Instituto Infnet

ASSESSMENT

Elizabeth Maria de Carvalho

Matrícula: 11681797771

Instituto Infnet

Assessment

Trabalho apresentado como recurso avaliativo da disciplina de Veículos Autônomos, ministrada pelo M.e. Adalberto Oliveira.

Rio de Janeiro
2021

OBJETIVO

Reforçar os conceitos aprendidos na disciplina de “Veículos Autônomos”.

Tarefa Única

1. **Durante a disciplina foram trabalhadas as etapas que permitem que um veículo robótico realize tarefas de maneira autônoma: controle de posição/postura, captura e tratamento de imagens, extração de dados a partir de uma imagem e controle de robôs utilizando os atributos da imagem, podendo ser do tipo IBVS (Image Based Visual Servoing) ou PVBS (Position Based Visual Servoing). Com base nestes conceitos e utilizando todo o material disponibilizado, desenvolva um pacote para o framework ROS que seja capaz de controlar o simulador TurtleBot3, modelo Waffle (Figura 1) para que ele realize uma tarefa de regulação de posição utilizando como base a abordagem de controle visual PVBS, encontrando a posição no mundo do alvo desejado, e se deslocando até ele de maneira autônoma.**

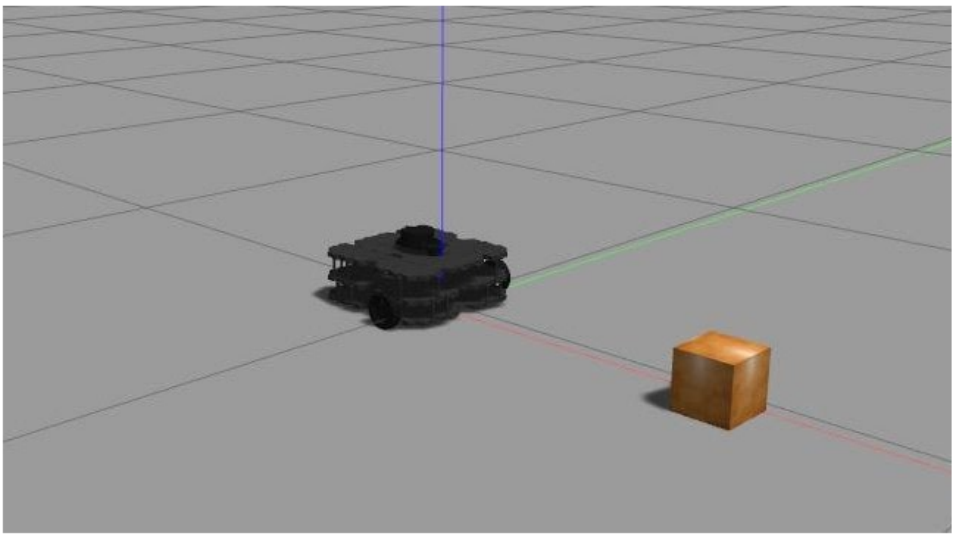


Figura 1 - Simulador Turtlebot3 (esquerda) e um objeto colocado no ambiente (direita).

O trabalho deverá atender aos seguintes requisitos:

- **Encontrar um objeto posicionado dentro do simulador, desde que dentro do limite de visão da câmera;**
- **Segmentar os objetos para extrair os seus atributos;**
- **Extrair o atributo “Ponto da Base” e sua coordenada em pixel;**
- **Encontrar a distância entre este ponto e o robô;**

- **Transformar o ponto do referencial da imagem para o referencial do mundo e utilizar como coordenada de destino;**
- **Controlar o robô de forma que ele chegue em um ponto a 30cm de distância da coordenada da base do objeto (regulação parcial);**
- **Exibir as telas de imagem original, máscara de segmentação e imagem processada;**
- **Informar a distância entre o robô e o objeto de destino, coordenada em pixel do ponto da base e a coordenada do objeto no mundo, ou por impressão no terminal ou por publicação no ROS (info ou publicação de mensagem);**

A aplicação deverá ser executada a partir de um arquivo *.launch, que deverá iniciar todos os nós do projeto, caso seja utilizado mais de um nó, assumindo que o simulador já esteja em execução. Os seguintes elementos deverão ser entregues como parte da avaliação:

- **Pacote ROS completo, com o nomenclatura AT_NOME_SOBRENOME.zip;**
- **Relatório contendo a descrição do(s) nó(s) desenvolvidos, os parâmetros utilizados, tais como limites da máscara, ganhos do controlador, dados do robô e demais configurações utilizadas, bem como os resultados da simulação, como imagens originais e segmentadas e máscara, e a estrutura do seu pacote (árvore de arquivos).**

Extra: realizar este mesmo trabalho fazendo a regulação para uma sequência de pelo menos dois ou mais objetos no ambiente. Para este caso, descreva no relatório qual deve ser a sequência que o robô deve seguir durante a operação.

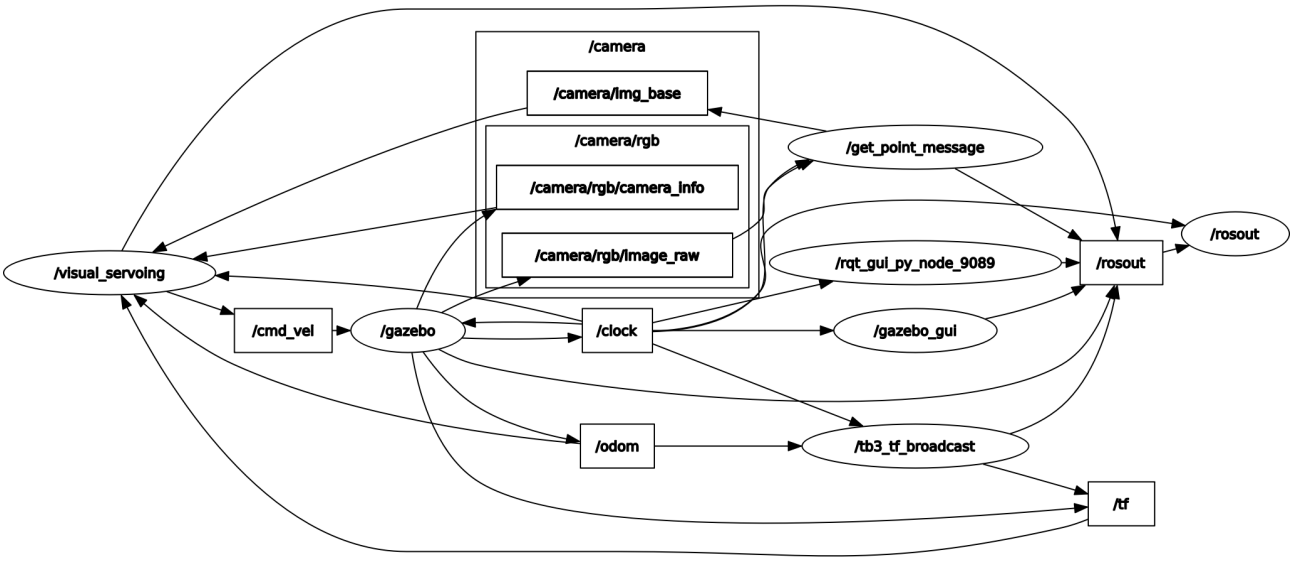
```
                        MANUAL TÉCNICO + EVIDÊNCIAS
#============================================================
#Etapa 1: Criando o pacote ec_at_va
      $ initros1
      $ source ~/catkin_ws/devel/setup.bash
      $ cd ~/catkin_ws/src/
      $ catkin_create_pkg ec_at_va rospy geometry_msgs
#============================================================
#Etapa 2: Realizada a adaptação dos scripts/launchs cedidos pelo professor para:
   -  2.1: calibrador_hsv_yaml_ROS.py
         -  Função = Criação da máscara para binarização do objeto lido no tópico
            informado via input, salvando os parâmetros (limites superiores e
            inferiores escolhidos) no arquivo 'cfg/mask_param.yaml'.
```

- 2.2: get_point_from_message.launch
  - Nó = get_point_message
  - Função = Coletada através da leitura de um tópico da câmera do Turtlebot3 no Gazebo, realizar o processamento da imagem utilizando os parâmetros da máscara e aplicando filtros para de forma a realizar a segmentação do objeto na cena, envolvendo-o com um boundbox, calculando o ponto central da base exibindo-o e publicando essa informação em um tópico.
  - Script = get_point_from_message.py
  - Parâmetros de entrada = realiza a leitura dos limites superior e inferior da máscara contidos no arquivo 'mask_param.yaml' e se a janela de visualização do resultado da aplicação do processo de máscara na imagem lida deverá ou não ser apresentada (show_image).
  - Tópicos = realiza leitura do tópico '/camera/rgb/image_raw' e escrita no tópico '/camera/img_base'.
- 2.3: tb3_transformations.launch
  - Nó = tb3_tf_broadcast
  - Função = Realizar a árvore de transformações necessária para que o ponto em pixel ao final seja uma coordenada do objeto no mundo.
  - Tópico = realiza leitura do tópico '/odom'.

