Contents

Chapter 1

Reactive movements

4_20_1042

Trainer: PPO

batch size	32	beta	5.0e-3
buffer size	256	hidden units	256
learning rate	5.0e-4	max steps	5.0e5
normalize	false	num layers	1
summary freq	100	time horizon	5
extrinsic strength	1.0	extrinsic gamma	0.9

Camera Sensor: 40x40
Grayscale: false
Observation space size = 25 (eje X)
Action space size = 3

Episode steps = 3000
Total steps = 51400
Time = 1000s

It may be necessary to decrease the action vector to only 1 (whichever it is; the clicks are treated as automatic). Improve coherence of movement.

4_20_1320

Trainer: PPO

batch size	32	beta	5.0e-3
buffer size	256	hidden units	256
learning rate	5.0e-4	max steps	5.0e5
normalize	false	num layers	1
summary freq	**1000**	time horizon	5
extrinsic strength	1.0	extrinsic gamma	0.9

Camera Sensor: 40x40
Grayscale: false
Observation space size = 25 (eje X)
Action space size = 1

Episode steps = 3000
Total steps = 60000
Time = 1200s

It needs more training, or different parameters. It looks like it's moving in the direction but it's still "vibrating" a lot. It may be necessary to think about raytracing instead of a camera.

4_20_1357

Trainer: PPO

batch size	32	beta	5.0e-3
buffer size	256	hidden units	256
learning rate	5.0e-4	**max steps**	**5.0e6**
normalize	false	num layers	1
summary freq	1000	time horizon	5
extrinsic strength	1.0	extrinsic gamma	0.9

Camera Sensor: 40x40
Grayscale: false
Observation space size = 25 (eje X)
Action space size = 1

Episode steps = 3000
Total steps = 170000
Time = 2200s (+)

Same agent as 4_20_1320 (checkpoint)

4_20_1440

Trainer: PPO

batch size	32	beta	5.0e-3
buffer size	256	hidden units	256
learning rate	5.0e-4	max steps	5.0e6
normalize	false	num layers	1
summary freq	1000	time horizon	5
extrinsic strength	1.0	extrinsic gamma	0.9

Camera Sensor: 40x40
Grayscale: false
Observation space size = 25 (eje X)
Action space size = 1

Episode steps = 3000
Total steps = 500000
Time = 6770s (+)

It seems to have stabilized at a reward of 12,500 - 12,600.
In reality it stays practically paralyzed, moving very little (almost nothing).

4_21_1232

Trainer: PPO

batch size	32	beta	5.0e-3
buffer size	256	hidden units	256
learning rate	5.0e-4	max steps	5.0e6
normalize	false	num layers	1
summary freq	1000	time horizon	5
extrinsic strength	1.0	extrinsic gamma	0.9

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 25 (eje X)
Action space size = 1

Episode steps = 3000
Total steps = 350000
Time = 7530s

The Bot has been improved with reaction time and more speed. More demanding reward conditions.

The graph seems to show good behavior. Keeps training.

4_21_1444

Trainer: PPO

batch size	32	beta	5.0e-3
buffer size	256	hidden units	256
learning rate	5.0e-4	max steps	5.0e6
normalize	false	num layers	1
summary freq	1000	time horizon	5
extrinsic strength	1.0	extrinsic gamma	0.9

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 25 (eje X)
Action space size = 1

Episode steps = 3000
Total steps = -
Time = -

Same agent as 4_21_1232.
It doesn't work properly, but if you start the bot the graph works as it should. On the other hand, although the graph follows the same patterns, it stagnates at an amount below (maybe the movement limit). It also causes untrained cases to occur.

4_21_1731

Trainer: PPO

batch size	32	beta	5.0e-3
buffer size	256	hidden units	256
learning rate	5.0e-4	max steps	5.0e6
normalize	false	num layers	1
summary freq	1000	time horizon	5
extrinsic strength	1.0	extrinsic gamma	0.9

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 25 (eje X)
Action space size = 1

Episode steps = 3000
Total steps = 310000
Time = 6700s

Fixed that the agent reads half of the movement value (bug), instead of what the bot actually returned.

I think it uses the previous movements as "cheat sheet" to know what to do, maybe you have to detect the previous movements of the agent to learn or make the agent to move, and the bot to correct.

-I had to interrupt the training because the reaction was coming out of the graph (now it's -2 to 2, instead of 1).

-Another interruption, the vector action only returns normalized, so you have to change the value that happens to the camera (not the other way around).

The graph fits perfectly in the training, but I think it's because you can see what the real bot has done in the previous movement. It may be necessary to add curiosity rewards in following trainings.

The model seemed to make a good default move, even with a change of direction, but then it tends to turn at full speed without stopping. It should certainly stop reading what the bot does in the last move.

4_22_1000

Trainer: PPO

batch size	32	beta	5.0e-3
buffer size	256	hidden units	256
learning rate	5.0e-4	max steps	5.0e6
normalize	false	num layers	1
summary freq	1000	time horizon	5
extrinsic strength	1.0	**extrinsic gamma**	**0.95**

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 25 (eje X)
Action space size = 1

Episode steps = 2000
Total steps = 41000
Time = 900s

The previous movements have been changed to those of the agent.
It tends towards chaos.

