# Алгоритм обучения

Для обучения использован алгоритм DDPG. Само обучение производится в двух вариантах – с использованием HER (Hindsight Experience Replay) и без. Результаты сравниваются и для работы берется лучшая модель.

Нейронная сеть, используемая в алгоритме, имеет следующий вид:

Изображение выглядит как текст

Автоматически созданное описание

Рисунок

На вход она получает State (массив чисел без вектора позиции цели) и Goal (отдельно вектор позиции цели), на выход выдаёт три числа в диапазоне [-1, 1] и далее нормализует их под другой заданный диапазон [action\_low, action\_high].

# Обучение

Для обучения алгоритму требуется передать State, состоящий из 12 нормализованных чисел (порядок важен):

1. Позиция цели – 3 числа (вектор).
2. Векторное расстояние между позицией цели и позицией конца руки – 3 числа (вектор).
3. Позиция конца руки – 3 числа (вектор).
4. Значение угла каждого сустава роборуки (суставов у руки 3) – 3 числа.

Алгоритм выдаёт 3 числа – значения углов, на которые надо повернуться каждому суставу (число -n говорит о том, что суставу надо повернуться на n градусов против часовой стрелки, а n – на n градусов по часовой).

Обучение длится 100 эпох, каждая из которых содержит 20 эпизодов, где 1 эпизод – это 500 обращений к алгоритму с целью получить action. В начале каждого эпизода среда обучения (сцена Unity) сбрасывается на стартовые значения.

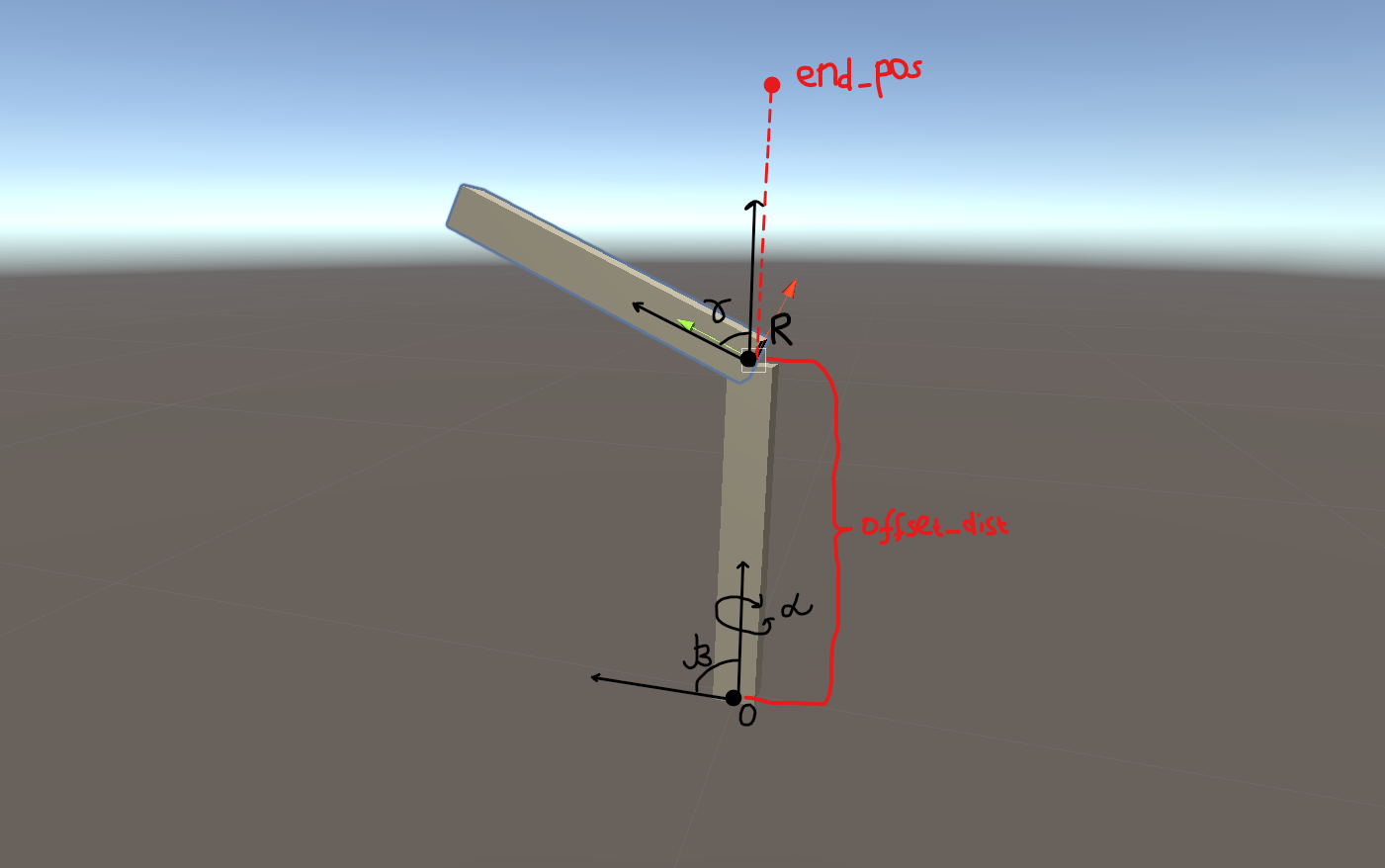
Каждый шаг (т.е. 500 раз за эпизод) роборука получает награду -1 с целью ускорить нахождение цели. При достижении цели она получает награду 10. При использовании HER происходит дополнительная оценка текущего состояния, а наградой является число, равное текущему расстоянию от конца руки до цели.