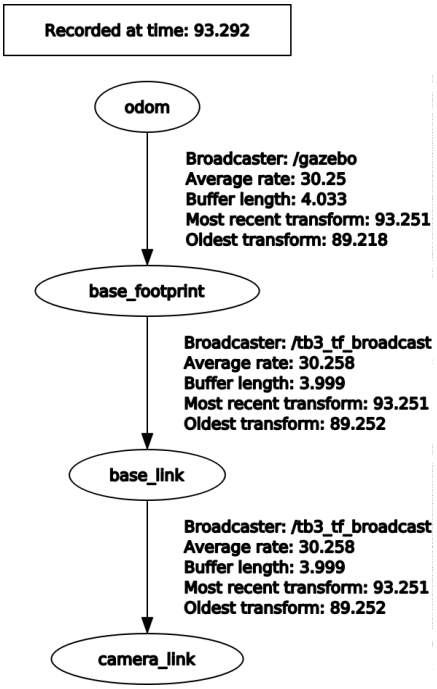
#==
#Etapa 3: Criação do launch 'tb3_visual_servoing_pbvs.launch', responsável por chamar os outros launchers, iniciando assim todos os nós necessários de uma única vez, bem como carregar os parâmetros/script necessários realização do PBVS.

- Nó = visual_servoing
- Função = Controlar a movimentação do Turtlebot3 por meio de postura parcial, tendo em vista que o mesmo ficará a uma distância de 30 cm do objeto cuja ponto da base na imagem em pixel está sendo publicado pelo nó get_point_message e será convertido para uma coordenada da câmera através da função get_img_point que para realizar essa tarefa recebe as informações da câmera a fim de realizar os cálculos de transformação necessários. Depois será convertido para uma coordenada no mundo pelo tópico tb3_tf_broadcast, onde finalmente será o ponto de destino a ser alcançado pelo robô através da função cartesian_control que realiza os cálculos do sinal de controle para orientar o robô e é baseado no algoritmo de controle cartesiano, utilizando a coordenada em que se encontra o robô e a de destino para verificar a diferença que falta ser alcançada aliado ao PID (controle de ganhos) para ajustar a postura de forma suave. O processo de controle do robô inicia-se automaticamente a partir do momento em que é lida a informação do ponto da base da imagem, ou seja, que o objeto foi detectado, do contrário, o robô se manterá em rotação com velocidade angular de 0.5 para realizar a busca do objeto limitado a seu campo de visão. O resultado final é um controle realizado no campo de visão do robô que estima a posição do objeto em relação à câmera (PBVS = Position-Based Visual Servo):
- Script = visual_servoing_pbvs.py
- Parâmetros = ganhos do controlador PID para regulação de postura (K_eu e K_ev, velocidade linear e velocidade angular respectivamente, ambos com 0.25, sendo assim o robô irá realizar seu deslocamento e ajuste de orientação em uma velocidade considerada adequada), altura da câmera em relação ao robô (10cm do

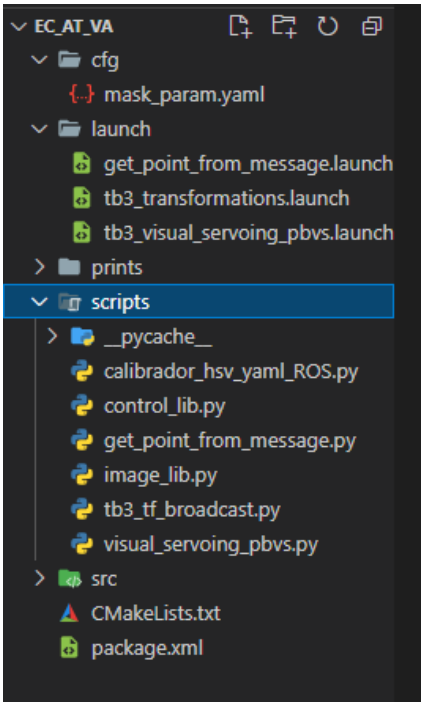
```
      solo)
   -  Tópicos = realizada a leitura dos tópicos ‘/camera/img_base’,
      ‘/camera/rgb/camera_info’ e ‘/odom’, e publica no tópico  ‘/cmd_vel’
#============================================================
```



Relação entre os tópicos gerada pela ferramenta RQT_GRAPH

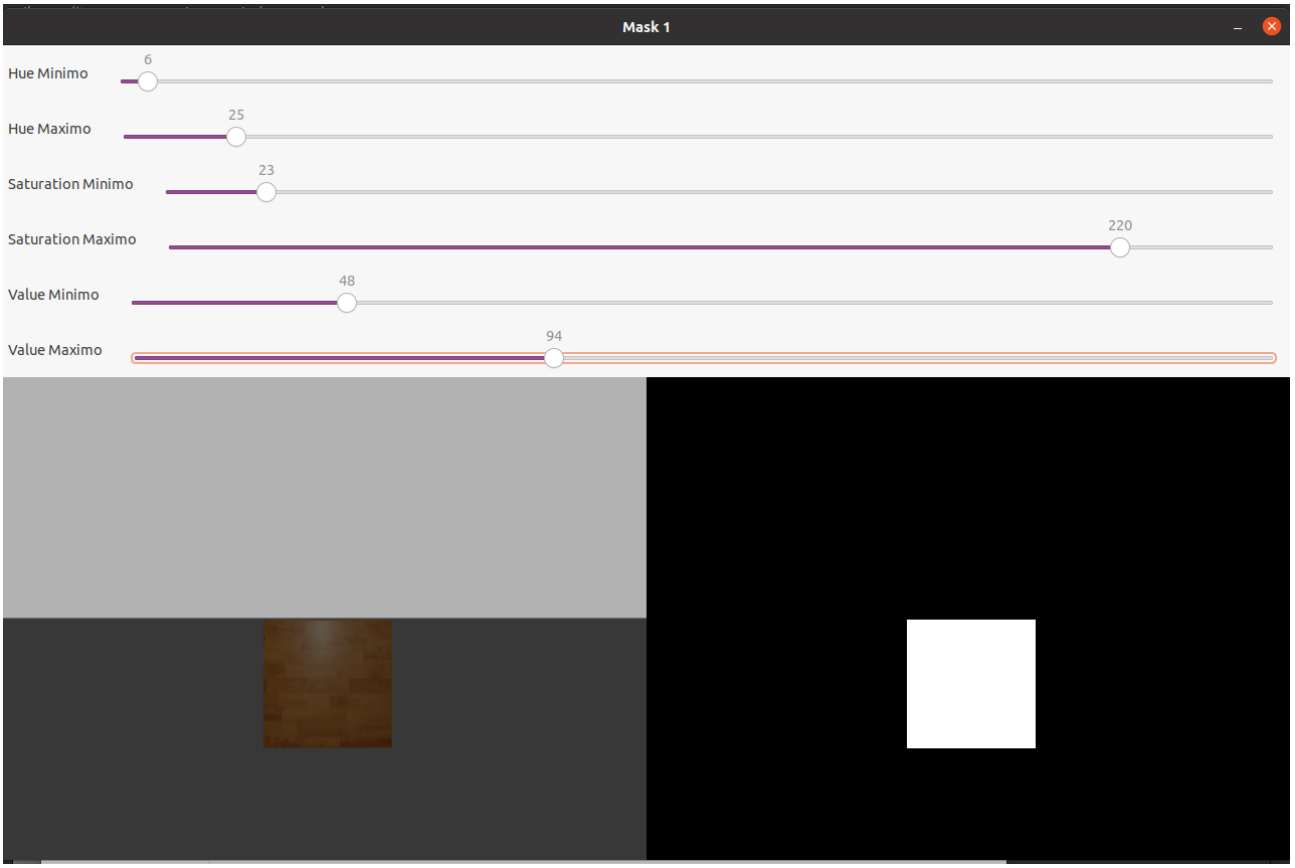


Árvore de transformações gerada pelo nó “tb3_tf_broadcast”

