2.5. Comparision

In this section, a comparison of the different learning approaches, described earlier in this chapter, will be made. This comparison is shown in table 2.5. Here, the different robot teaching methods are compared based on multiple criteria. This comparison is a summary of information found in literature and made assumptions based on this. The literature used for this, will be described in the remaining of this section. In order to have a more reliable comparison, further research should be conducted.

Table 2.5: Comparison of robot teaching methods. It should be noted that this comparison is based on an interpretation of literature. In order to have a more conclusive analysis, research is required. The methods are compared based on some criteria, here + indicates a high score relating to the criteria, +/- an average score, and - a bad score. These score were given with respect to another. From left to right the different criteria are; *Computational efficiency* describing the amount of necessary to learn. *Stability* describes the ability to deal with perturbation. Smoothness refers to the smoothness of the trajectory. Generalizability describes the ability of the system to deal with situation not occurred during the learning process (e.g. new starting point or goal). The difficulty of implementing a certain system is indicated by *implementation*. *High-dimensional input* states whether a system can deal with large amount of data. Lastly, *online* describes whether the system were originally implemented for online usage.

	Method	Comp. efficiency	Stability	Smoothness	Generalizability	Implementation	High-dim. input	Online
IL	BC	+	+/-	+	+	+	+/-	-
	IRL	+/-	+/-	+	+	+/-	+	+
RL	Value function	+/-	- [1]	+/-	-	-	-	+
	Policy Search	-	-	+/-	-	-	+	+
	Actor-Critic	-	+/-	+/-	+/-	-	+	+

[1] *No information about the stability of value function method could be found, so this is an assumption based on the information of the other RL methods.*

The *computational efficiency* describes the amount of data and computational time, required to converge to a solution [10]. Especially for real-time application, this is of the highest importance as a high computational cost would result in a slower system, thus making this system less reactive. Slow learning might not necessarily be a problem when learning in a ”save” environment, it could be a problem when adaptation to the environment is required. IL method often have a higher computational efficiency than RL. This is due to the fact that fewer iterations are needed to obtain the policy. RL on the other hand, use a trial-and-error approach, resulting in a much more iteration. Looking at the different IL method, BC often only have to do one iteration per demonstration. There is, however, a difference in computational time between the different BC method. So, does DMP just have to update one parameter. Whereas, GP has to compute a new covariance matrix (which can be large for high-dimensional inputs) and has to invert this. As IRL typically have to update their cost function multiple times, their sample efficiency is slightly worse than those of the BC. Looking at the RL methods, value function are known to have a high sample efficiency in comparison to the PS method [6]. This could be translated in the former having a higher computational efficiency than the latter. As actor-critic methods uses the bootstrapped value-function to reduce the variance of the gradient estimate, they gain the sample efficiency of the value function.

Stability refers to the ability of the system to deal with perturbations. In cases where perturbation would result in big difference in the behaviour, the system said to be unstable. Instead, it is desired that the behaviour only changes a small fraction or maybe not even at all. This is desired to ensure a safe design, which is of importance to avoid a robot damaging itself, and something or someone else. The stability of the BC methods is not always ensured. So, is DMP statically asymptotically stable [78], meaning it has a stable attraction to a target position. Whereas ProMP [12], GMR and GPR [79] do not provide this guarantee. As RL methods generally search a large part of the solution space to find a policy, it is assumed that they generally have a good stability. However, this is not necessary the case as most method depend on multiple conditions. So does PG require the setting of a learning rate,

which, if chosen incorrectly, results in unstable behaviour [73]. This instability problem was, however, tackled by formulating PS as an inference problem with latent variables and, using the EM algorithm, to determine a new policy. The Inf.Th. principle states that in order to provide a stable behaviour, the distance between the old and new trajectory distribution, should be bounded. A method which has been identified to increase stability is the usage of replay [71]. During replay, the robot is initiated in states which are not stored, in order to improve exploration efficiency. This also has the effect of becoming more stable. As the actor-critic algorithm A3C allows for multiple agents to asynchronously explore different policies in parallel, the usage of replay was not require to enesure stability.

With *smoothness*, the smoothness of the encoded behaviour (often trajectory) is meant. To minimize the risk of damage to the robot, it is desired that the behaviour is smooth, without sharp changes [10]. This criterion is hard to quantify as it has to be visually examined. Therefore, results become quite subjective. For the IL approaches, the different smoothness was based on literature. For each approach, multiple papers were used (for BC [80, 31, 81, 37, 41, 50] and for IRL [61, 82]) to determine the smoothness of the method. As IL methods try to imitate the expert, which is assumed to have a smooth behaviour, the resulting behaviour of the robot is in most cases also smooth. The same could not be said for RL method. Here, the smoothness of the system highly depends on the chosen reward function. Just as for the stability, the usage of a bad trajectory would result in a rough behaviour.

Generalizability is the ability of a system to perform well in similar environments as used for the training of the system [59]. Examples of this are states and action unobserved in the demonstration, starting the task at a different initial state, and reusing skills in different problem settings [6]. BC method often generalize quite well. However, in some cases some extensions are required [12]. An example is ProMP which learn the distribution of demonstrated trajectories in parameter space. To generalize to a new start and goal position, the learned distribution has to be conditioned. An example of this was given in [81]. Research has also proven IRL method to generalize well to un-encountered settings [62]. Generally speaking, RL often has a lack of generalizability [71]. A solution could be to increase the sample dataset and train on multiple (random) environments [83]. This, however, comes at the cost of computational efficiency. A difference can also be seen between model-based and model-free approaches. Where the former often generalizes better compared to the latter [71]. There have been PS algorithm which generalize relatively well. Most of these are episodic-based policy evaluation strategies [73]. It should be noted that research on generalizability for RL, still seems to be limited.

It is desired that a system is easy to implement. Difficult *implementation* can mean that a lot of time needs to be spent to implement it, and in addition, that more can go wrong implementing it. For example, as more assumptions have to be made. Overall, BC methods are quite easy to implement. The reason for this is that most methods are model-free [12]. In contrast, IRL methods are often model-based, meaning that if no model of the system is available, implementation can become a lot more difficult. The implementation of RL the method is often more difficult than implementing IL. Besides the difficulty of determining a reward function, it is also quite costly to do multiple iterations [66]. Therefore, the solution of running the system first in solution has been used. This can however not completely replace the real-time learning, as even small differences in the environment can result in totally different behaviours.

The ability of BC method to deal with high-dimensional inputs depends on the algorithm. Both DMP and ProMP are often not able to do so, whereas GMM, GPR and HMM are [39]. As RL often uses a continuous stream of data, the amount of data becomes large. This makes it often quite difficult for RL method to deal with robotic learning. Nevertheless, PS has proven to be able to scale well with high-dimensional state space [6]. The same cannot be said for value function methods.

The last criteria relate to *online* learning. For each method, online implementations could often be found. However, these implementations often requires some adaptation to the original method. Therefore, this criterium states whether methods were originally meant for online implementations. BC method usually obtained the demonstrations first, after which a model was computed. Recent can be found for online implementation of this method [2, 29], but as stated they require some adaptations of the original method. The principle of IRL is to use the demonstration to find a reward function, and use this to compute a policy (using some RL method) [59]. As the updating of the policy is done iteratively, this method can also be stated as an online learning process. RL method have always been used in an online manner. As the policy is iteratively updated based on the sensory feedback [66]. In contrast to

the IL approaches, is offline usage of RL not common [71].