VeriDream Deliverable 5

Coordination with the AI4EU platform

This project has received funding from the
European Union's Horizon 2020 research and innovation
programme under Grant Agreement No. 951992

Project	
Grant number	951992
Acronym	VeriDream
Deliverable	
Number	5
Rel. Number	5.2
Workpackage	5 – Dissemination
Nature	O
Dissemination	PU
Due date	31.03.2021
Lead	GoodAI
Contributers	GoodAI, Sorbonne, ARMINES, DLR

Document History

Version	Date	Partner	Description
0.1	22.02.2021	DLR	Creation
0.2	24.02.2021	DLR, ARMINES	Added SRL Toolbox
0.3	24.02.2021	DLR	Added stable-baselines 2 and 3
0.4	26.02.2021	Sorbonne	Added Quality Diversity
0.5	02.03.2021	DLR	Added Augmented Autoencoder
0.6	10.03.2021	GoodAI	Added links to AI4EU platform
1.0	10.03.2021	DLR	Final version

Overview of the deliverable

In this deliverable, we describe the VeriDream AI software packages that have been published on AI4EU, the EU on-demand platform for AI. This deliverable provides brief descriptions of the algorithms, as required for the AI4EU platform. In the deliverable "D2.1: Guidelines for using DREAM methods", more detailed descriptions of the methods will be provided, including their input/output formats and limitations. The application of these methods to the industrial use cases will be described in the deliverables "D1.1: Use Case Requirements and Matchmaking Results" and "D2.2: Closing the innovation loop".

Augmented Autoencoder

The Augmented Autoencoder (AAE) pipeline estimates a 6D pose of an object purely trained on synthetic data. In order to estimate the pose of an object a bounding box detector first has to detect and classify the object. The resulting region of interest is then forwarded to the AAE, which predicts the 3D orientation of the corresponding object. To do so, the AAE generates a latent representation of the object appearance, which can be translated to a 3D orientation via a codebook. Based on this rotation information the translation can be computed via the projective distance. The resulting pose can be either directly used or further refined by a Iterative Closest Point (ICP) approach.

Original developer(s)	Martin Sundermeyer (DLR)
Current version	0.0.1
License	MIT
Link to AI4EU resource	`https://www.ai4eu.eu/resource/augmented-autoencoder-0`

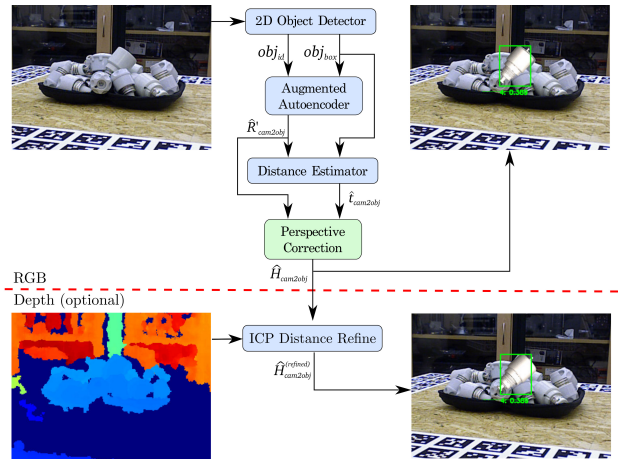

Figure 1: Overview of the processing pipeline in the Augmented Autoencoder software.

Quality Diversity

While reinforcement learning algorithms converge towards a single policy, it may be useful to generate multiple policies instead of just one. This diversity is an indication of what behaviors are within reach. It also helps to cross the reality gap as some policies may transfer better than others and finally the repertoire of generated policies can be used as a set of primitive actions by an upper level policy or planning algorithm. Diversity algorithms like Novelty Search try to cover a "behavior space". The behavior space is, in general, provided and indicates the space that is worth exploring. It is a projection of the robot trajectory in a smaller dimension space. Quality Diversity algorithms also take into account and thus generate set of solutions that are both diverse and efficient according to a given quality measure. It should be noted that the quality is, in general considered at a local level: the comparison between solutions is done between solutions that are similar, i.e. that are close in the behavior space.

