Deep RL Arm Manipulation

This project is based on the Nvidia open source project "jetson-reinforcement" developed by Dustin Franklin and train on Nvidia Jeston TX2 platform. The goal of the project is to create a DQN agent and define reward functions to teach a robotic arm to realize two following objectives:

1. Have any part of the robot arm touch the object of interest, with at least a 90% accuracy.
2. Have only the gripper base of the robot arm touch the object, with at least a 80% accuracy.

Background

C++ API (application programming interface)

C++ API provides an interface to the Python code written with PyTorch, but the wrappers use Python's low-level C to pass memory objects between the user's application and Torch without extra copies. By using a compiled language (C/C++) instead of an interpreted one, performance is improved, and speeded up even more when GPU acceleration is leveraged.

Gazebo Arm Plugin

This plugin is responsible for creating the DQN agent and training it to learn to touch the prop. The gazebo plugin shared object file, `libgazeboArmPlugin.so`, attached to the robot model in `gazebo-arm.world`, is responsible for integrating the simulation environment with the RL agent. The plugin is defined in the `ArmPlugin.cpp` file, also located in the `gazebo` folder.

The `ArmPlugin.cpp` file takes advantage of the C++ API. This plugin creates specific constructor and member functions for the class `ArmPlugin` defined in `ArmPlugin.h`.

You can refer to the documentation for more details on the above：

- Subscribers in Gazebo
- Gazebo API

Parameters Tuning

There are several sections need to be tuned in `ArmPlugin.cpp`, I used different parameter settings for two tasks.

Task 1

- **Hyperparameters**

I used velocity control in this task, i.e., `VELOCITY_CONTROL` is set to be `true`.

`INPUT_WIDTH` and `INPUT_HEIGHT` are reduced into 64 from 512 to save memory.

`USE_LSTM` is true with lstm size of 256.

The initial reward win and loss are set as +-10.

```
// Turn on velocity based control
#define VELOCITY_CONTROL true
#define VELOCITY_MIN -0.2f
#define VELOCITY_MAX  0.2f

// TODO - Tune the following hyperparameters
#define INPUT_WIDTH   64
#define INPUT_HEIGHT  64
#define OPTIMIZER "RMSprop"
#define LEARNING_RATE 0.05f
#define REPLAY_MEMORY 10000
#define BATCH_SIZE 32
#define USE_LSTM true
#define LSTM_SIZE 256

// TODO - Define Reward Parameters
#define REWARD_WIN  10.0f
#define REWARD_LOSS -10.0f
```

- Issue a reward based on collision between the desired arm part and the object.

```
if((strcmp(contacts->contact(i).collision1().c_str(), COLLISION_ITEM) == 0))
{
    rewardHistory = REWARD_WIN;
    newReward  = true;
    endEpisode = true;
    return;
}
else {
    // Give penalty for non correct collisions
    rewardHistory = REWARD_LOSS;
    newReward  = true;
    endEpisode = true;
}
```

- Set a penalty for robot wasting time.

```
if( maxEpisodeLength > 0 && episodeFrames > maxEpisodeLength )
{
    printf("ArmPlugin - triggering EOE, episode has exceeded %i frames\n", maxEpisodeLength);
    rewardHistory = REWARD_LOSS;
    newReward     = true;
    endEpisode    = true;
}
```

- Set a penalty for robot hitting the ground.

```
if( gripBBox.min.z < groundContact )
{
    printf("GROUND CONTACT, EOE\n");
    rewardHistory = REWARD_LOSS * 20.0f;
    newReward     = true;
    endEpisode    = true;
}
```

- A `distPenalty` reward is added to encourage arm approach the object quickly for the intermediary reward.

```
const float distDelta  = lastGoalDistance - distGoal;
const float alpha = 0.2f;

// compute the smoothed moving average of the delta of the distance to the goal
avgGoalDelta  = (avgGoalDelta * alpha) + (distDelta * (1.0f - alpha));
float distPenalty = (1.0f - exp(distGoal)) * 0.1f;

if(avgGoalDelta > 0)
{rewardHistory = REWARD_WIN * 0.5f - distGoal * 0.5f + distPenalty ;}
else
{rewardHistory = avgGoalDelta + distPenalty;}
newReward     = true;
```

Task 2

- Hyperparameters

  This task is more challenge, I failed to realize velocity control, hence position control is used here, i.e., `VELOCITY_CONTROL` is false.

