward setup, the states and actions of the teacher executions map directly to the learner. In reality, however, a direct mapping will often not be possible, as the learner and teacher will likely differ in sensing or mechanics. For example, a robot learner's camera will not detect state changes in the same manner as a human teacher's eyes, nor will its gripper apply force in the same manner as a human hand. The challenges which arise from these differences are referred to broadly as *Correspondence Issues* [5].

3.1. Correspondence

The issue of correspondence deals with the identification of a mapping between the teacher and the learner that allows the transfer of information from one to the other. In this survey, we define correspondence with respect to two mappings, shown in Fig. 3: the *record mapping*, and the *embodiment mapping*.

- The *Record Mapping* (Teacher Execution → Recorded Execution) refers to whether the exact states/actions experienced by the teacher during the demonstration execution are recorded within the dataset.
- The *Embodiment Mapping* (Recorded Execution → Learner) refers to whether the states/actions recorded within the dataset are exactly those that the learner would observe/execute.

When the *record mapping* is the identity $I(z, a)$, the states/actions experienced by the teacher during execution are directly recorded in the dataset. Otherwise this teacher information is encoded according to some record mapping function $g_R(z, a) \neq I(z, a)$, and this encoded information is recorded within the dataset. Similarly, when the *embodiment mapping* is the identity $I(z, a)$, the states/actions in the dataset map directly to the learner. Otherwise the embodiment mapping consists of some function $g_E(z, a) \neq I(z, a)$. For any given learning system, it is possible to have neither, either or both of the record and embodiment mappings be the identity. Note that the mappings do not change the content of the demonstration data, but only the reference frame within which it is represented. Fig. 4 shows the intersection of these configurations, which we discuss further within subsequent sections.

The embodiment mapping is particularly important when considering real robots, compared with simulated agents. Since actual robots execute real actions within a physical environment, providing them with a demonstration involves a physical execution by the teacher. Learning within this setting depends heavily upon an

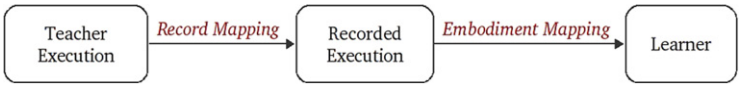

Fig. 3. Mapping a teacher execution to the learner.

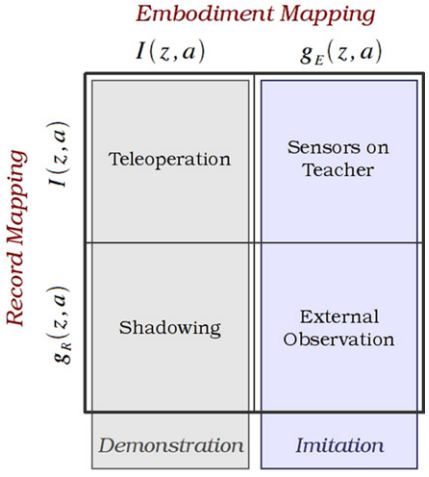

Fig. 4. Intersection of the record and embodiment mappings. The left and right columns represent an identity (Demonstration) and non-identity (Imitation) embodiment mapping, respectively. Each column is then subdivided by an identity (top) or non-identity (bottom) record mapping. Typical approaches to providing data are listed within the quadrants.

accurate mapping between the recorded dataset and the learner's abilities.

Recalling again our box relocation example, consider a human teacher using her own body to demonstrate moving the box, and that a camera records the demonstration. Let the teacher actions, A_T, be represented as human joint angles, and the learner actions, A_L, be represented as robot joint angles. In this context, the robot observes the teacher's demonstration of the task through the camera images. The teacher's exact actions are unknown to the robot; instead, this information must be extracted from the image data. This is an example of a $g_R(z,a) \neq I(z,a)$ *record* mapping $A_T \rightarrow D$. Furthermore, the physical embodiment of the teacher is different from that of the robot and his actions (A_T) are therefore *not* the same as those of the robot (A_L). Therefore in order to make the demonstration data meaningful for the robot, a mapping $D \rightarrow A_L$ must be applied to convert the demonstration into the robot's frame of reference. This is one example of a $g_E(z,a) \neq I(z,a)$ *embodiment* mapping.