# Использование обученной модели на сцене Unity

Для проверки алгоритма в сцене Unity сперва загружается обученная модель командой agent.saver.restore(agent.sess, "model/her\_rotate\_scaled.ckpt"), после чего в цикле происходит обращение к агенту для получения массива action командой action = agent.choose\_action([state], [goal], 0).

# Матричные преобразования

Для использования алгоритма на настоящей роборуке необходимо научиться получать state из настоящей среды. Главная проблема состоит в поиске позиции конца руки. Для её решения необходимо использовать матрицы преобразования.



Рисунок

Чтобы найти позицию конца руки, необходимо сперва совершить сдвиг на offset\_dist (перейти в смещенную систему координат, то есть смещаемся к точке R, вокруг которой будем вращать другое сочленение роборуки), совершить поворот на угол γ, вернуться обратно на offset\_dist (в главную систему координат). После этого необходимо также совершить повороты на углы β и α (не переходя в другую систему координат, т. к. точки вращения для этих углов находятся в точке 0). Всё это по порядку выполнено в функции get\_rotate\_offset\_position, которая на вход принимает следующие параметры:

1. rotate\_axis0 – ось вращения (OX, OY или OZ), вокруг которой необходимо совершить поворот на угол α;
2. rotate\_angle0 – угол α;
3. rotate\_axis1– ось вращения (OX, OY или OZ), вокруг которой необходимо совершить поворот на угол β;
4. rotate\_angle1 – угол β;
5. rotate\_axis2 – ось вращения (OX, OY или OZ), вокруг которой необходимо совершить поворот на угол γ;
6. rotate\_angle2 – угол γ;
7. offset\_axis – ось (OX, OY или OZ), вдоль которой совершается смещение к точке R, являющейся точкой вращения на угол γ;
8. offset\_dist – расстояние от точки 0 до точки R;
9. end\_position – координаты точки конца руки в вытянутом состоянии (точка end\_pos на рис. 2).

Функция возвращает модифицированные координаты (вектор (x, y, z)), то есть координаты точки, которая была последовательно повернута на три угла, при чем на один из них в смещенной системе координат.

Изображение выглядит как текст

Автоматически созданное описание

Для перехода в другую систему координат и поворота в ней на угол γ вызывается функция rotate\_and\_offset, в которой матрицы смещения и поворота умножаются на матрицу координат точки old\_coord (точка показана на рис. 2). Она принимает следующие параметры:

1. offset\_axis – ось (OX, OY или OZ), вдоль которой совершается смещение к точке, являющейся точкой вращения на угол alpha;
2. offset – смещение;
3. rotate\_axis – ось вращения (OX, OY или OZ), вокруг которой необходимо совершить поворот на угол alpha;
4. alpha – угол вращения;
5. old\_coord ­– координаты точки, которую необходимо модифицировать.

Функция возвращает координаты точки (вектор (x, y, z)), повернутой в смещенной системе координат.

Изображение выглядит как текст

Автоматически созданное описание

В функции get\_rotate\_offset\_position (возвращает координаты повернутой точки (вектор (x, y, z))) для поворота на углы β и α используется функция rotate\_around\_axis:

**Изображение выглядит как текст

Автоматически созданное описание**

Функции, возвращающие матрицы смещения и поворота:

Изображение выглядит как стол

Автоматически созданное описание

Изображение выглядит как стол

Автоматически созданное описание

Матрицы поворота взяты из источника <https://www.bsuir.by/m/12_113415_1_70397.pdf>:

Изображение выглядит как текст

Автоматически созданное описание

Функция умножения матриц.