  I don't find obvious improvement using LSTM, but still use it with a smaller `LSTM_SIZE` here.

  `EPS_END` is reduced to 0.01 so that the robot arm is more stable when it finish learning.

  `Adam` optimizer is adopted due to its advantage in other deep learning applications.

  `REWARD_WIN` and ·REWARD_LOSS` are set as +-1 to have a better understanding.

  Note: `EPS_DECAY` can be reduced if your reward function is appropriate. The premises is your robot should find the right way within this epochs. This can save lots of time when you try to save your final result.

  ```
  // Turn on velocity based control
  #define VELOCITY_CONTROL false
  #define VELOCITY_MIN -0.2f
  #define VELOCITY_MAX  0.2f

  // Define DQN API Settings
  #define INPUT_CHANNELS 3
  #define ALLOW_RANDOM true
  #define DEBUG_DQN false
  ```

```
#define GAMMA 0.9f
#define EPS_START 0.9f
#define EPS_END 0.01f
#define EPS_DECAY 200

// TODO - Tune the following hyperparameters
#define INPUT_WIDTH   64
#define INPUT_HEIGHT  64
#define OPTIMIZER "Adam"
#define LEARNING_RATE 0.01f
#define REPLAY_MEMORY 10000
#define BATCH_SIZE 128
#define USE_LSTM true
#define LSTM_SIZE 16

// TODO - Define Reward Parameters
#define REWARD_WIN  1.0f
#define REWARD_LOSS -1.0f
```

- More sophisticated reward function is adopted when arm achieve its goal.

```
rewardHistory = REWARD_WIN*10.0f + (1.0f - (float(episodeFrames) /
float(maxEpisodeLength))) * REWARD_WIN * 100.0f;
```

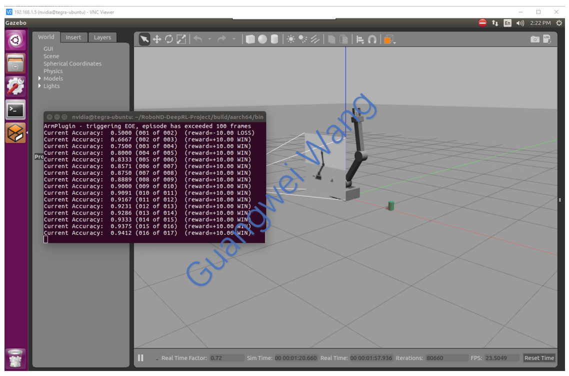
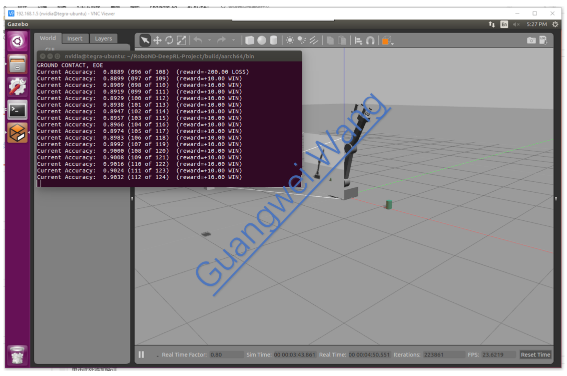
- An improved intermediary reward for task 2

```
if( episodeFrames > 1 )
{
    const float distDelta  = lastGoalDistance - distGoal;
    const float alpha = 0.9f;
    // compute the smoothed moving average of the delta of the distance to the goal
    avgGoalDelta  = (avgGoalDelta * alpha) + (distDelta * (1.0f - alpha));
    float distPenalty = (1.0f - exp(distGoal));
    if(avgGoalDelta > 0.01)
    {rewardHistory = (REWARD_WIN + distPenalty*0.1f)*0.1f;}
    else
    {rewardHistory = - distGoal*2.0f;}
    newReward     = true;
}
```

Results

Task	Total Runs	Accuracy
1: Arm	17	94.12%
1: Arm	124	90.32%
2: Gripper base	35	80.00%
2: Gripper base	491	90.02%