4_22_1031

Trainer: PPO

batch size	32	beta	5.0e-3
buffer size	256	hidden units	256
learning rate	5.0e-4	max steps	5.0e6
normalize	false	num layers	1
summary freq	1000	time horizon	5
extrinsic strength	1.0	**extrinsic gamma**	**0.9**

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 0
Action space size = 1

Episode steps = 3000
Total steps = 185000
Time = 4000s

All observation other than the camera has been removed. The bot reacts in time 0 now. An idle address has also been removed.

It was on the right track but tends to go into chaos after 1h.

4_22_1143

Trainer: PPO

batch size	32	beta	5.0e-3
buffer size	256	hidden units	256
learning rate	**2.5e-4**	max steps	5.0e6
normalize	false	num layers	1
summary freq	1000	time horizon	5
extrinsic strength	1.0	extrinsic gamma	0.9
curiosity strength	**0.01**	**curiosity gamma**	**0.99**
curiosity encoding size	**256**		

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 0
Action space size = 1

Episode steps = 3000

Total steps = 180000
Time = 4100s

Decreased idle randomness and increased acceptable range.

The graph fits well, although with some noise (perhaps out of curiosity), but it also tends to hit the planes that appear on the opposite side (this didn't happen before, it only reacted well to those that appeared in front). In viewport it also seems very similar, and the noise is not so noticeable.

Saved as a checkpoint.

–Note: the behaviour is a bit different because when training the screen is displayed in a 16:9 ratio (instead of 1:1 from the editor), that makes the camera react a bit differently.

4_22_1319

Trainer: PPO

batch size	32	beta	5.0e-3
buffer size	256	hidden units	256
learning rate	2.5e-4	max steps	5.0e6
normalize	false	num layers	1
summary freq	1000	time horizon	5
extrinsic strength	1.0	extrinsic gamma	0.9
curiosity strength	0.01	curiosity gamma	0.99
curiosity encoding size	256		

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 0
Action space size = 1

Episode steps = 3000
Total steps = 256000
Time = 1500s (+)

Correct the proportion in the build. Actually, as the render target's camera goes out from the agent's one, it doesn't cause problems. However the isVisible should be changed in relation to that camera.

It tends again to chaos.

4_22_1352

Trainer: PPO

batch size	32	beta	5.0e-3
buffer size	256	hidden units	256
learning rate	**1.0e-4**	max steps	5.0e6
normalize	false	num layers	1
summary freq	1000	time horizon	5
extrinsic strength	1.0	extrinsic gamma	0.9

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 0
Action space size = 1

Episode steps = 3000
Total steps = 373000
Time = 8050s

Before training the reward is increased by being in range.

VERY slow training, consider setting a "learning rate schedule" so that the learning rate goes from more to less (but it may already be by default, you would have to decrease the number of training steps).

It seems more or less stable (like the previous one achieved).

4_22_1615

Trainer: PPO

batch size	32	beta	5.0e-3
buffer size	256	**hidden units**	**128**
learning rate	**1.0e-3**	**learning rate schedule**	**linear**
max steps	**5.0e4**	normalize	false
num layers	**2**	summary freq	1000
time horizon	5	extrinsic strength	1.0
extrinsic gamma	0.9		

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 0
Action space size = 1

Episode steps = 3000
Total steps = 50000
Time = 1075s

Learning rate added (so that it can be changed explicitly) The steps are now determined. The structure of the network has also been changed.

It has reached a reward of -0.313, maybe for increasing the number of layers.

4_22_1640

Trainer: PPO

batch size	32	beta	5.0e-3
buffer size	256	**hidden units**	**256**
learning rate	1.0e-3	learning rate schedule	linear
max steps	**1.0e5**	normalize	false
num layers	**1**	summary freq	1000
time horizon	5	extrinsic strength	1.0
extrinsic gamma	0.9		

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 0
Action space size = 1

Episode steps = 3000
Total steps = 100000
Time = 2160s

It may be necessary to lower the initial learning rate to increase the effectiveness of the training. In the first 50,000 steps it makes very little progress (from -.524 to -.415).

In the end it reaches -.353.

4_22_1719

Trainer: PPO

batch size	**1024**	beta	5.0e-3
buffer size	**4096**	hidden units	256
learning rate	**2.0e-4**	**learning rate schedule**	**constant**
max steps	**2.0e5**	normalize	false
num epoch	**5**	num layers	1
summary freq	1000	**time horizon**	**8**
extrinsic strength	1.0	extrinsic gamma	0.9

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 0
Action space size = 1

Episode steps = 3000
Total steps = 25000
Time = 540s

Several changes to adapt the network to continuous space. Constant Learning Rate to check when it converges (or diverges) Does not converge after 25000 steps.

4_22_1754

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	4096	hidden units	256
learning rate	**1.0e-4**	learning rate schedule	constant
max steps	**2.0e6**	normalize	false
num epoch	**3**	num layers	1
summary freq	1000	**time horizon**	**32**
extrinsic strength	1.0	extrinsic gamma	0.9

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 0
Action space size = 1

Episode steps = 3000
Total steps = 115000
Time = 2420s

Long training to try to get it stabilized.
–It doesn't stabilize after 100,000 steps, I think it has a bigger problem with how the rewards are given. I'll turn them into linear.