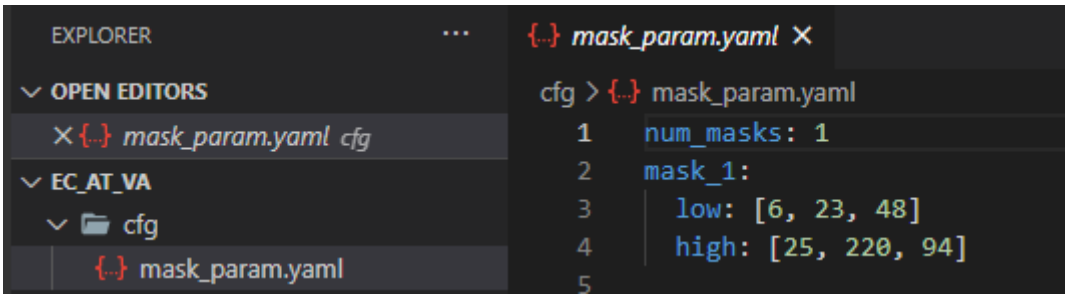


Árvore de diretórios e arquivos do pacote "ec_at_va"

```
#============================================================
#Etapa 4: Dar permissão de execução (chmod +x) para os scripts abaixo:
    -   calibrador_hsv_yaml_ROS.py
    -   get_point_from_message.py
    -   tb3_tf_broadcast.py
    -   visual_servoing_pbvs.py
#============================================================
#Etapa 5: Build do pacote
$ cd ~/catkin_ws/
$ catkin_make
#============================================================
#Etapa 6: Processamento da imagem e criação da máscara do objeto. O resultado
contendo os valores dos limites superiores e inferiores e salvo no arquivo
'mask_param.yaml' informado via input, bem como o número de máscaras criadas.
$ cd ~/catkin_ws/src/ec_at_va/cfg
$ rosrun ec_at_va calibrador_hsv_yaml_ROS.py
#============================================================
```



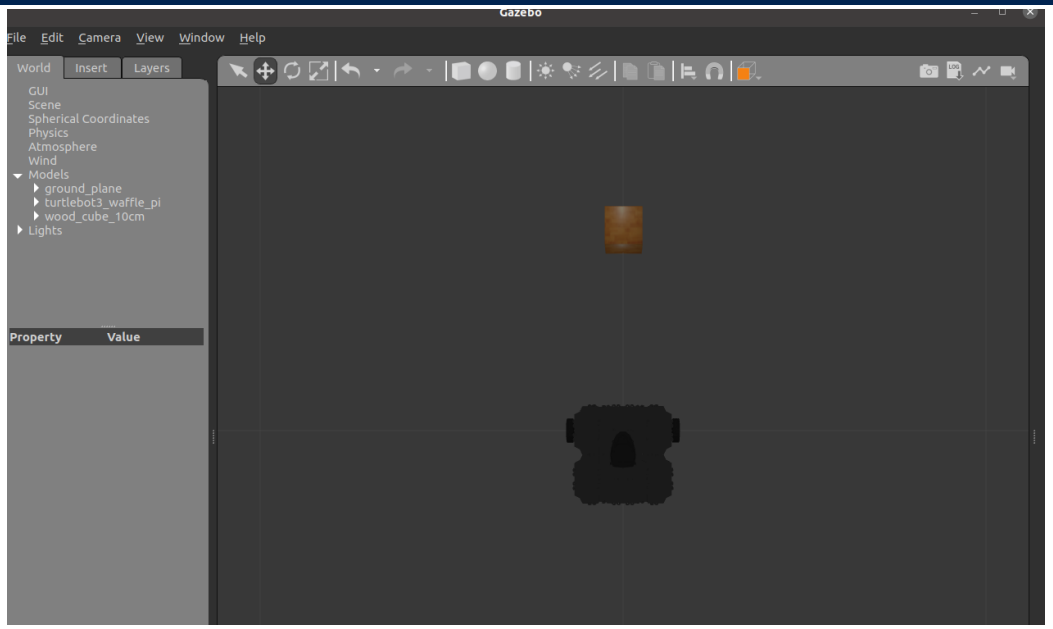
A esquerda, imagem original oriunda da câmera do Turtlebot3 no Gazebo. A direita, a máscara de segmentação. No topo podem ser vistos os valores utilizados para gerar a máscara.



Valores dos limites da máscara de segmentação salvos no arquivo "mask_param.yaml"

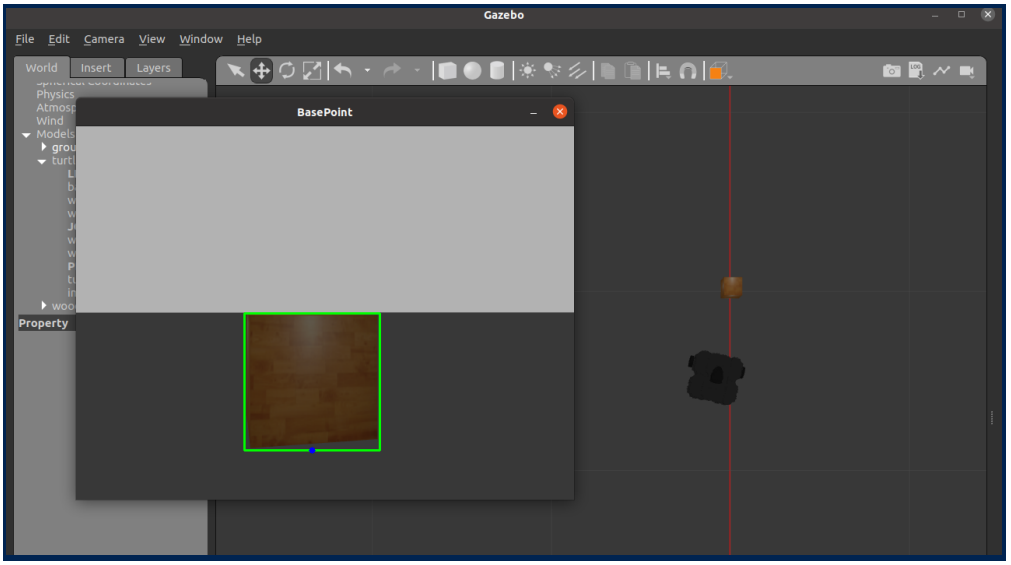
```
#============================================================
#Etapa 7: Inicializando o Gazebo com o cenário 'empty_world' do TurtleBot3 e inserir o objeto (modelo) 'Wood cube 10cm', posicionando o mesmo a uma distÂncia considerável mas sem sair do raio de visão da câmera
$ initros1
$ source catkin_ws/devel/setup.bash
$ export TURTLEBOT3_MODEL=waffle_pi
```

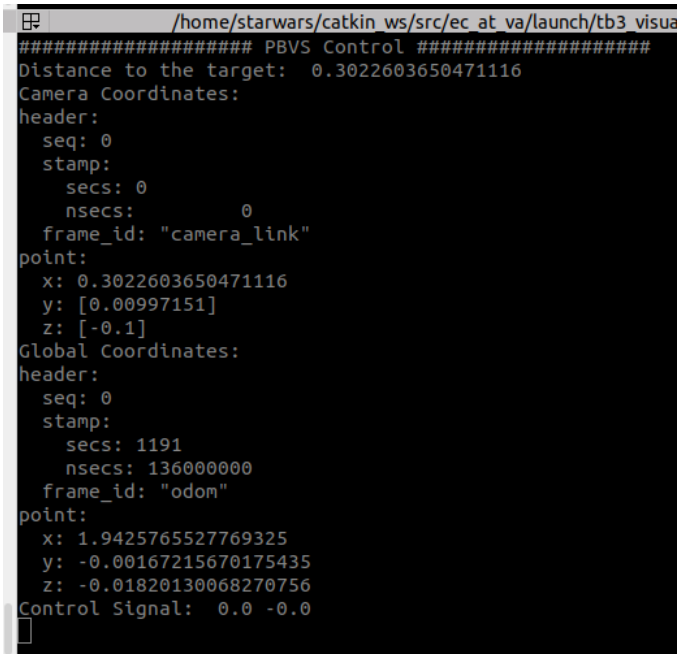
```
$ roslaunch turtlebot3_gazebo turtlebot3_empty_world.lanch
#=================================================================
```



Gazebo com Turtlebot3 modelo 'Waffle PI" e um cubo de 10cm.

```
#=================================================================
#Etapa 8: Inicializar o launch para regulação de postura parcial (30cm de distância do objeto - para isso foi configurado um threshold de 0.37) utilizando a técnica de PBVS
$ initros1
$ source catkin_ws/devel/setup.bash
$ roslaunch ec_at_va tb3_visual_servoing_pbvs.launch
#=================================================================
```





Vídeo de demonstração: https://youtu.be/ZIRpPIw4BBg

2. **O Aprendizado por Reforço é uma técnica de aprendizado de máquina que permite que um agente aprenda a melhor ação a ser tomada em um determinado estado a partir da interação com o ambiente em que está inserido. O objetivo do agente será acumular a maior recompensa possível durante cada época de treinamento. Uma das abordagens de aprendizado é o Deep Q-Learning (DQN), em que uma rede neural para estimar qual a ação que trará a maior recompensa futura durante o treinamento, dado um determinado estado. Com base nesses conceitos e utilizando o material disponibilizado em aula, realize o treinamento de um agente que seja capaz de pousar um drone em uma determinada posição do ambiente, como mostrado na Figura 2. Para isso, a partir dos valores inicialmente configurados, realize o ajuste dos seguintes hiperparâmetros responsáveis pelo processo de aprendizado:**
    - **Fator de desconto (*gamma*);**
    - **Fator de exploração (*epsilon*);**
    - **Épocas de treinamento (*Episodes*);**
    - **Tamanho da rede (neurônios por camada);**