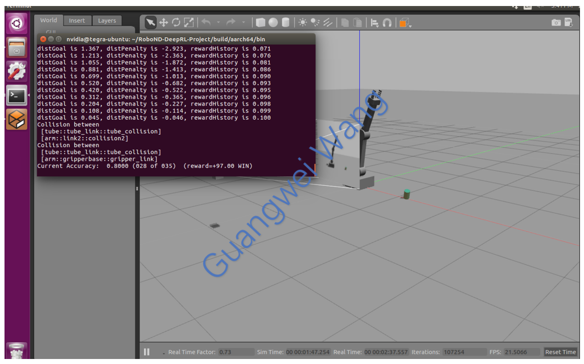
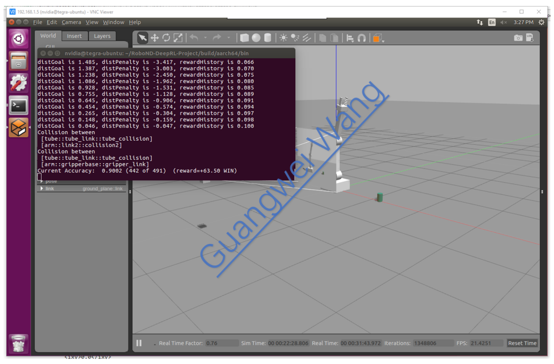
- Task 1: Any parts of arm touching the prop

```
ArmPlugin - triggering EOE, episode has exceeded 100 frames
Current Accuracy:  0.5000 (001 of 002)  (reward=-10.00 LOSS)
Current Accuracy:  0.6667 (002 of 003)  (reward=+10.00 WIN)
Current Accuracy:  0.7500 (003 of 004)  (reward=+10.00 WIN)
Current Accuracy:  0.8000 (004 of 005)  (reward=+10.00 WIN)
Current Accuracy:  0.8333 (005 of 006)  (reward=+10.00 WIN)
Current Accuracy:  0.8571 (006 of 007)  (reward=+10.00 WIN)
Current Accuracy:  0.8750 (007 of 008)  (reward=+10.00 WIN)
Current Accuracy:  0.8889 (008 of 009)  (reward=+10.00 WIN)
Current Accuracy:  0.9000 (009 of 010)  (reward=+10.00 WIN)
Current Accuracy:  0.9091 (010 of 011)  (reward=+10.00 WIN)
Current Accuracy:  0.9167 (011 of 012)  (reward=+10.00 WIN)
Current Accuracy:  0.9231 (012 of 013)  (reward=+10.00 WIN)
Current Accuracy:  0.9286 (013 of 014)  (reward=+10.00 WIN)
Current Accuracy:  0.9333 (014 of 015)  (reward=+10.00 WIN)
Current Accuracy:  0.9375 (015 of 016)  (reward=+10.00 WIN)
Current Accuracy:  0.9412 (016 of 017)  (reward=+10.00 WIN)
```

```
GROUND CONTACT, EOE
Current Accuracy:  0.8889 (096 of 108)  (reward=-200.00 LOSS)
Current Accuracy:  0.8899 (097 of 109)  (reward=+10.00 WIN)
Current Accuracy:  0.8909 (098 of 110)  (reward=+10.00 WIN)
Current Accuracy:  0.8919 (099 of 111)  (reward=+10.00 WIN)
Current Accuracy:  0.8929 (100 of 112)  (reward=+10.00 WIN)
Current Accuracy:  0.8938 (101 of 113)  (reward=+10.00 WIN)
Current Accuracy:  0.8947 (102 of 114)  (reward=+10.00 WIN)
Current Accuracy:  0.8957 (103 of 115)  (reward=+10.00 WIN)
Current Accuracy:  0.8966 (104 of 116)  (reward=+10.00 WIN)
Current Accuracy:  0.8974 (105 of 117)  (reward=+10.00 WIN)
Current Accuracy:  0.8983 (106 of 118)  (reward=+10.00 WIN)
Current Accuracy:  0.8992 (107 of 119)  (reward=+10.00 WIN)
Current Accuracy:  0.9000 (108 of 120)  (reward=+10.00 WIN)
Current Accuracy:  0.9008 (109 of 121)  (reward=+10.00 WIN)
Current Accuracy:  0.9016 (110 of 122)  (reward=+10.00 WIN)
Current Accuracy:  0.9024 (111 of 123)  (reward=+10.00 WIN)
Current Accuracy:  0.9032 (112 of 124)  (reward=+10.00 WIN)
```