4_22_1842

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	4096	hidden units	256
learning rate	**5.0e-4**	**learning rate schedule**	**linear**
max steps	2.0e6	normalize	false
num epoch	3	num layers	1
summary freq	1000	time horizon	32
extrinsic strength	1.0	extrinsic gamma	0.9

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 0
Action space size = 1

Episode steps = 3000
Total steps = 61000
Time = 1200s

It shows no sign of stabilizing, so the range of success may still be very demanding.

To save time in converging to the original bot function, I have thought to use learning curriculum that determines the thresholds of the rewards, making them smaller and smaller.

4_23_1049

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	4096	hidden units	256
learning rate	5.0e-4	learning rate schedule	linear
max steps	2.0e6	normalize	false
num epoch	3	num layers	1
summary freq	1000	time horizon	32
extrinsic strength	1.0	extrinsic gamma	0.9

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 0
Action space size = 1

Episode steps = 3000
Total steps = 116000
Time = 2400s

Training with curriculum. Still having trouble converging.

4_23_1246

Trainer: PPO

batch size	**32**	beta	5.0e-3
buffer size	**256**	hidden units	256
learning rate	**3.0e-4**	learning rate schedule	linear
max steps	**1.0e6**	normalize	false
num epoch	3	num layers	1
summary freq	1000	**time horizon**	**5**
extrinsic strength	1.0	extrinsic gamma	0.9

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 0
Action space size = 1

Episode steps = 2000
Total steps = 394000
Time = 8500s

Increased the minimum reward to pass the level. Changes in variables until a stable one is achieved.

It manages to stabilize the idle movement (after quite a while) but doesn't do impulses well.

4_23_1551

Trainer: PPO

batch size	**1024**	beta	5.0e-3
buffer size	**4096**	hidden units	256
learning rate	**5.0e-4**	learning rate schedule	linear
max steps	1.0e6	normalize	false
num epoch	3	num layers	1
summary freq	1000	**time horizon**	**8**
extrinsic strength	1.0	**extrinsic gamma**	**0.8**
curiosity strength	**0.05**	**curiosity gamma**	**0.9**
curiosity encoding size	**128**		

Render Target Sensor: 64x64
Grayscale: false

Observation space size = 0
Action space size = 1

Episode steps = 2000
Total steps = 453000
Time = 7300s

Added penalty for out of range (beyond maximumRange) and curiosity. Also the extrinsic gamma is lowered to make the present (at this time) count more.

Reaches level 2 with 0.4 reward but shows no improvement. The reward range may need to be increased.

4_23_1915

Trainer: PPO

batch size	1024	**beta**	**5.0e-4**
buffer size	4096	hidden units	256
learning rate	**1.0e-3**	learning rate schedule	linear
max steps	1.0e6	normalize	false
num epoch	3	**num layers**	**2**
summary freq	1000	**time horizon**	**2048**
extrinsic strength	1.0	**extrinsic gamma**	**0.9**
curiosity strength	0.05	curiosity gamma	0.9
curiosity encoding size	128		

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 0
Action space size = 1

Episode steps = 2000
Total steps = 760000
Time = 16700s

Training by watching tensorboard. Test with 2 layers, it's probably slower.

The training is too slow, although the graphs seem to have correct progression. The cause may be the form of the reward function or the fact that it had a higher range before (up to -120).

4_24_1026

Trainer: PPO

batch size	**32**	**beta**	**5.0e-3**
buffer size	**256**	hidden units	256
learning rate	**2.5e-4**	learning rate schedule	linear
max steps	**5.0e6**	normalize	false
num epoch	3	**num layers**	**1**
summary freq	1000	**time horizon**	**5**
extrinsic strength	1.0	extrinsic gamma	0.9
curiosity strength	**0.01**	**curiosity gamma**	**0.99**
curiosity encoding size	**256**		

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 0
Action space size = 1

Episode steps = 3000
Total steps = 112000
Time = 2400s

Training with old reward functions (TR=0.005), and with the old parameters In 1500-2000s it starts to get reasonably close. The graphs don't show any strange behaviour, but the rewards are always negative (the network should break in 180k-200k steps, now it is at 100k). The model is correct.

4_24_1600

Trainer: PPO

batch size	32	beta	5.0e-3
buffer size	256	hidden units	256
learning rate	2.5e-4	learning rate schedule	linear
max steps	5.0e6	normalize	false
num epoch	3	num layers	1
summary freq	1000	time horizon	5
extrinsic strength	1.0	extrinsic gamma	0.9
curiosity strength	0.01	curiosity gamma	0.99
curiosity encoding size	256		

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 0
Action space size = 1

Episode steps = 3000
Total steps = 435000
Time = 10000s

Training with the same parameters and without curriculum (with TR=0.01, MR=1). Using linear rewards but multiplied by 40.

From 80000 steps or a little earlier you can see that it adapts to the form (except for some transitions between jumps).

At 150000 steps it almost acts like one of the previous models: it goes from -16 to -2 reward and the graphs don't show anything strange.

At 190000 steps first positive reward (0.340).

At 275,000 steps the idle stages are virtually out of noise. However, it can be seen that the network does not fit well when there are more than 2 targets at once, this may be because of how the bot algorithm prioritizes them and the fact that the network still has no memory. The reward is 3.