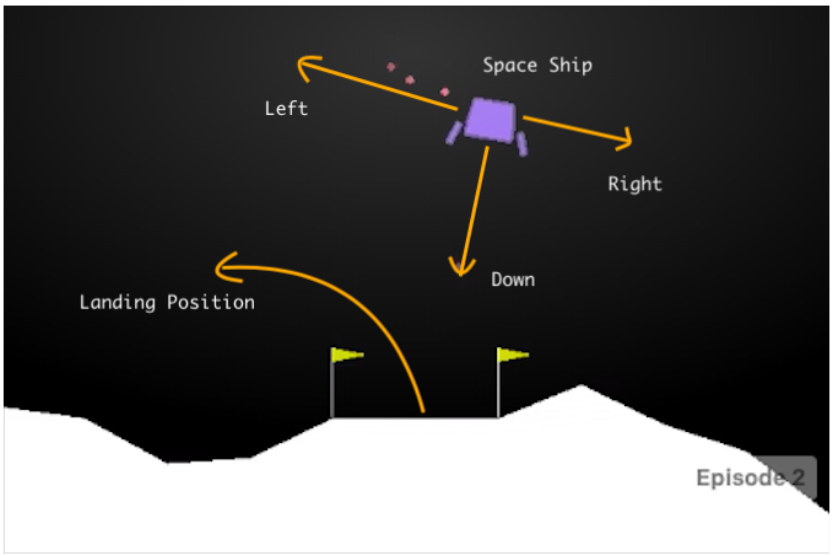


Figura 2 - Simulador Lunar Lander v2 Gym.

Escolha pelo menos um desses parâmetros e faça o seu ajuste para avaliar o impacto de sua variação no processo de aprendizado. Para avaliar o processo de aprendizado, construa um gráfico com as recompensas recebidas durante o treinamento em cada configuração utilizada, comparando a evolução da aprendizagem do agente em cada configuração.

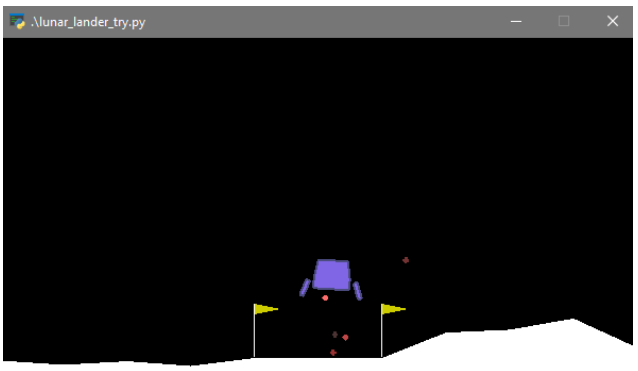
Após o treinamento, avalie o comportamento do agente com cada uma das configurações treinadas, verificando a posição final que o drone atingiu. Para isso, faça cinco rodas de pousos com cada agente e colete a posição final do drone. Para cada agente monte uma tabela com as coordenadas finais em cada pouso, o erro (coordenada desejada – coordenada do pouso) para cada coordenada (x e y), e a média do erro para cada coordenada em cada pouso. Utilizando os dados coletados, faça uma avaliação de qual configuração obteve o melhor resultado, tanto durante o treinamento como durante a inferência, apontando qual é a melhor configuração. Extra: realize a variação de mais de um parâmetro e avalie a interação desses parâmetros e seu impacto para o processo de aprendizado e no processo de inferência.

Resposta:

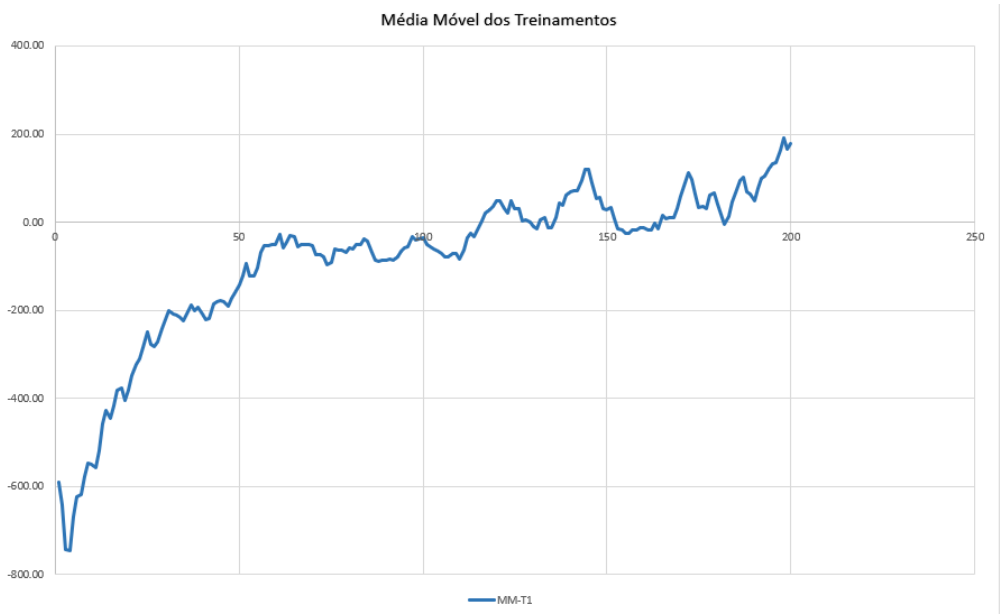
Obs: O número de episódios de teste foram fixados em 5 a fim de simplificar as análises, contudo é sabido que para uma análise mais completa seria necessário mais rodadas tanto da fase de treino quanto de teste.

<u>Treinamento 1 (Base):</u>

Como resultado "**base**", foram utilizados **200 episódios**, **10%** de taxa de exploração [**Epsilon**] (define o quanto o agente irá explorar do ambiente, um valor maior significa que de decisões aleatória - sem ter como base a tabela - serão tomadas), **98%** de fator de desconto [**Gama**] (o agente se preocupará em recompensas de curto prazo) e um espaço amostral [**Batch**] de **32**. Além disso, foram utilizadas **2 camadas de redes neurais**, com **200 "neurônios" cada**.



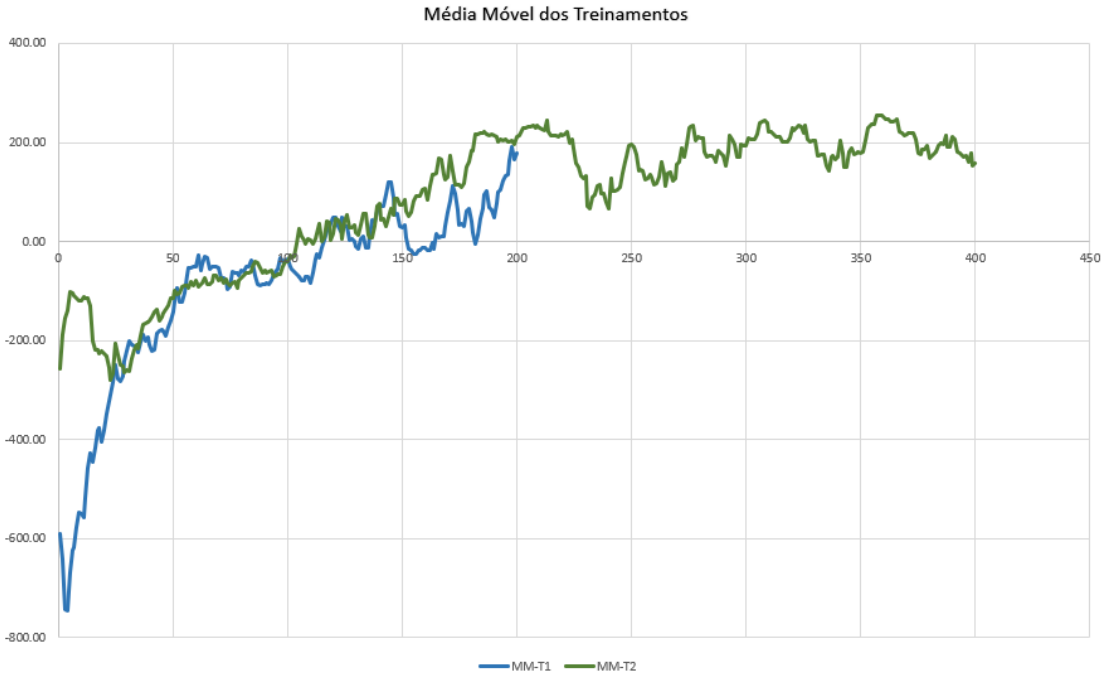
Como resultado, o agente conseguiu ajustar sua posição em relação a Y de forma mais controlada ocasionando em descidas menos abruptas sem contudo concluir o objetivo final (realizar o pouso no local determinado). Sua média de recompensas iniciou de forma bem baixa, o que indica que o agente obteve muitos erros e que começou a indicar aprendizado a partir do episódio 165.