The categorization of LfD data sources that we present in this article groups approaches according to the absence or presence of the record and embodiment mappings. We select this categorization to highlight the levels at which correspondence plays a role in demonstration learning. Within a given learning approach, the inclusion of each additional mapping introduces a potential injection point for correspondence difficulties; in short, the more mappings, the more difficult it is to recognize and reproduce the teacher's behavior. However, mappings also reduce constraints on the teacher and increase the generality of the demonstration technique.

In our categorization, we first split LfD data acquisition approaches into two categories based on the embodiment mapping, and thus by execution platform:

- *Demonstration*: There is no embodiment mapping, because demonstration is performed on the actual robot learner (or a physically identical platform). Thus $g_E(z,a) \equiv I(z,a)$.
- *Imitation*: There exists an embodiment mapping, because demonstration is performed on a platform which is *not* the robot learner (or a not physically identical platform). Thus $g_E(z,a) \neq I(z,a)$.

We then further distinguish approaches within each of these categories according to record mapping, relating to how the demonstration is recorded. Fig. 5 introduces our full categorization of the various approaches for building the demonstration dataset. We structure our discussion of data acquisition in subsequent sections according to this categorization.

3.2. Demonstration

When teacher executions are *demonstrated*, by our definition there exists no embodiment mapping issue between the teacher and learner. This situation is presented in the left column of Fig. 4. There may exist a non-direct record mapping, however, for state and/or actions, if the states experienced (actions taken) by the demonstrator are not recorded directly, and must instead be inferred from the data. Based on this distinction, we identify two common approaches for providing demonstration data to the robot learner as:

- *Teleoperation* (Section 3.2.1): A demonstration technique in which the teacher *operates* the robot learner platform and the robot's sensors record the execution. The record mapping is direct; thus $g_R(z,a) \equiv I(z,a)$.
- *Shadowing* (Section 3.2.2): A demonstration technique in which the robot learner records the execution using its own sensors while attempting to match or *mimic* the teacher motion as the teacher executes the task. There exists a non-direct record mapping; thus $g_R(z,a) \neq I(z,a)$.

Again, for both teleoperation and shadowing the robot records from its own sensors as its body executes the behavior, and so the embodiment mapping is direct, $g_E(z,a) \equiv I(z,a)$.

The record mapping distinction plays an important role in the application and development of demonstration algorithms. As described below, teleoperation is not suitable for all learning platforms, while shadowing techniques require an additional processing component to enable the learner to mimic the teacher. In the following subsections we discuss various works that utilize these demonstration techniques.

3.2.1. Teleoperation

During teleoperation, a robot is operated by the teacher while recording from its own sensors. Since the robot directly records the states/actions experienced during the execution, the record mapping is direct and $g_R(z,a) \equiv I(z,a)$. Teleoperation provides the most direct method for information transfer within demonstration learning. However, teleoperation requires that operating the robot be manageable, and as a result not all systems are suitable for this technique. For example low-level motion demonstrations are difficult on systems with complex motor control, such as high degree of freedom humanoids.

Demonstrations recorded through human teleoperation via a joystick are used in a variety of applications, including flying a robotic helicopter [6], robot kicking motions [7], object grasping [8,9], robotic arm assembly tasks [10] and obstacle avoidance and navigation [11,12]. Teleoperation is also applied to a wide variety

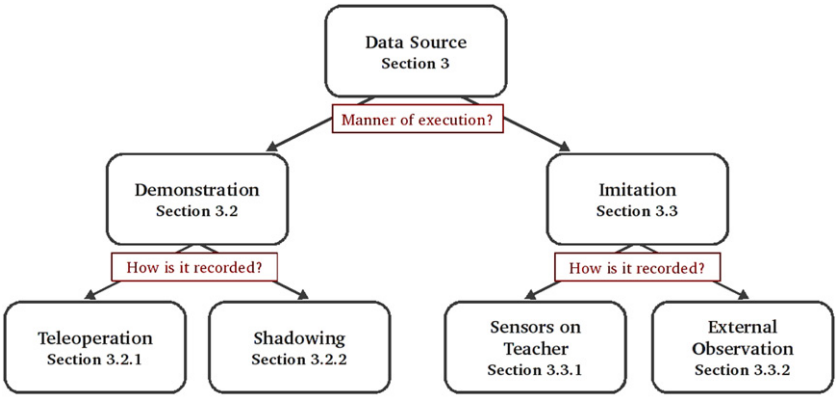

Fig. 5. Categorization of approaches to building the demonstration dataset.