- Task 2: Gripper base touching the prop
  https://youtu.be/pjYxtJ0pTRY

```
distGoal is 1.367, distPenalty is -2.923, rewardHistory is 0.071
distGoal is 1.213, distPenalty is -2.363, rewardHistory is 0.076
distGoal is 1.055, distPenalty is -1.872, rewardHistory is 0.081
distGoal is 0.881, distPenalty is -1.413, rewardHistory is 0.086
distGoal is 0.699, distPenalty is -1.013, rewardHistory is 0.090
distGoal is 0.520, distPenalty is -0.682, rewardHistory is 0.093
distGoal is 0.420, distPenalty is -0.522, rewardHistory is 0.095
distGoal is 0.312, distPenalty is -0.365, rewardHistory is 0.096
distGoal is 0.204, distPenalty is -0.227, rewardHistory is 0.098
distGoal is 0.108, distPenalty is -0.114, rewardHistory is 0.099
distGoal is 0.045, distPenalty is -0.046, rewardHistory is 0.100
Collision between
 [tube::tube_link::tube_collision]
 [arm::link2::collision2]
Collision between
 [tube::tube_link::tube_collision]
 [arm::gripperbase::gripper_link]
Current Accuracy:  0.8000 (028 of 035)  (reward=+97.00 WIN)
```

```
distGoal is 1.485, distPenalty is -3.417, rewardHistory is 0.066
distGoal is 1.387, distPenalty is -3.003, rewardHistory is 0.070
distGoal is 1.238, distPenalty is -2.450, rewardHistory is 0.075
distGoal is 1.086, distPenalty is -1.962, rewardHistory is 0.080
distGoal is 0.928, distPenalty is -1.531, rewardHistory is 0.085
distGoal is 0.755, distPenalty is -1.128, rewardHistory is 0.089
distGoal is 0.645, distPenalty is -0.906, rewardHistory is 0.091
distGoal is 0.454, distPenalty is -0.574, rewardHistory is 0.094
distGoal is 0.265, distPenalty is -0.304, rewardHistory is 0.097
distGoal is 0.148, distPenalty is -0.159, rewardHistory is 0.098
distGoal is 0.046, distPenalty is -0.047, rewardHistory is 0.100
Collision between
 [tube::tube_link::tube_collision]
 [arm::link2::collision2]
Collision between
 [tube::tube_link::tube_collision]
 [arm::gripperbase::gripper_link]
Current Accuracy:  0.9002 (442 of 491)  (reward=+63.50 WIN)
```

Challenges

Challenge1: Object Randomization

The object will instantiate at different locations along the x-axis.