Treinamento 2:

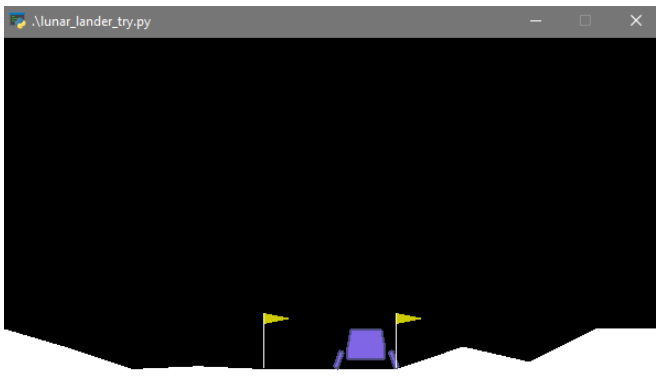
Foram utilizados **400 episódios**, **10%** de taxa de exploração (**Epsilon**), **98%** de fator de desconto (**Gama**), um espaço amostral (**Batch**) de **32**, **2 camadas de redes neurais**, com **400** e **200 "neurônios"** respectivamente. Em comparação com o anterior, temos o dobro de episódios e a primeira camada da rede com o dobro de neurônios.

Como resultado, o agente também passou a realizar tentativas de ajustar seu curso em relação ao eixo X, conseguindo pousar com as duas pernas em 60% das vezes. Sua média de recompensas se manteve positiva a partir do episódio 104 e com poucas oscilações a partir do episódio do 249.



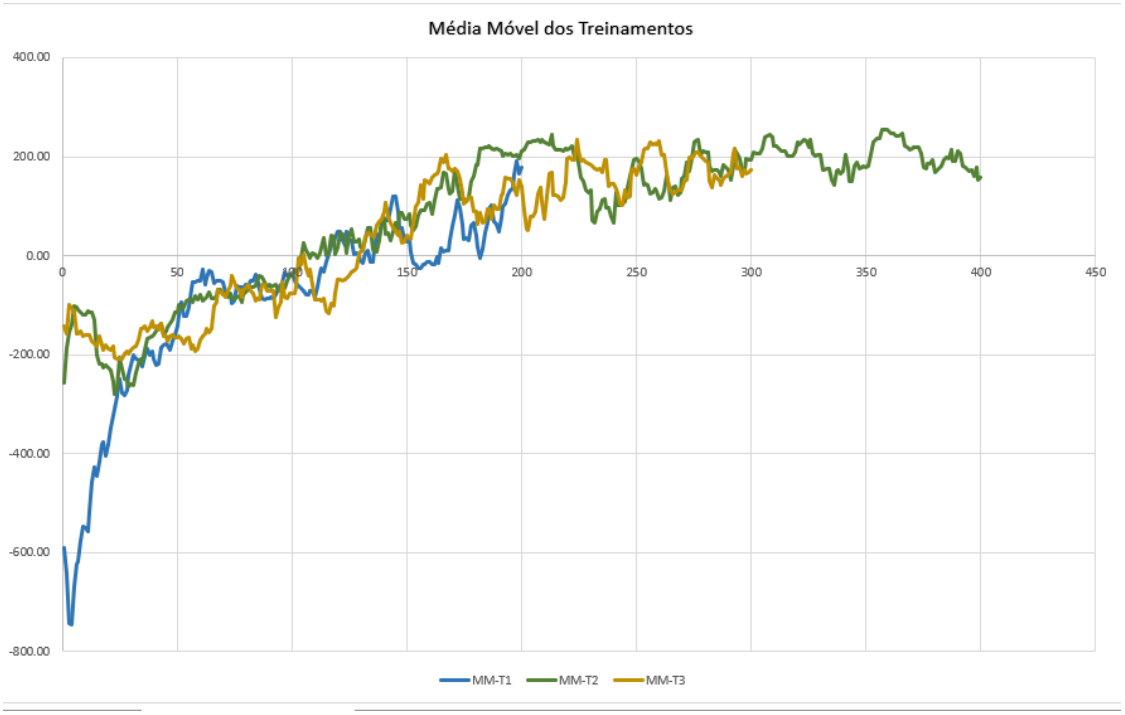
Treinamento 3:

Foram utilizados **300 episódios**, **10%** de taxa de exploração (**Epsilon**), **98%** de fator de desconto (**Gama**), um espaço amostral (**Batch**) de **32**, **2 camadas de redes neurais**, com **400 "neurônios"** cada. Em comparação com o anterior, temos 100 episódios a menos, porém com o dobro de neurônios na segunda camada da rede.



Como resultado, o agente também passou a realizar mais tentativas de ajustar seu curso em relação ao eixo X e ao eixo Y (elevação), conseguindo pousar com a "Perna 1" em 60% das vezes e com a "Perna 2" em 100% das vezes. Sua média de

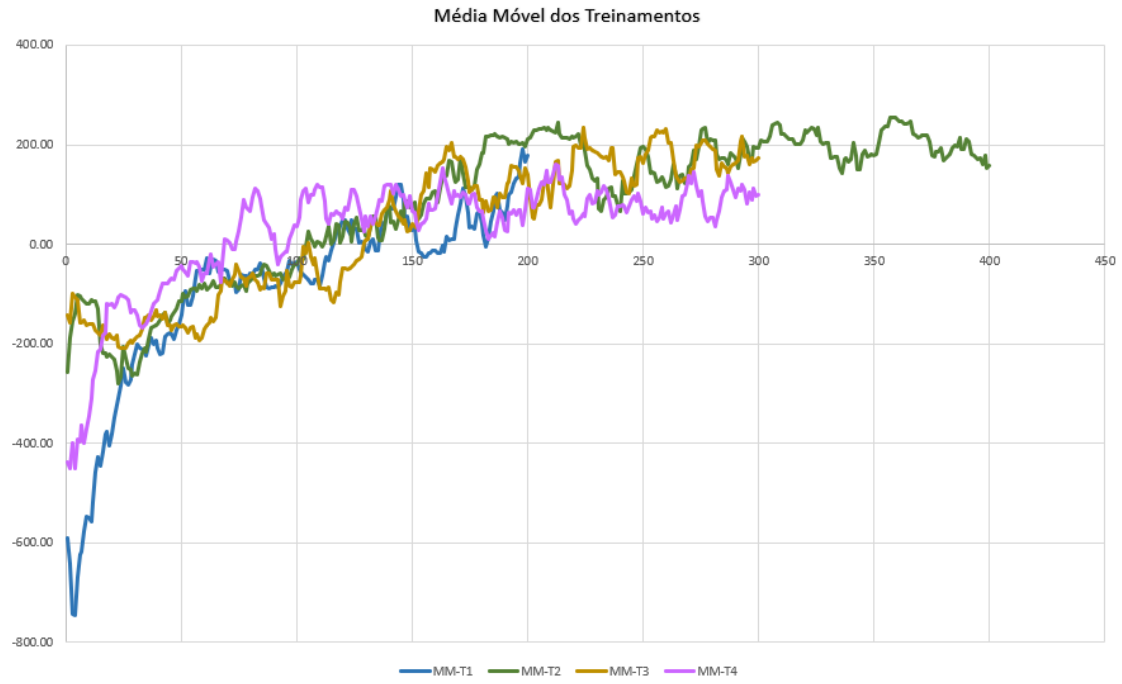
recompensas se manteve positiva a partir do episódio 129, porém com médias menores do que o treinamento anterior até o episódio 226.



<u>Treinamento 4:</u>

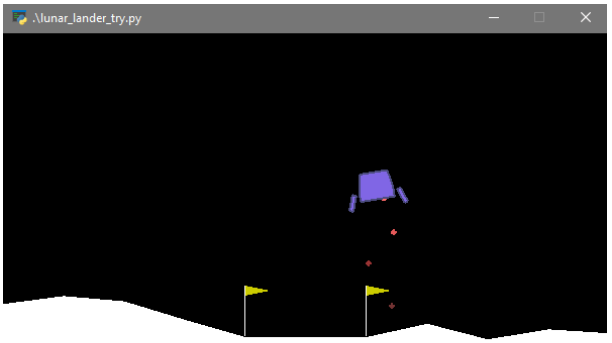
Foram utilizados **300 episódios**, **20%** de taxa de exploração (**Epsilon**), **98%** de fator de desconto (**Gama**), um espaço amostral (**Batch**) de **32, 3 camadas de redes neurais**, com **200 "neurônios"** cada. Em comparação com o anterior, temos 10% a mais de taxa de exploração do ambiente e uma camada a mais de processamento de rede, contudo a soma de neurônios utilizados é menor do que a anterior (atual 600 ao todo, anterior 800 ao todo).