of simulated domains, ranging from static mazes [13,14] to dynamic driving [15,16] and soccer domains [17], and many other applications.

Human teachers also employ techniques other than direct joysticking. In kinesthetic teaching, a humanoid robot is not actively controlled but rather its passive joints are moved through desired motions [18]. Demonstration may also be performed through speech dialog, in which the teacher specifically tells the robot what actions to execute in various states [19–21]. Both of these techniques might be viewed as variants on traditional teleoperation, or alternatively as a sort of high-level teleoperation. In place of a human teacher, hand-written controllers are also used to teleoperate robots [22–24,12].

We note that demonstration data recorded using real robots frequently does not represent the *full* observation state of the teacher. This occurs if, while executing, the teacher employs extra sensors that are not recorded. For example, if the teacher observes parts of the world that are inaccessible from the robot's cameras (e.g., behind the robot, if its cameras are forward-facing), then state, as observed by the teacher, differs from what is actually recorded as data. A small number of works do address this problem. For example, in Grudic and Lawrence [25] a vision-based robot is teleoperated while the teacher looks exclusively at a screen displaying the robot's camera output.

3.2.2. Shadowing

During shadowing, the robot platform mimics the teacher's demonstrated motions while recording from its own sensors. The record mapping is not direct, $g_R(z, a) \neq I(z, a)$, because the states/actions of the true demonstration execution are not recorded. Rather, the learner records its own mimicking execution, and so the teacher's states/actions are *indirectly* encoded within the dataset. In comparison to teleoperation, shadowing requires an extra algorithmic component which enables the robot to track and actively shadow (rather than passively be teleoperated by) the teacher.

Navigational tasks learned from shadowing have a robot follow an identical-platform robot teacher through a maze [26], follow a human teacher past sequences of colored markers [27] and mimic routes determined from observations of human teacher executions [28]. A humanoid learns arm gestures, as well as the rules of a turn-taking gesture game, by mimicking the motions of a human demonstrator [29].

3.3. Imitation

As previously defined, embodiment issues do exist between the teacher and learner for *imitation* approaches. This situation is presented in the right column of Fig. 4, indicating the presence of a non-identity embodiment mapping $g_E(z, a) \neq I(z, a)$. Within this setting, we further divide approaches for providing imitation data by whether the record mapping is the identity or not:

- *Sensors on teacher* (Section 3.3.1): An imitation technique in which sensors located on the executing body are used to record the teacher execution. The record mapping is direct; thus $g_R(z, a) \equiv I(z, a)$.
- *External observation* (Section 3.3.2): An imitation technique in which sensors external to the executing body are used to record the execution. These sensors may or may not be located on the robot learner. There exists a non-direct record mapping; thus $g_R(z, a) \neq I(z, a)$.

The record mapping distinction again plays an important role in the application and development of imitation algorithms, just as it did with demonstration approaches. The sensors on teacher approach provides precise measurements, but requires a lot of overhead in the way of specialized sensors and customized surroundings. By contrast, external observation is a more general method, but also provides less reliable measurements. In the following subsections, we discuss these imitation techniques and the various works that utilize them. Additionally, we point the reader to a review that examines the problems of correspondence and knowing what to imitate [30].

3.3.1. Sensors on teacher

The sensors-on-teacher approach utilizes recording sensors located *directly on* the executing platform. This means no record mapping, and so $g_R(z, a) \equiv I(z, a)$, which alleviates one potential source for correspondence difficulties. The strength of this technique is that the teacher provides precise measurements of the example execution. However, the overhead attached to the specialized sensors, such as human-wearable sensor-suits, or customized surroundings, such as rooms outfitted with cameras, is non-trivial and limits the applicability settings of this technique.