Она принимает следующие параметры:

1. Матрицу a;
2. Матрицу b;
3. Число m – количество элементов в строке первой матрицы;
4. Число n – количество элементов в столбце первой матрицы;
5. Число p – количество элементов в столбце второй матрицы;
6. Число q – количество элементов в строке второй матрицы.

Функция возвращает матрицу размера m на q (m элементов в столбце, q элементов в строке);

Изображение выглядит как текст

Автоматически созданное описание

# Использование обученной модели на настоящей роборуке

Для работы с обученной моделью необходимо научиться передавать ей State из реального мира. Для этого используется функция read\_state. Она обращается к трём сервам, получает у них значения углов, вычитая из каждого из них 120, так как у руки в вытянутом состоянии все сервы имеют значение в 120 градусов (это позволяет серве вращаться в обе стороны, т.к. её диапазон [0, 240]) Это необходимо для алгоритма, который обучен на том, что у руки в вытянутом состоянии углы равны нулю. По значениям этих углов и длин сочленений далее находится позиция конца руки с помощью функции get\_rotate\_offset\_position. Далее собирается массив state из вектора расстояния между концом руки и целью, вектора позиции конца руки и трех значения углов (нормированных), а также массив goal, состоящий из вектора позиции цели (задается вручную).

Изображение выглядит как текст

Автоматически созданное описание

Для применения обученной модели получаем action из агента и передаём его в функцию set\_action.

Изображение выглядит как текст

Автоматически созданное описание

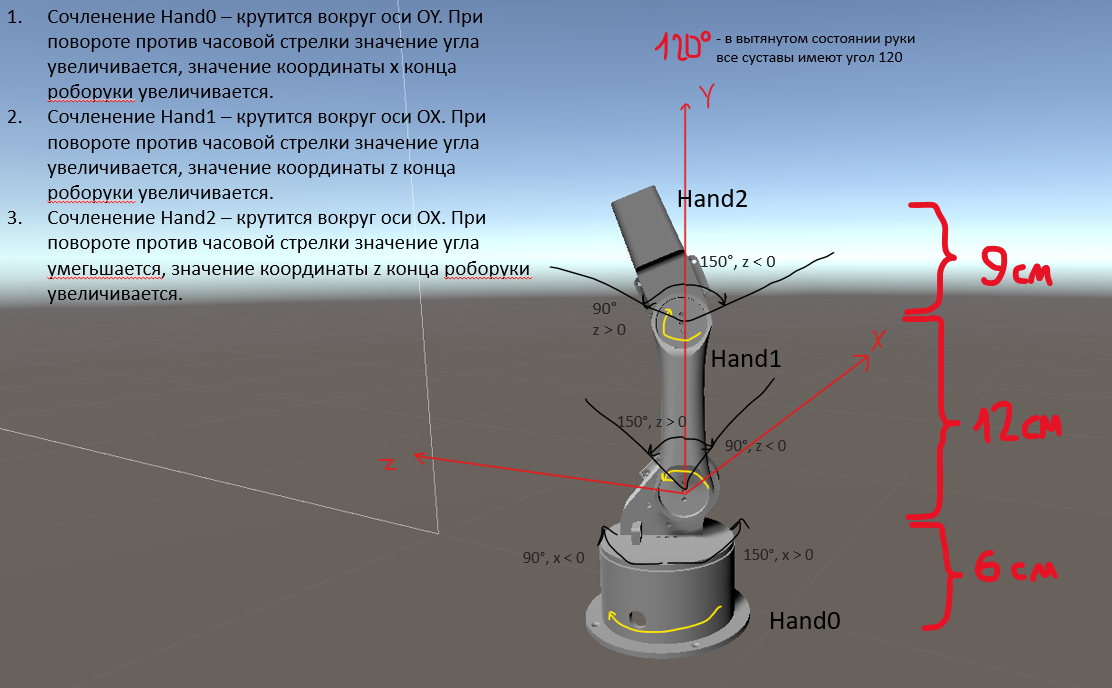
В функции set\_action считывается текущий угол каждой сервы, и к нему прибавляется угол, на который необходимо повернуться. Action – массив из трех значений, где первый элемент – это дельта для первого сустава, второй – для второго, третий – для третьего.

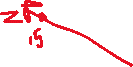
Функция check\_if\_done оценивает расстояние от конца руки до цели. Порог необходимо подбирать.

Изображение выглядит как текст

Автоматически созданное описание

Роборука с её параметрами изображена на рис. 3.





Рисунок