Mesmo com 3 camadas de rede, quando comparado ao “Treinamento 2” (série em verde no gráfico mais abaixo) observa-se que obteve um resultado inferior, que está associado a maior porcentagem da taxa de exploração que foi configurada no “Treinamento 4” e ao número menor de episódios. Conseguiu pousar com a “Perna 1” em 60% das vezes e com a “Perna 2” em 40% das vezes (menor do que quando comparado ao treino anterior). Sua média de recompensas se manteve positiva a partir do episódio 97, porém com médias um pouco menores do que o treinamento anterior.

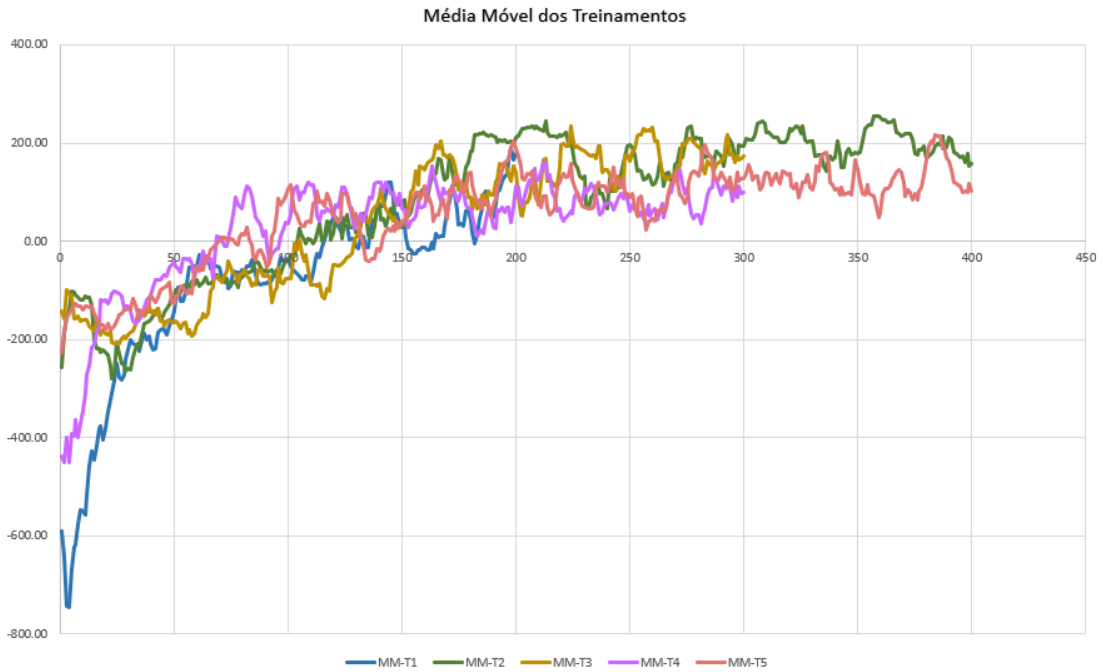


<u>Treinamento 5:</u>

Foram utilizados **400 episódios**, **20%** de taxa de exploração (**Epsilon**), **98%** de fator de desconto (**Gama**), um espaço amostral (**Batch**) de **32, 3 camadas de redes neurais**, com **400 “neurônios”** cada. Em comparação com o anterior, temos 100 episódios a mais e o dobro de neurônios em cada camada.

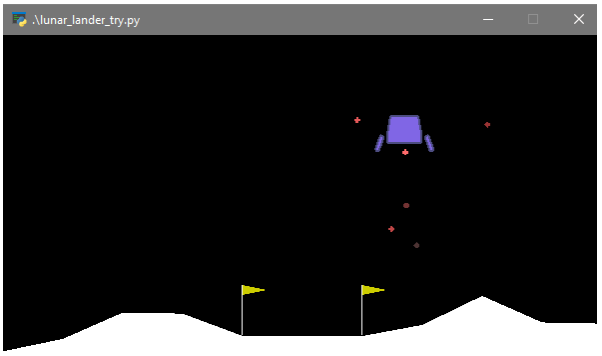


Com dobro de neurônios totais (1200) e o dobro da taxa de exploração (20%), quando comparado ao “Treinamento 2” (série em verde no gráfico mais abaixo) observa-se que obteve um resultado um pouco inferior ainda, que está associado a maior porcentagem da taxa de exploração que foi configurada no “Treinamento 5”. Mas se comparado ao treinamento anterior (série MM-T4 em Lilás) observa-se que houve melhora no desempenho geral. Conseguiu pousar com as duas pernas em 40%. Sua média de recompensas se manteve positiva e estável a partir do episódio 142, ficando próximo ao resultado obtido no treino anterior.

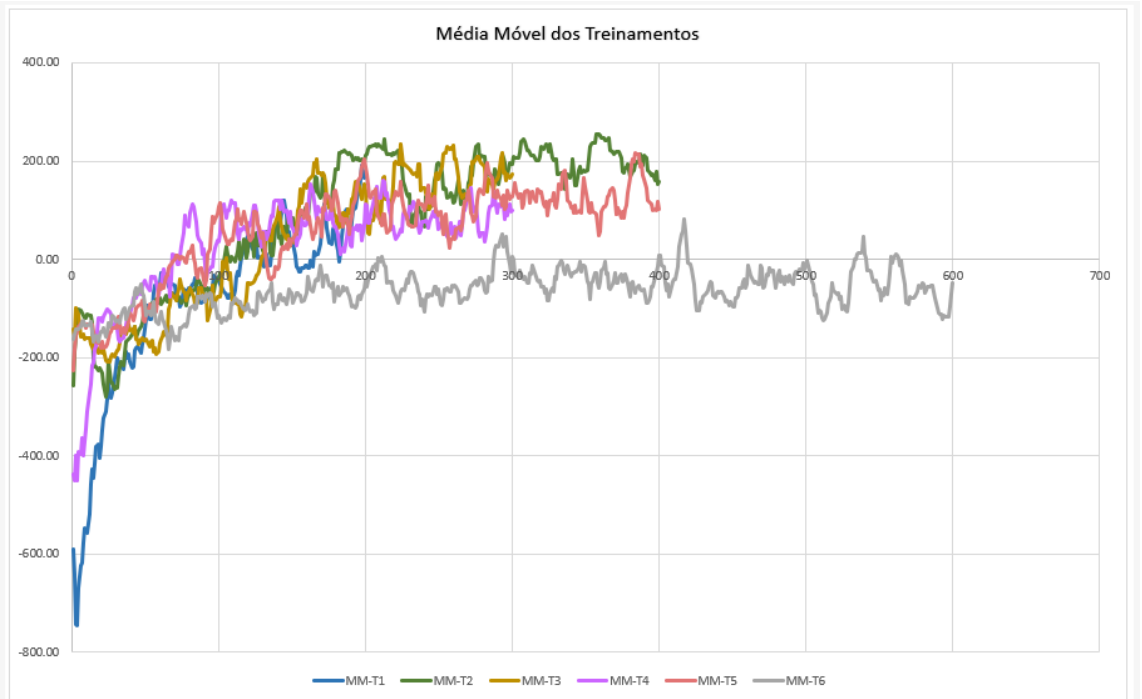


Treinamento 6:

Foram utilizados **600 episódios**, **20%** de taxa de exploração (**Epsilon**), **88%** de fator de desconto (**Gama**), um espaço amostral (**Batch**) de **32, 3 camadas de redes neurais**, com **400 "neurônios"** cada. Em comparação com o anterior, temos 200 episódios a mais e 10% a menos de fator de desconto, o que fará com que o agente pense em recompensas de prazos um pouco maiores.