Human teachers commonly use their own bodies to perform example executions by wearing sensors able to record the person's state and actions. This is especially true when working with humanoid or anthropomorphic robots, since the body of the robot resembles that of a human. The recorded joint angles of a human teach drumming patterns to a 30-DoF humanoid [31], and in later work walking patterns as well [32]. A humanoid learns referee signals by pairing kinesthetic teachings (3.2.1) with human teacher executions recorded via wearable motion sensors [33]. Another approach has a human wearing sensors control a simulated human, which maps to a simulated robot and then to a real robot arm [34].

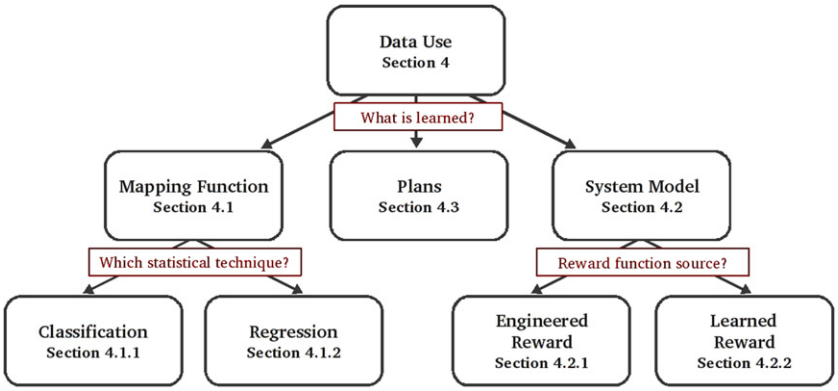

Fig. 6. Categorization of approaches to learning a policy from demonstration data.

3.3.2. External observation

Imitation performed through external observation relies on data recorded by sensors located *externally to* the executing platform, meaning that a record mapping exists, and $g_R(z, a) \neq I(z, a)$. Since the robot does not directly record the actual states/actions experienced by the teacher during the execution, they must be inferred, introducing a new source of uncertainty for the learner. Some LfD implementations first extract the teacher states/actions from this recorded data, and then map the extracted states/actions to the learner. Others map the recorded data directly to the learner, without ever explicitly extracting the states/actions of the teacher. Compared to the sensors-on-teacher approach, the data recorded under this technique is less precise and less reliable. The method, however, is more general and is not limited by the overhead of specialized sensors and settings.

An early seminal work has a robotic arm learn pole balancing via stereo-vision and human demonstration [35]. Unlike most other approaches in this section, human demonstration trains a non-anthropomorphic robot and data is obtained by tracking the movement of the pole, not the teacher's arm itself. Similarly, the position of the human player's (teacher's) puck is tracked when teaching a 37-DoF humanoid to play air hockey [36]. The robot learner itself directly observes the teacher executions, as in other humanoid systems, where cameras are often located directly on the robot body as a parallel to human eyes.

Typically, the external sensors used to record human teacher executions are vision-based. Motion capture systems utilizing visual markers are applied to teaching human motion [37–39] and manipulation tasks [40]. Visual features are tracked in biologically-inspired frameworks that link perception to abstract knowledge representation [41] and teach an anthropomorphic hand to play the game Rock–Paper–Scissors [42]. Background subtraction is used to extract teacher motion from images [29]. This work takes a unique approach to building the embodiment mapping $g_E(z, a)$; the correspondence is *learned*, rather than being hand-engineered, by having the human teacher shadow the robot's motions.

A number of systems combine external sensors with other information sources; in particular, with sensors located directly on the teacher. A force-sensing glove is combined with vision-based motion tracking to teach grasping movements [43,44]. Movement, end effector position, stereo vision, and tactile information teaches generalized dexterous motor skills to a humanoid robot [45]. The combination of 3D marker data with torso movement and joint angle data is used for applications on a variety of simulated and robot humanoid platforms [46]. Other approaches combine speech with visual observation of human gestures [47] and object tracking [48], within the larger goal of speech-supported LfD learning.