Afterwards, the reward increases steadily because it adapts to the idle with more precision, but the more time goes by the less I am convinced about how it adapts to some impulses (it may be due to underfitting, since it doesn't have the memory installed yet either in image or in movements).

Putting it in Unity is less convincing because it doesn't follow the same movement if it's executed autonomously (it's smoothed out even more, the impulses are slower).

In the following tests: increase the learning-rate and put the values normalized; in another one increase even more the multiplication factor. Changing the shape of the function may improve (or worsen) the learning curve. After the 2 tests, it would be convenient to put the movement memory back (25 previous values).

4_24_1935

Trainer: PPO

batch size	32	beta	5.0e-3
buffer size	256	hidden units	256
learning rate	**1.0e-3**	learning rate schedule	linear
max steps	5.0e6	normalize	false
num epoch	3	num layers	1
summary freq	1000	time horizon	5
extrinsic strength	1.0	extrinsic gamma	0.9
curiosity strength	0.01	curiosity gamma	0.99
curiosity encoding size	256		

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 0

Action space size = 1

Episode steps = 3000
Total steps = 190000
Time = 4300s

Standardized rewards, but learning rate multiplied by 40.

1.0e-2 is too much learning rate, it starts at the extremes. 5.0e-3 too. 2.5e-3 seems stable at the start but becomes unsettled on the first pass.

1.0e-3 seems to be the stable maximum, which is 4 times more than the previous one.

It doesn't seem to be as effective, in 100000 steps it doesn't catch the jumps well, even if the reward increases from -0.43 to -0.2

4_24_2130

Trainer: PPO

batch size	32	beta	5.0e-3
buffer size	256	hidden units	256
learning rate	**2.5e-4**	learning rate schedule	linear
max steps	5.0e6	normalize	false
num epoch	3	num layers	1
summary freq	1000	time horizon	5
extrinsic strength	1.0	extrinsic gamma	0.9
curiosity strength	0.01	curiosity gamma	0.99
curiosity encoding size	256		

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 0
Action space size = 1

Episode steps = 3000
Total steps = 366000
Time = 8400s

Reward multiplied by 80, parameters of 4_24_1600.

Similar results are obtained as when multiplied by 40.

In this case you get to 0 at 196000, and to -4 at about 150000 (practically the same proportion).

It would be advisable to adjust the learning curriculum again with these rewards (or those of *40) and check whether or not it is faster. Then add the movement memory again.

4_25_1915

Trainer: PPO

batch size	32	beta	5.0e-3
buffer size	256	hidden units	256
learning rate	2.5e-4	learning rate schedule	linear
max steps	5.0e6	normalize	false
num epoch	3	num layers	1
summary freq	1000	time horizon	5
extrinsic strength	1.0	extrinsic gamma	0.9
curiosity strength	0.01	curiosity gamma	0.99
curiosity encoding size	256		

Render Target Sensor: 64x64

Grayscale: false

Observation space size = 0

Action space size = 1

Episode steps = 3000

Total steps = 305000

Time = 7000s

Training as 4_24_1600 but with curriculum (adapted, TR from 0.5 to 0.01) It should approach the form in 80000 steps and have positive reward (in the last level) at 190000 to equal the training without curriculum.

In the end it seems to work similarly with less restrictive parameters, but as the minimum conditions are unbalanced it has not passed the first level (behaving almost the same). A next training will be done by setting the 0 of limits to overcome levels.

Probably the curriculum can be used to increase the restrictions, but this requires more observations.

The parameter of curiosity is of no use to me.

4_26_1421

Trainer: PPO

batch size	32	beta	5.0e-3
buffer size	256	hidden units	256
learning rate	2.5e-4	learning rate schedule	linear
max steps	5.0e6	normalize	false
num epoch	3	num layers	1
summary freq	1000	time horizon	5
extrinsic strength	1.0	extrinsic gamma	0.9
curiosity strength	0.01	curiosity gamma	0.99
curiosity encoding size	256		

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 25
Action space size = 1

Episode steps = 3000
Total steps = 573000
Time = 13200s

Training without curriculum, with the 25 added observations (from the agent). The change of direction has also been activated, to see if it learns the context.

The reward is stagnant over -10 from the 160000 steps. The graph shows how it oscillates over the center. For future models it may be necessary to increase the complexity of the training and the network. Another way is to make the network in control and the bot indicate how to act.

4_26_1815

Trainer: PPO

batch size	**1024**	beta	5.0e-3
buffer size	**8192**	hidden units	256
learning rate	2.5e-4	learning rate schedule	linear
max steps	5.0e6	normalize	false
num epoch	**5**	**num layers**	**2**
summary freq	1000	**time horizon**	**256**
extrinsic strength	1.0	extrinsic gamma	0.9

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 25
Action space size = 1

Episode steps = 3000
Total steps = 763000
Time = 16750s

It should improve accuracy, but be much slower.

Idea: put as previous values of movement the average between the bot and the agent.

The graph recognizes impulses very well, but the idle oscillates between 0 with quite a lot of noise. This must be because the network doesn't remember the previous images, and doesn't distinguish when to change direction. The growth is much slower and it stagnates again on the same level of reward. This could be solved by pre-recording the image or simplifying the algorithm in cases where more than one target appears.