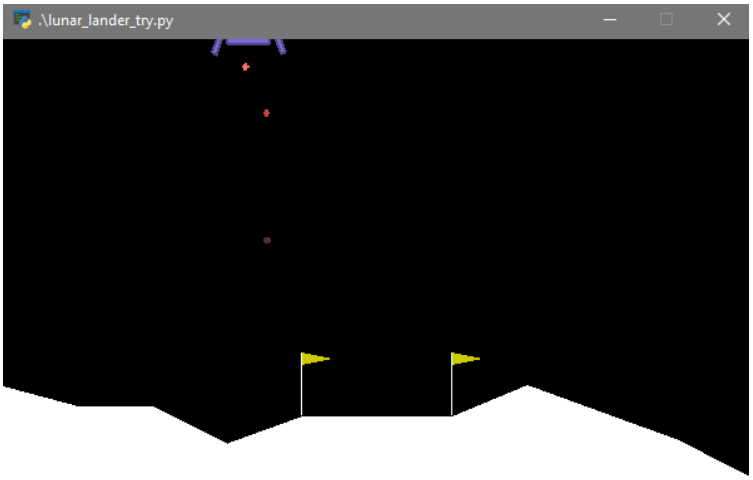


De todos os treinamentos realizados até o momento, este foi o que teve o pior desempenho. O agente demonstrou ter aprendido a não cair e tão somente, em nenhuma das tentativas suas pernas alcançaram o solo.

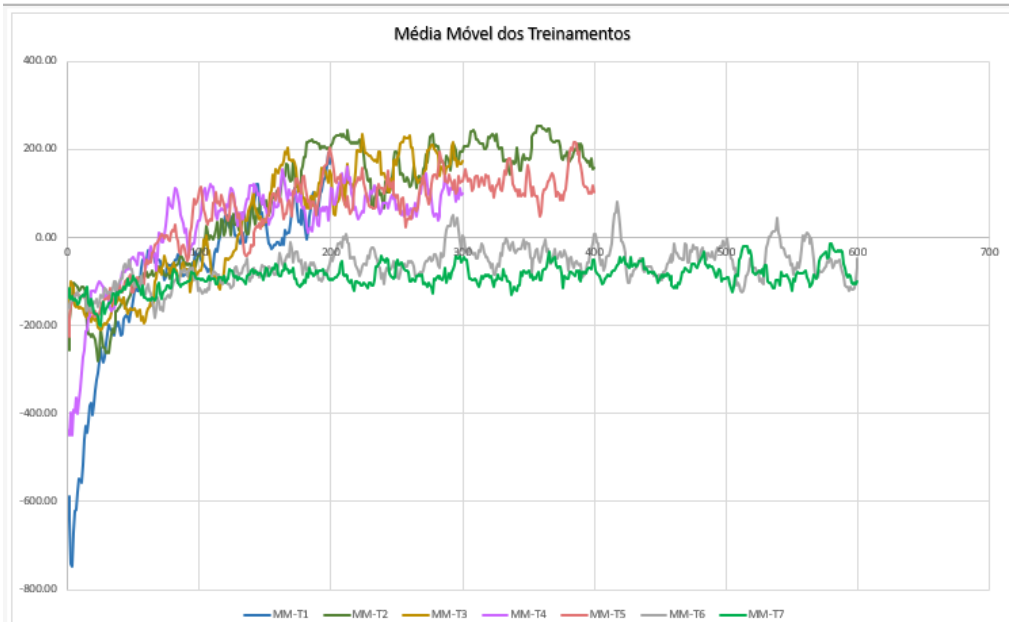


Treinamento 7:

Foram utilizados **600 episódios**, **20%** de taxa de exploração (**Epsilon**), **88%** de fator de desconto (**Gama**), um espaço amostral (**Batch**) de **64, 3 camadas de redes neurais**, com **400 "neurônios"** cada. Em comparação com o treinamento anterior, temos o dobro de espaço amostral.

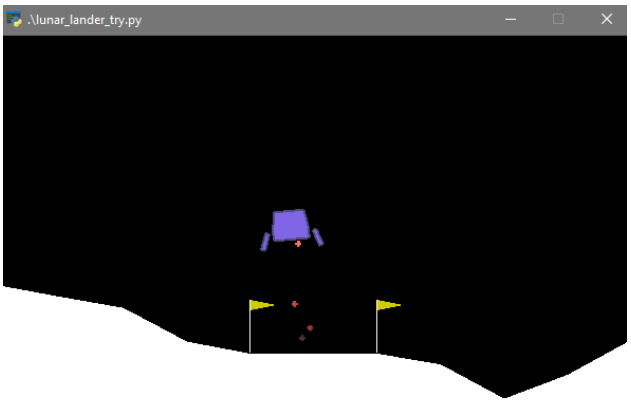


Também obteve resultados abaixo das médias anteriores, sendo mais baixo do que o treinamento anterior. O agente demonstra ter aprendido somente a não cair. Não houveram episódios onde a recompensa final obteve valores positivos. O que nos leva a crer que um aumento da taxa de exploração aliada a diminuição do fator de desconto gera mais ruído no treinamento.

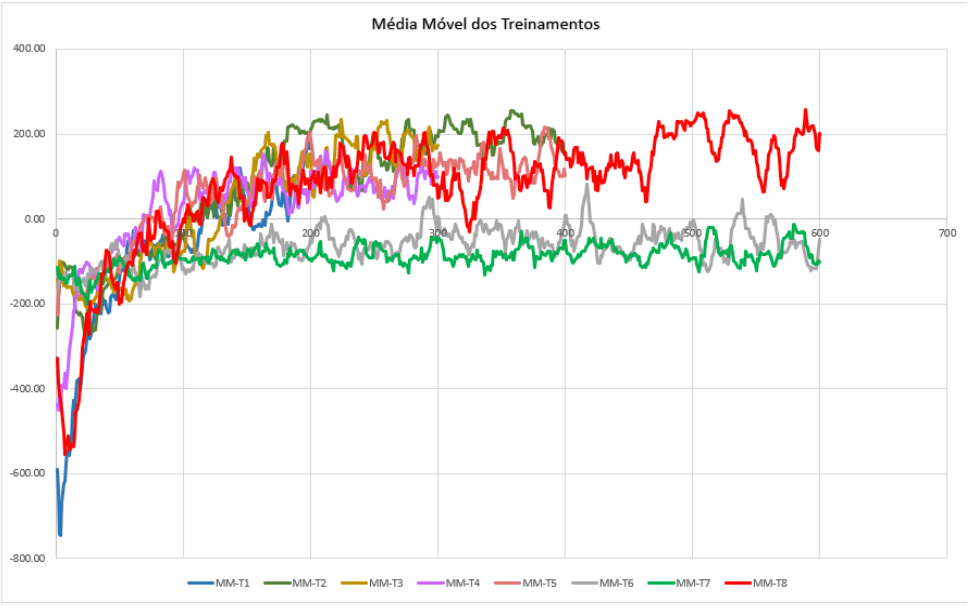


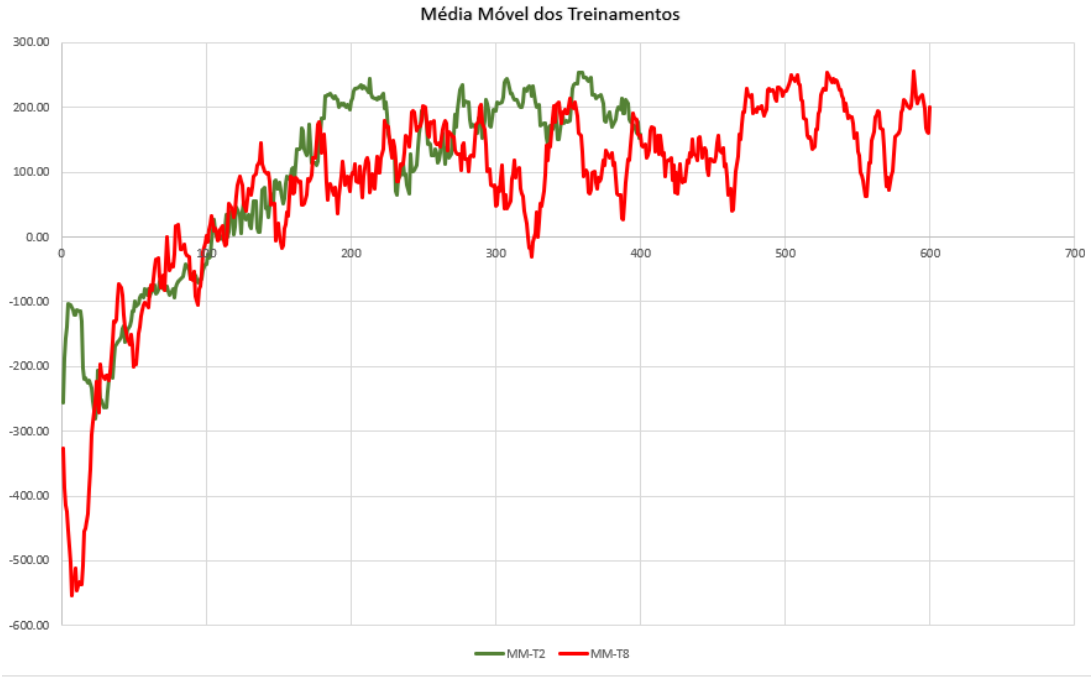
Treinamento 8:

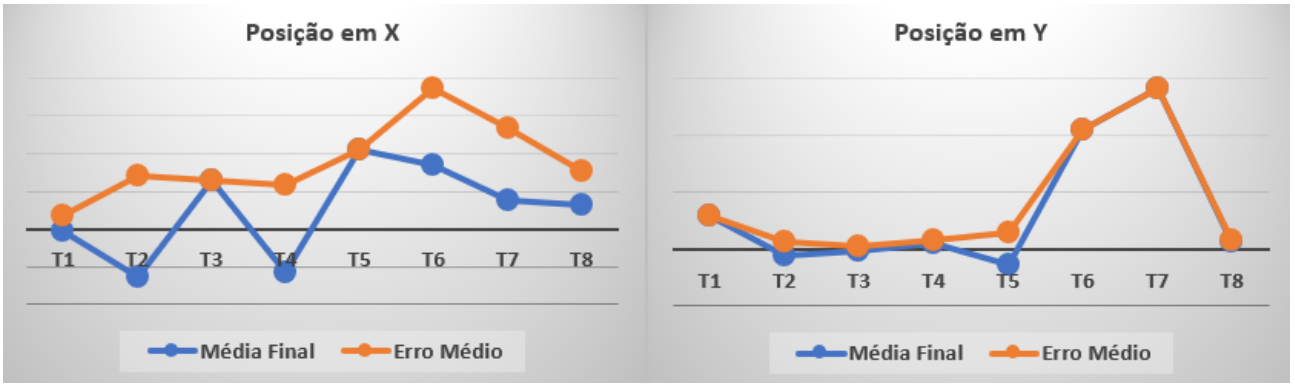
Como o melhor resultado obtido foi no “Treinamento 2”, então os hiperparâmetros deste treinamento foram ajustados de forma aumentar a capacidade da rede e avaliar se assim teríamos resultados ainda mais satisfatórios. Foram utilizados **600 episódios**, **10%** de taxa de exploração (**Epsilon**), **98%** de fator de desconto (**Gama**), um espaço amostral (**Batch**) de **32, 4 camadas de redes neurais**, com **200**, **400, 400 e 200 “neurônios”** respectivamente. Em comparação com o “Treinamento 2“ temos 200 episódios a mais, o dobro de camadas de processamento, bem como o dobro de “neurônios” totais.

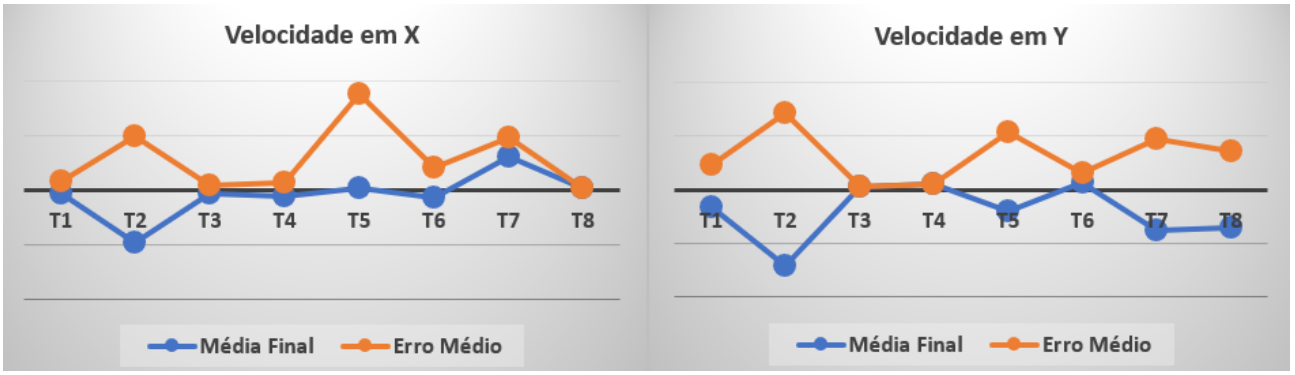


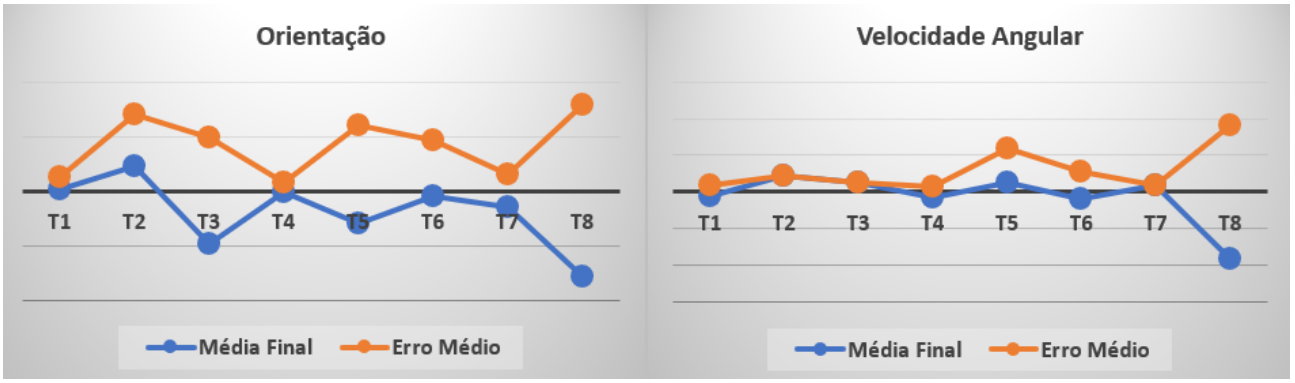
O agente apresentou comportamento similar ao “Treinamento 2”, tendo realizado tentativas de ajuste de posição nos eixos X e Y de forma atingir o objetivo final.



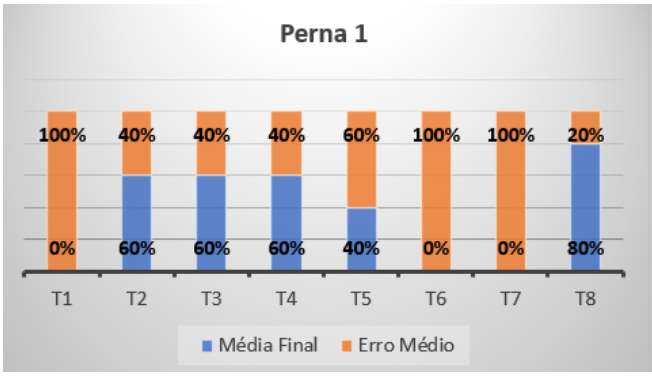

Média Móvel dos Treinamentos
300.00
200.00
100.00
0.00
-100.00
-200.00
-300.00
-400.00
-500.00
-600.00
0
100
200
300
400
500
600
700
MM-T2
MM-T8

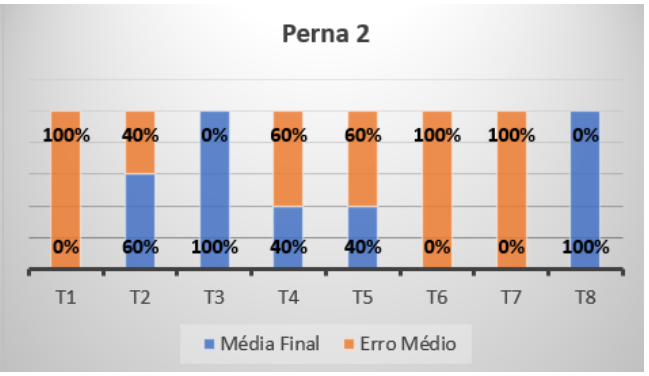

Posição em X
T1
T2
T3
T4
T5
T6
T7
T8
Média Final
Erro Médio
Posição em Y
T1
T2
T3
T4
T5
T6
T7
T8
Média Final
Erro Médio


Velocidade em X
T1
T2
T3
T4
T5
T6
T7
T8
Média Final
Erro Médio
Velocidade em Y
T1
T2
T3
T4
T5
T6
T7
T8
Média Final
Erro Médio


Orientação
T1 T2 T3 T4 T5 T6 T7 T8
Média Final
Erro Médio

Velocidade Angular
T1 T2 T3 T4 T5 T6 T7 T8
Média Final
Erro Médio


Perna 1
100% 40% 40% 40% 60% 100% 100% 20%
0% 60% 60% 60% 40% 0% 0% 80%
T1 T2 T3 T4 T5 T6 T7 T8
Média Final
Erro Médio


Perna 2
100% 40% 0% 60% 60% 100% 100% 0%
0% 60% 100% 40% 40% 0% 0% 100%
T1 T2 T3 T4 T5 T6 T7 T8
Média Final
Erro Médio

---------- FIM ----------