Several works also explore learning through external observation of non-human teachers. A robot learns a manipulation task through visual observation of an identical robotic teacher [49], and a simulated agent learns about its own capabilities and unvisited parts of the state space through observation of other simulated agents [50]. A generic framework for solving the correspondence problem between differently embodied robots allows a robotic agent to learn new behaviors through imitation of another, possibly physically different, agent [51].

3.4. Other approaches

Within LfD there do exist exceptions to the data source categorization we have so far presented. These exceptions record only states during demonstration, without recording actions. For example, by drawing a path through a 2D representation of the physical world, high-level path-planning demonstrations are provided to a rugged outdoor robot [52] and a small quadruped robot [53,54]. Since the dataset does not provide actions, no state–action mapping is learned for action selection. Instead, actions are selected at runtime by employing low-level motion planners and controllers [52,53], or by providing transition models $T(s'|s, a)$ [54].

4. Deriving a policy: The source of the state to action mapping

Given a dataset of state–action examples that have been acquired using one of the methods described in the previous section, we now discuss methods for deriving a policy using this data. LfD has seen the development of three core approaches to deriving policies from demonstration data, as summarized in Fig. 2. Learning a policy can involve simply learning an approximation to the state-action mapping (*mapping function*), or learning a model of the world dynamics and deriving a policy from this information (*system model*). Alternately, a sequence of actions can be produced by a planner after learning a model of action pre- and post-conditions (*plans*). Across all of these learning techniques, minimal parameter tuning and fast learning times requiring few training examples are desirable.

We introduce a full categorization of the various approaches to deriving a policy from the demonstration dataset in Fig. 6. Approaches are initially split between the three core derivation techniques described above. Further splits, if present, are approach-specific. We discuss each of these categories in depth in the following sections. Additionally, we direct the reader to a literature review that focuses on statistical and mathematical approaches to deriving motor control policies [3].

4.1. Mapping function

The *mapping function* approach to policy learning calculates a function that approximates the state to action mapping, $f() : Z \rightarrow A$, for the demonstrated behavior (Fig. 2a). The goal of this type of algorithm is to reproduce the underlying teacher policy, which is unknown, and to generalize over the set of available training examples such that valid solutions are also acquired for similar states that may not have been encountered during demonstration.

The details of function approximation are influenced by many factors. These include whether the state input and action output are continuous or discrete, whether the generalization technique uses the data to approximate a function prior to execution time or directly at execution time, whether it is feasible or desirable to keep the entire demonstration dataset around throughout learning, and whether the algorithm updates online.

In general, mapping approximation techniques fall into two categories depending on whether the prediction output of the algorithm is discrete or continuous. *Classification* techniques produce discrete output (Section 4.1.1), and *regression* techniques produce continuous output (Section 4.1.2). Many techniques for performing classification and regression have been developed outside of LfD and we refer the reader to [55] for a full discussion.

4.1.1. Classification

Classification approaches categorize their input into discrete classes, thereby grouping similar input values together. In the context of policy learning, the input to the classifier is robot states and the discrete output classes are robot actions. Below, we present a summary of classification methods applied at three action control levels (basic motion control, action primitives, complex behaviors), as defined in Section 2.2. Although a single classification algorithm can be applied at any level, we distinguish between them to highlight the generality of this learning technique.

Low-level robot actions include basic commands such as moving forward or turning. Example applications that learn a mapping from states to low-level actions include controlling a car within a simulated driving domain using *Gaussian Mixture Models (GMMs)* [56], flying a simulated airplane using *decision trees* [57] and learning obstacle avoidance and navigation behaviors using *Bayesian network* [11] and *k-Nearest Neighbors (kNN)* [58] classifiers.