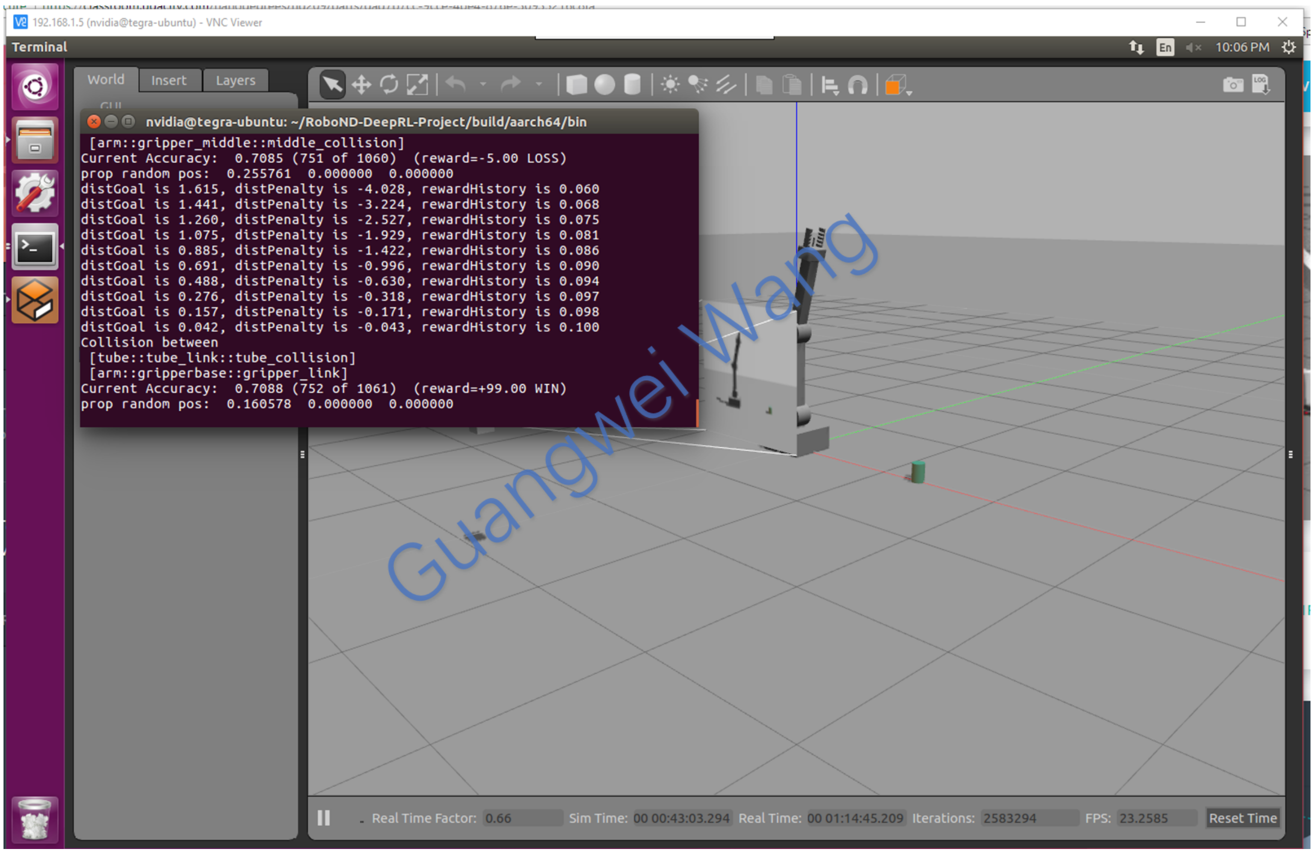
- Use a larger `INPUT_WIDTH` and `INPUT_HEIGHT` value to distinguish object more clear.

```
#define INPUT_WIDTH   128
#define INPUT_HEIGHT  128
```

- Revise the intermediary reward, give more penalty for arm stop

```
if(avgGoalDelta > 0.01)
{rewardHistory = (REWARD_WIN + distPenalty*0.1f)*0.1f;}
else
{rewardHistory = REWARD_LOSS - distGoal*2.0f;}
```

- 70% accuracy is achieved. A better result can be obtained with larger input width and height value, however, tx2 cannot deal with this computational task.



192.168.1.5 (nvidia@tegra-ubuntu) - VNC Viewer
Terminal
World
Insert
Layers
nvidia@tegra-ubuntu: ~/RoboND-DeepRL-Project/build/aarch64/bin
Real Time Factor: 0.66
Sim Time: 00 00:43:03.294
Real Time: 00 01:14:45.209
Iterations: 2583294
FPS: 23.2585
Reset Time

```
 [arm::gripper_middle::middle_collision]
Current Accuracy:  0.7085 (751 of 1060)  (reward=-5.00 LOSS)
prop random pos:  0.255761  0.000000  0.000000
distGoal is 1.615, distPenalty is -4.028, rewardHistory is 0.060
distGoal is 1.441, distPenalty is -3.224, rewardHistory is 0.068
distGoal is 1.260, distPenalty is -2.527, rewardHistory is 0.075
distGoal is 1.075, distPenalty is -1.929, rewardHistory is 0.081
distGoal is 0.885, distPenalty is -1.422, rewardHistory is 0.086
distGoal is 0.691, distPenalty is -0.996, rewardHistory is 0.090
distGoal is 0.488, distPenalty is -0.630, rewardHistory is 0.094
distGoal is 0.276, distPenalty is -0.318, rewardHistory is 0.097
distGoal is 0.157, distPenalty is -0.171, rewardHistory is 0.098
distGoal is 0.042, distPenalty is -0.043, rewardHistory is 0.100
Collision between
 [tube::tube_link::tube_collision]
 [arm::gripperbase::gripper_link]
Current Accuracy:  0.7088 (752 of 1061)  (reward=+99.00 WIN)
prop random pos:  0.160578  0.000000  0.000000
```

Further Work

- Investigate how to use LSTM appropriately
- Realize velocity control for task 2
- Add more constraints to make arm touching prop softly.
- Realize project challenges, such as random prop position, high arm DOF, etc.