4_27_1035

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	8192	hidden units	256
learning rate	2.5e-4	learning rate schedule	linear
max steps	5.0e6	normalize	false
num epoch	5	num layers	2
summary freq	1000	time horizon	256
extrinsic strength	1.0	extrinsic gamma	0.9

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 25
Action space size = 1

Episode steps = 3000
Total steps = 132000
Time = 2900s

Improved bot so that it always chooses the closest target, at the moment the change of direction is removed. In this one we are going to use curriculum to see if it improves the learning speed.

The curriculum doesn't seem effective, after almost 3000s it has only reached -14 of reward (from the easy level).

4_27_1123

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	8192	hidden units	256
learning rate	2.5e-4	learning rate schedule	linear
max steps	5.0e6	normalize	false
num epoch	5	num layers	2
summary freq	1000	time horizon	256
extrinsic strength	1.0	extrinsic gamma	0.9

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 25
Action space size = 1

Episode steps = 3000
Total steps = 366000
Time = 8000s

Same model as before but without curriculum. Similar result, it stagnates at -10.

4_27_1339

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	8192	hidden units	256
learning rate	2.5e-4	learning rate schedule	linear
max steps	5.0e6	normalize	false
num epoch	5	num layers	2
summary freq	1000	**time horizon**	**32**
extrinsic strength	1.0	extrinsic gamma	0.9

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 25
Action space size = 1

Episode steps = 3000
Total steps = 303000
Time = 6600s

Tests with different parameters: time horizon (from 256 to 32) does not seem to change anything.

4_27_1533

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	8192	**hidden units**	**64**
learning rate	**5.0e-4**	**learning rate schedule**	**constant**
max steps	5.0e6	normalize	false
num epoch	**3**	**num layers**	**3**
summary freq	1000	**time horizon**	**256**
extrinsic strength	1.0	extrinsic gamma	0.9

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 25
Action space size = 1

Episode steps = 3000
Total steps = 942000
Time = 19900s

Different network architecture. It has more layers but fewer units per layer.

It's close to the shape but still has a lot of noise, plus it's very slow. The curiosity parameter may need to be retrieved and add more observations. After many hours it gets to -6 reward.

It may also be possible to reduce the length of each episode.

4_28_1800

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	8192	**hidden units**	**128**
learning rate	**4.0e-4**	learning rate schedule	constant
max steps	5.0e6	normalize	false
num epoch	3	**num layers**	**2**
summary freq	1000	time horizon	256
extrinsic strength	1.0	extrinsic gamma	0.9
curiosity strength	**0.05**	**curiosity gamma**	**0.99**
curiosity encoding size	**256**		

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 25
Action space size = 2

Episode steps = 1000
Total steps = 74000
Time = 1550s

Drastic changes in rewards (factors and standard deviations). Now returns 2 values in vector action (the expected maximum and minimum).

With this test I know that the model can set the maximum and minimum. Incomplete.

4_28_1833

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	8192	hidden units	128
learning rate	4.0e-4	learning rate schedule	constant
max steps	5.0e6	normalize	false
num epoch	3	num layers	2
summary freq	1000	time horizon	256
extrinsic strength	1.0	extrinsic gamma	0.9
curiosity strength	0.05	curiosity gamma	0.99
curiosity encoding size	256		

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 25
Action space size = 2

Episode steps = 1000
Total steps = 360000
Time = 7600s

Same test as before, but longer. It manages to wrap the movement but with too wide a range.

4_29_1034

Trainer: PPO

batch size	1024	**beta**	**1.0e-2**
buffer size	8192	**hidden units**	**256**
learning rate	**6.0e-4**	**learning rate schedule**	**linear**
max steps	5.0e6	normalize	false
num epoch	**10**	num layers	2
summary freq	1000	**time horizon**	**512**
extrinsic strength	1.0	extrinsic gamma	0.9
curiosity strength	0.05	curiosity gamma	0.99
curiosity encoding size	256		

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 25
Action space size = 2

Episode steps = 1000
Total steps = 226000
Time = 5400s

The reward now takes into account the relationship between standard deviations.
Like the previous one, it stagnates at a certain point (this time at -1, but with similar behavior).

4_29_1209

Trainer: PPO

batch size	1024	beta	1.0e-2
buffer size	8192	hidden units	256
learning rate	**2.0e-5**	learning rate schedule	linear
max steps	5.0e6	normalize	false
num epoch	10	num layers	2
summary freq	1000	time horizon	512
extrinsic strength	1.0	extrinsic gamma	0.9
curiosity strength	0.05	curiosity gamma	0.99
curiosity encoding size	256		

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 25
Action space size = 2

Episode steps = 1000
Total steps = 200000
Time = 4700s

Same training but with a lower learning rate (and scores multiplied by 20). Doesn't seem to advance too far so it's not saved.

