Columbus

We built Project Columbus [1], a framework for trivial 2D OpenAI Gym [2] environments that are supposed to test an agent's ability to solve tasks that require different forms of exploration. The state space is a 2D square. The action space is a 2D circle, which either controls the agent's speed or acceleration. Figure 1.1 shows an environment built using Columbus. The blue dot is the agent; in the top-left, a virtual joystick is displayed, that visualizes the agent's current output (section 1.5). The top-right shows a visualization for the CNNObservable (section 1.4, section 1.5).

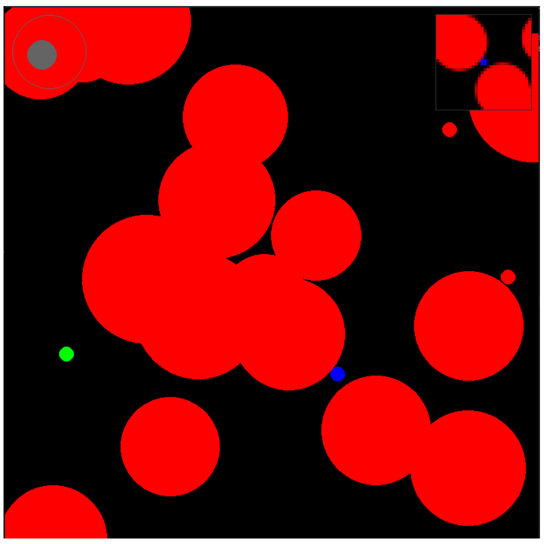
Figure 1.1: Example Environment built with Columbus

1.1 Entities

Environments contain entities that are described using configuration files. To allow building environments with more complex dynamics that can not be described by the format used by the configuration files, new environments can also be added by sub-classing the base ColumbusEnv class and overriding setup() and expanding __init__().

For many attributes of the environment or the entities, random generation based on defined bounds is possible. Entities are either spheres or rectangles, everything that is green ('reward') gives a configurable reward upon contact, and everything red ('enemy') gives a configurable penalty (negative reward) when touched. Auxiliary rewards/penalties can be enabled and are then calculated automatically. They decrease based on the inverse-square-law when moving away from rewards/enemies.

1.2 Physics

Entities can be solid, colliding with the agent upon contact. They can also be movable, gaining speed when pushed by the agent. Collisions between entities (and the agent) can have a configurable amount of elasticity.

The physics regarding collisions (and some other aspects) is implemented only phenomenological, allowing non-physical behavior given the right coefficient-setting. (For example: The 'mass' of objects does not have to follow a total ordering and collisions can ignore the conservation of energy to emulate the agent 'kicking' a ball when colliding.) Using default coefficients leads to behavior, that looks like real physics.

1.3 Human Interaction

It is possible to run environments in an interactive mode, where the input to the environment is not provided using the Gym Interface, but instead via mouse input on a virtual joystick. This allows testing environments for being hard/easy enough or having correctly tuned physics parameters.

It is also possible to interact with an environment to which an Neural Network (NN) is connected; overriding the controls to test the agent's reaction or re-sampling an environment if it contains random generation is possible.

1.4 Observation-Space

The observations emitted by the environment are defined by attaching 'Observables' to the environments. Available Observables are:

- StateObservable: Allows giving out the coordinates of designated entities (optionally relative to the agent) or the speed of entities.
- RayObservable: Generates outputs by casting rays from the agent's position (comparable to a LIDAR system). A configurable number of rays and independent channels for classes of entities.
- CompassObservable: Works like the StateObservable with coordinates relative to the agent, but we assign a bigger range of possible output values to those, that are close to zero. We found that agents relying only on a StateObservable often moved close to a reward and then just jiggled around. The output is described by

$$o_i = \tanh\left(\frac{r_i}{2\sqrt{\sum_{t=0}^{N} r_i^2}}\right) \quad \text{where } r_i = x_{i,object} - x_{i,agent}.$$

- CNNObservable: Returns a rendering as a tensor, that has a lower resolution, is zoomed in, and centered on the agent. (Not used in this thesis)

1.5 Visualizations

Columbus allows adding additional visualizations onto the rendered scene:

- The input received by the agent is displayed as a virtual joystick. In interactive mode, this can be turned into a usable joystick controlled using the mouse.
- Every observable implements a visualization, that is only dependent on the output it generated and not on the actual internal state of the environment. This allows sanity-checking Observables.
- A confidence ellipsoid (1σ) of the current covariance produced by the policy can be rendered around the agent.
- The path taken by the agent through the environment can be drawn.
- For PPO (and TRPL) the value function can be drawn as background for the environment. Since evaluating the value function for every point of the environment is expensive, the

value map is cached and only re-rendered, when the input perceived by the Observables could have changed (e.g. a reward was collected and teleported to a new location)

1.6 Columbus as a generic State-Space Simulator

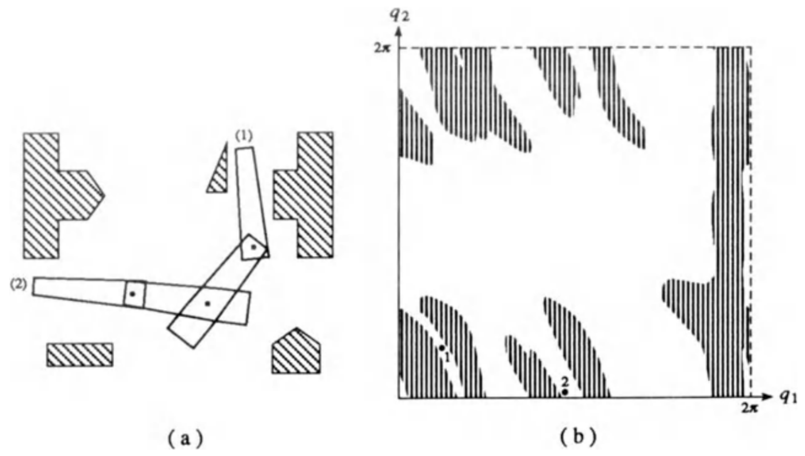

Figure 1.2: State-Space (C-Space) for a 2 DoF robot. Source: [3]

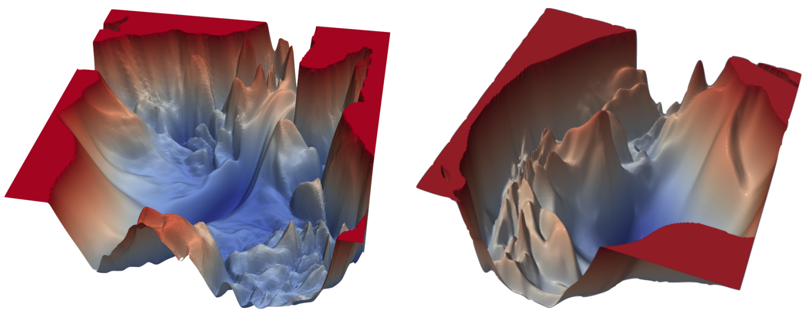

Figure 1.3: State-Space (Loss-Space) for a NN. Source: [4]

All environments that we build with Columbus share the same basic structure: Movement through a euclidean space, searching for a global optimum (reward) while avoiding low rewards (enemies) and escaping local optima (dead ends). A Reinforcement Learning (RL)-Problem comprised of a robot trying to perform a task in an environment with obstacles by controlling the individual motors looks like a very different environment. But similarly, small movements through the state-space also result in small movements in the state-space (and observation space depending on how the observations are formed). We can describe the observation space as a Riemann-Manifold with local euclidean topology; our actions describe continuous movement through this observation-space (Figure 1.2). Even training a NN can be regarded as an agent moving through a space with locally euclidean topology (Figure 1.3). The state space is in this case spanned by the parameters of the model we wish to train and our action is the parameter update. Here we also need to escape local optima to reach a minimal loss.

But comparing the state-space of Columbus to that of a NN optimization is probably a stretch since the topological effects of the hierarchical nature of NN on the state-space are in no way represented in Columbus.

Bibliography

[1] Roth, D.: Columbus, https://git.dominik-roth.eu/dodox/Columbus

[2] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W.: OpenAI Gym (2016)

[3] Latombe, J.C.: Robot Motion Planning. Springer US, Boston, MA (1991). https://doi.org/10.1007/978-1-4615-4022-9, http://link.springer.com/10.1007/978-1-4615-4022-9

[4] Li, H., Xu, Z., Taylor, G., Studer, C., Goldstein, T.: Visualizing the Loss Landscape of Neural Nets p. 11