Original developer(s)	Alexandre Coninx, Stéphane Doncieux (Sorbonne University)
Current version	GECCO2020
License	CeCILL v2.1
Link to AI4EU resource	`https://www.ai4eu.eu/resource/diversity-algorithms`

State Representation Learning: S-RL Toolbox

S-RL Toolbox: Reinforcement Learning (RL) and State Representation Learning (SRL) Toolbox for Robotics

State representation learning aims at learning compact representations from raw observations in robotics and control applications. The goal of such a representation is mainly to improve sample efficiency of reinforcement learning algorithms by introducing priors on the relevant information for control, but it is also relevant for other control approaches such as model predictive control or planning. Approaches used for this objective fall in four main categories: auto-encoders, learning forward models, inverse dynamics or learning using generic priors on the state characteristics. However, the diversity in applications and methods makes the field lack standard evaluation datasets, metrics and tasks. This toolbox therefore provides a set of environments, data generators, robotic control tasks, metrics and tools to facilitate iterative state representation learning, the evaluation of the learned representations in reinforcement learning settings and the visual analysis of their behavior.

Original developer(s)	Antonin Raffin (ARMINES)
Current version	1.2.0
License	MIT
Link to AI4EU resource	`https://www.ai4eu.eu/resource/s-rl-toolbox-reinforcement-learning-rl-and-state-representation-learning-srl-toolbox`

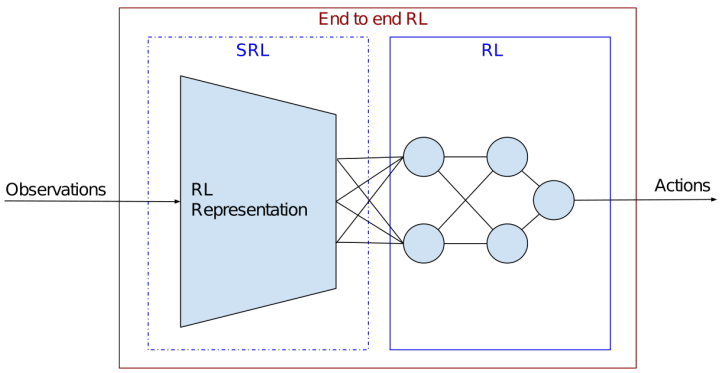

Figure 2: Illustration of the SRL Toolbox. Its main aim is to separate state representation learning from reinforcement learning.

Stable Baselines 2 (SB2)

The Stable Baselines algorithms will make it easier for the research community and industry to replicate, refine, and identify new ideas, and will create good baselines to build projects on top of. We expect these tools will be used as a base around which new ideas can be added, and as a tool for comparing a new approach against existing ones. We also hope that the simplicity of these tools will allow beginners to experiment with a more advanced toolset, without being buried in implementation details.

Original developer(s)	Antonin Raffin (ARMINES)
Current version	2.10.1
License	MIT
Link to AI4EU resource	`https://www.ai4eu.eu/resource/stable-baselines-2-sb2`

Figure 3: Stable Baselines logo

Stable Baselines 3 (SB3)

Stable Baselines3 provides open-source implementations of deep reinforcement learning (RL) algorithms in Python. The implementations have been benchmarked against reference codebases, and automated unit tests cover 95% of the code. The algorithms follow a consistent interface and are accompanied by extensive documentation, making it simple to train and compare different RL algorithms.

The backend is the main difference between SB2 (backend: TensorFlow) and SB3 (backend: PyTorch)[1]. Both versions have been developed by Antonin Raffin; SB2 when he was at ARMINES, SB3 during his time at DLR.

Original developer(s)	Antonin Raffin (DLR)
Current version	0.10.0
License	MIT
Link to AI4EU resource	`https://www.ai4eu.eu/resource/stable-baselines-3-sb3`

[1]Further differences are listed here: `https://stable-baselines3.readthedocs.io/en/master/guide/migration.html`