4_29_1334

Trainer: PPO

batch size	1024	**beta**	**5.0e-3**
buffer size	8192	hidden units	256
learning rate	**5.0e-5**	**learning rate schedule**	**constant**
max steps	5.0e6	normalize	false
num epoch	10	num layers	2
summary freq	1000	time horizon	512
extrinsic strength	1.0	**extrinsic gamma**	**0.8**
curiosity strength	**0.1**	**curiosity gamma**	**0.9**
curiosity encoding size	**128**		

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 25
Action space size = 2

Episode steps = 1000
Total steps = 297000
Time = 7920s

For some reason the 2 curves end up below the bot value, and it doesn't get much of a penalty. There must be some mistake in the rewards that allows him not to learn properly and still increase the reward.

4_29_1827

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	8192	hidden units	256
learning rate	**5.0e-4**	learning rate schedule	constant
max steps	5.0e6	normalize	false
num epoch	**5**	num layers	2
summary freq	1000	time horizon	512
extrinsic strength	1.0	extrinsic gamma	0.8
curiosity strength	**0.01**	curiosity gamma	0.9
curiosity encoding size	128		

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 25
Action space size = 2

Episode steps = 1000
Total steps = 265000
Time = 6200s

The reward rises to positive but stabilizes at +12 after 130,000 steps. It reaches a positive reward at 90000.

The network learns quickly to separate the maximum and minimum but with very wide ranges. It would be necessary to increase the requirement by penalizing more neutral scores (and perhaps vary how each reward factor affects it).

4_30_1043

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	8192	hidden units	256
learning rate	5.0e-4	learning rate schedule	constant
max steps	5.0e6	normalize	false
num epoch	5	num layers	2
summary freq	1000	time horizon	512
extrinsic strength	1.0	extrinsic gamma	0.8
curiosity strength	0.01	curiosity gamma	0.9
curiosity encoding size	128		

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 25
Action space size = 2

Episode steps = 1000
Total steps = 317000
Time = 7440s

Training with curriculum. It has the parameter "requirement", which makes that when the reward reaches 0.5, 1 is subtracted from future rewards so that it must adjust the graph more. It has several levels until subtracting 25 points (it goes from 1 to 1).

There are several levels of the curriculum that go up too fast, they should be compressed into less. On the other hand, I have found that it is necessary to use a weighted average to make it react correctly to impulses.

4_30_1316

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	8192	hidden units	256
learning rate	5.0e-4	learning rate schedule	constant
max steps	5.0e6	normalize	false
num epoch	5	num layers	2
summary freq	1000	time horizon	512
extrinsic strength	1.0	extrinsic gamma	0.8
curiosity strength	0.01	curiosity gamma	0.9
curiosity encoding size	128		

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 25
Action space size = 2

Episode steps = 1000
Total steps = 265000
Time = 6215s

Same training but with several errors corrected (such as the relationship between stds or some score limits), in addition to having added a weighted mean and a minimum standard deviation. The learning curriculum levels should be adjusted later.

The network stagnates over 0, it does not get past the first level. It should learn to adjust the lines.

4_30_1512

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	8192	**epsilon**	**0.3**
hidden units	256	**learning rate**	**1.0e-3**
learning rate schedule	constant	max steps	5.0e6
normalize	false	num epoch	5
num layers	**1**	summary freq	1000
time horizon	**5**		
extrinsic strength	1.0	extrinsic gamma	0.8
curiosity strength	0.01	curiosity gamma	0.9
curiosity encoding size	128		

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 25
Action space size = 2

Episode steps = 1000
Total steps = 128000
Time = 3050s

Tests with different parameters. It learns faster but it gets stuck again. This seems to have its cause in the fact that if the agent estimates a very large range in relation to the deviation of the bot even though the reward is decreased, the punishment is also decreased. To adjust this phenomenon I thought to change the deviation ratio parameter to another factor that will also depend on coherence (using clipped cubic functions).

It may be necessary to combine learning with behavioral cloning at first.

4_30_1714

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	8192	epsilon	0.3
hidden units	256	learning rate	1.0e-3
learning rate schedule	constant	max steps	5.0e6
normalize	false	num epoch	5
num layers	1	summary freq	1000
time horizon	5		
extrinsic strength	1.0	extrinsic gamma	0.8
curiosity strength	0.01	curiosity gamma	0.9
curiosity encoding size	128		

Render Target Sensor: 64x64

Grayscale: false
Observation space size = 25
Action space size = 2

Episode steps = 1000
Total steps = 225000
Time = 5500s

Corrected the deviation ratio factor, now penalizes more or less depending on coherence (if by covering a large range fails, the punishment is increased, if it succeeds, the reward is decreased).

Apart from what was mentioned before of BC with GAIL, it would also be good to try a multiplicative (instead of subtractive) requirement once the rewards are balanced. In spite of lowering the condition to move up a level, it does not converge but stays where it was.

4_30_1849

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	8192	epsilon	0.3
hidden units	256	**learning rate**	**2.0e-3**
learning rate schedule	constant	max steps	5.0e6
normalize	false	num epoch	5
num layers	1	summary freq	1000
time horizon	5		
extrinsic strength	1.0	extrinsic gamma	0.8
curiosity strength	0.01	curiosity gamma	0.9
curiosity encoding size	128		

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 25
Action space size = 2

Episode steps = 1000
Total steps = 225000
Time = 5500s

Same training as before but with a higher learning rate and greater initial demand.

Still not stabilized. It will be necessary to penalize the fact that the range is greater at the moment of stability (instead of decreasing reward).