When states are mapped to motion primitives, the primitives typically are then composed or sequenced together. For example, Pook and Ballard [8] classify primitive membership using kNN, and then recognize each primitive within the larger demonstrated task via *Hidden Markov Models (HMMs)*, to teach a robotic hand and arm an egg flipping manipulation task. HMMs and teach a basic assembly task [59], and motor-skill tasks by identifying and generalizing upon the intentions of the user [60]. A biologically-inspired framework automatically extracts primitives from demonstration data, classifies data via *vector-quantization* and then composes and/or sequences primitives within a hierarchical Neural Network [46]. The work is applied to a variety of test beds, including a 20 DoF simulated humanoid torso, a 37 DoF avatar (a simulated humanoid which responds to external sensors), Sony AIBO dogs and small differential drive Pioneer robots. Under this framework, the simulated humanoid torso is taught a variety of dance, aerobics and athletics motions [61], the avatar reaching patterns [38] and the Pioneer sequenced location visiting tasks [62].

Similar approaches have been used for the classification of high-level behaviors. The behaviors themselves are generally developed (by hand or learned) prior to task learning. A similarity measure on an eigenvector representation recognizes gesture behaviors for an anthropomorphic hand [44], and within this framework HMMs classify demonstrations into gestures for a box sorting task with a Pioneer robot [63]. The Bayesian likelihood method selects actions for a humanoid robot in a button pressing task [64], and *Support Vector Machines (SVMs)* classify behaviors for a robotic ball sorting task [65].

4.1.2. Regression

Regression approaches map demonstration states to continuous action spaces. Similar to classification, the input to the regressor are robot states, and the *continuous* output are robot actions. Since the continuous-valued output often results from combining multiple demonstration set actions, typically regression approaches aply to low-level motions and not high-level behaviors. A key distinction between methods is whether the mapping function approximation occurs at run time, or prior to run time. Below we present a summary of regression techniques for LfD along a continuum between these two extremes.

At one extreme lies Lazy Learning [66], where function approximation does not occur until a current observation point in need of mapping is present. The simplest Lazy Learning technique is kNN, applied to action selection within robotic marble-maze [67] and simulated ball interception [22] domains. More complex approaches include *Locally Weighted Regression (LWR)* [68]. One LWR technique further anchors local functions to the phase of nonlinear oscillators [69] to produce rhythmic movements, specifically drumming [70] and walking [32] patterns with a humanoid robot. While Lazy Learning approaches are fast and expend no effort approximating the function in areas unvisited during execution time, they do require keeping around all of the training data.

In the middle of the data processing continuum lie techniques in which the original data is converted to another, possibly sparsified, representation prior to run time. This *converted* data is then used by Lazy Learning techniques at run time. For example, *Receptive Field Weighted Regression (RFWR)* [71] first converts demonstration data to a Gaussian and local linear model representation. *Locally Weighted Projection Regression (LWPR)* [72] extends this approach to scale with input data dimensionality and redundancy. Both RFWR and LWPR are able to incrementally update the number of representative Gaussians as well as regression parameters online. Successful robotics applications using LWPR include an AIBO robot performing basic soccer skills [23] and a humanoid playing the game of air hockey [67]. This latter work actually employs multiple techniques, first using kNN to select a behavior class and then LWPR to generate low-level actions using the demonstration data within this class. The approaches at this position on the continuum benefit from not needing to evaluate all of the training data at run time, but at the cost of extra computation and generalization prior to execution.

At the opposite extreme lie approaches which form a complete function approximation prior to execution time. At run time, they no longer depend on the presence of the underlying data (or any modified representations thereof). A seminal work uses a *Neural Network (NN)* to enable autonomous driving of a van at speed on a variety of road types [73]. NN techniques also enable a robot arm to perform a peg in hole task [49], and a set of b-spline wavelets approximate time-sequenced data for full body humanoid motion [39]. Also possible are statistical approaches that represent the demonstration data in a single or mixture distribution sampled at run time. Demonstration data encoded as a joint distribution over objects, hand position and hand orientation is used by a humanoid for grasping novel and known household objects [47]. *Gaussian Mixture Regression (GMR)* teaches a humanoid robot basketball referee signals [33], and *Sparse On-Line Gaussian Processes (SOGP)* teaches an AIBO robot to perform basic soccer skills [74]. The regression techniques in this area all have the advantage of *no* training data evaluations at run time,