5_01_1005

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	8192	epsilon	0.3
hidden units	256	learning rate	2.0e-3
learning rate schedule	constant	max steps	5.0e6
normalize	false	num epoch	5
num layers	1	summary freq	1000
time horizon	5		
extrinsic strength	1.0	extrinsic gamma	0.8
curiosity strength	0.01	curiosity gamma	0.9
curiosity encoding size	128		

Render Target Sensor: 64x64

Grayscale: false

Observation space size = 25

Action space size = 2

Episode steps = 1000

Total steps = 90000

Time = 2100s

Training with an initial demand of 15. Compare with the previous ones to see where it stagnates.

Curiously, it is still stagnating at 0, but this one seems to be behaving a little better. Now I will test a training with much more demand, then the multiplicative demand and finally rewards that only penalize for moves being consistent and only reward for moves being inconsistent (more or less).

5_01_1110

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	8192	epsilon	0.3
hidden units	256	learning rate	2.0e-3
learning rate schedule	constant	max steps	5.0e6
normalize	false	num epoch	5
num layers	1	summary freq	1000
time horizon	5		
extrinsic strength	1.0	extrinsic gamma	0.8
curiosity strength	0.01	curiosity gamma	0.9
curiosity encoding size	128		

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 25
Action space size = 2

Episode steps = 1000
Total steps = 75000
Time = 1750s

Same training but with 50 demands. It may come closer to the target or divert it by being excessively negative. The graph converges to -50 so that the subtractive requirement has proven to be ineffective.

5_01_1245

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	8192	epsilon	0.3
hidden units	256	learning rate	2.0e-3
learning rate schedule	constant	max steps	5.0e6
normalize	false	num epoch	5
num layers	1	summary freq	1000
time horizon	5		
extrinsic strength	1.0	extrinsic gamma	0.8
curiosity strength	0.01	curiosity gamma	0.9
curiosity encoding size	128		

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 25
Action space size = 2

Episode steps = 1000
Total steps = 420000
Time = 10700s

Simpler training without a curriculum: if the movement is coherent it can only be penalized, if it is incoherent it can only be rewarded. At the moment I use linear formulas.

As the rewards are now, it slowly goes up to -7 and at 360000 steps it diverges. Halfway through it seemed to fit the coherent movement, but it doesn't capture the impulses well (it doesn't get enough reward).

5_01_1552

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	8192	epsilon	0.3
hidden units	256	learning rate	2.0e-3
learning rate schedule	constant	max steps	5.0e6
normalize	false	num epoch	5
num layers	1	summary freq	1000
time horizon	5		
extrinsic strength	1.0	extrinsic gamma	0.8
curiosity strength	**0.1**	**curiosity gamma**	**0.8**
curiosity encoding size	128		

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 25
Action space size = 2

Episode steps = 1000
Total steps = 390000
Time = 10000s

Inconsistent movement reward improved by 10, and increased curiosity.

This time it has not collapsed, and although it adapts quite well to coherent movement it does not carry impulses very well (maybe because it is not penalized at all, although sometimes it "feigns" to follow them).

5_01_1851

Trainer: PPO

batch size	1024	**beta**	**8.0e-3**
buffer size	8192	epsilon	0.3
hidden units	256	learning rate	2.0e-3
learning rate schedule	**linear**	**max steps**	**6.0e5**
normalize	false	num epoch	5
num layers	1	summary freq	1000
time horizon	5		
extrinsic strength	1.0	extrinsic gamma	0.8
curiosity strength	0.1	curiosity gamma	0.8
curiosity encoding size	128		

Render Target Sensor: 64x64

Grayscale: false

Observation space size = 25

Action space size = 2

Episode steps = 1000

Total steps = 600000

Time = 15800s

Added small negative rewards when an impulse fails (less than for coherent movements). In addition, training has been limited to 600,000 steps with a linear learning rate.

The movement seems correct on the graph, although it would be nice to decrease the amount of noise. However, it is enough to use GAIL with correct rewards.

*Thinking about it, it would also be good to treat cases where the standard deviation is 0, since a model like this where the deviation has a certain minimum would not be good for behaviours where the bot can stand completely still. FURTHER, it would be convenient if the deviation could be decreased once it has reacted, to make the transitions between movements more fluid (having a wide random range, the movement can be somewhat chaotic).

5_03_1758

Trainer: PPO

batch size	1024	beta	8.0e-3
buffer size	8192	epsilon	0.3
hidden units	256	learning rate	2.0e-3
learning rate schedule	linear	max steps	6.0e5
normalize	false	num epoch	5
num layers	1	summary freq	1000
time horizon	5		
extrinsic strength	1.0	extrinsic gamma	0.8
curiosity strength	0.1	curiosity gamma	0.8
curiosity encoding size	128		
gail strength	**0.01**	**gail gamma**	**0.99**
gail encoding size	**128**		

Demo path: demos/MyStdDevDemo.demo

Render Target Sensor: 64x64

Grayscale: false

Observation space size = 25

Action space size = 2

Episode steps = 1000

Total steps = 516000

Time = 15200s

Added agent demonstration and GAIL reward. Pretraining parameter could be added later to check if it improves training. The result is very similar to the previous one, and that is why it is not considered as good.

5_03_2223

Trainer: PPO

batch size	1024	beta	8.0e-3
buffer size	8192	epsilon	0.3
hidden units	256	learning rate	2.0e-3
learning rate schedule	linear	max steps	6.0e5
normalize	false	num epoch	5
num layers	1	summary freq	1000
time horizon	5		
extrinsic strength	1.0	extrinsic gamma	0.8
curiosity strength	0.1	curiosity gamma	0.8
curiosity encoding size	128		
gail strength	**0.6**	gail gamma	0.99
gail encoding size	128		

Demo path: demos/MyStdDevDemo.demo

Render Target Sensor: 64x64

Grayscale: false

Observation space size = 25

Action space size = 2

Episode steps = 1000

Total steps = 280000

Time = 8000s

More weight to GAIL rewards. With 0.7 gamma, it diverts at 82k steps.

With the above parameters it has strange behaviors, like placing the 2 lines over 0. It may need pre-training or a larger dataset.

5_04_1008

Trainer: PPO

batch size	1024	beta	8.0e-3
buffer size	8192	epsilon	0.3
hidden units	256	learning rate	2.0e-3
learning rate schedule	linear	max steps	6.0e5
normalize	false	num epoch	5
num layers	1	summary freq	1000
time horizon	5		
pretraining strength	**0.5**	**pretraining steps**	**10000**
extrinsic strength	**0.7**	extrinsic gamma	0.8
curiosity strength	0.1	curiosity gamma	0.8
curiosity encoding size	128		
gail strength	0.6	gail gamma	0.99
gail encoding size	128		

Demo path: demos/MyStdDevDemo.demo

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 25
Action space size = 2

Episode steps = 1000
Total steps = 153000
Time = 4300s

At 40,000 steps it has received a great reward in curiosity that has caused the model to collapse, from there it has not recovered.

5_04_1136

Trainer: PPO

batch size	1024	beta	8.0e-3
buffer size	8192	epsilon	0.3
hidden units	256	learning rate	2.0e-3
learning rate schedule	linear	max steps	6.0e5
normalize	false	num epoch	5
num layers	1	summary freq	1000
time horizon	5		
pretraining strength	0.5	**pretraining steps**	**20000**
extrinsic strength	0.7	**extrinsic gamma**	**0.9**
gail strength	**0.5**	gail gamma	0.99
gail encoding size	**256**		

Demo path: demos/MyStdDevDemo.demo

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 25
Action space size = 2

Episode steps = 1000
Total steps = 600000
Time = 16200s

Curiosity eliminated, changes in other parameters and more pretraining. Doesn't seem to get as close to the target as it used to, it stagnates earlier. For the next training, base rewards on gail only.

5_04_1612

Trainer: PPO

batch size	1024	beta	8.0e-3
buffer size	8192	epsilon	0.3
hidden units	256	**learning rate**	**5.0e-4**
learning rate schedule	linear	max steps	6.0e5
normalize	false	num epoch	5
num layers	1	summary freq	1000
time horizon	**64**		
pretraining strength	0.5	pretraining steps	20000
gail strength	**1.0**	gail gamma	0.99
gail encoding size	**128**		

Demo path: demos/MyStdDevDemo.demo

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 25
Action space size = 2

Episode steps = 1000
Total steps = 393000
Time = 9400s

Rewards based on GAIL only. The learning rate has been lowered to stabilize learning.

No improvement at all. Maybe the problem is that the dataset is small.

5_05_1058

Trainer: PPO

batch size	1024	beta	8.0e-3
buffer size	8192	epsilon	0.3
hidden units	256	learning rate	5.0e-4
learning rate schedule	linear	max steps	6.0e5
normalize	false	num epoch	5
num layers	1	summary freq	1000
time horizon	64		
pretraining strength	0.5	**pretraining steps**	**40000**
gail strength	1.0	gail gamma	0.99
gail encoding size	128		

Demo path: demos/LongStdDDemo.demo

Render Target Sensor: 64x64

Grayscale: false

Observation space size = 25

Action space size = 2

Episode steps = 1000

Total steps = 60000

Time = 1500s

Found a mistake that made the demos all wrong, the previous models do not serve as an example. Retraining with parameters similar to 5_04_1612 and a longer and more correct demo. In this one only GAIL is used to see the result.

At the beginning it has had a very high rise, but it has stagnated afterwards. As the adversary network was not going to train anymore, I don't expect it to go up much. Set pretraining steps to 0 to keep the opponent net training. Maybe it can be turned off at the end. Also increase the strength.

5_05_1158

Trainer: PPO

batch size	1024	beta	8.0e-3
buffer size	8192	epsilon	0.3
hidden units	256	learning rate	5.0e-4
learning rate schedule	linear	max steps	6.0e5
normalize	false	num epoch	5
num layers	1	summary freq	1000
time horizon	64		
pretraining strength	**0.7**	**pretraining steps**	**600000**
gail strength	1.0	gail gamma	0.99
gail encoding size	128		

Demo path: demos/LongStdDDemo.demo

Render Target Sensor: 64x64

Grayscale: false

Observation space size = 25

Action space size = 2

Episode steps = 1000

Total steps = 110000

Time = 2700s

The reward goes up but ends up swinging in negative values. It may be necessary to increase the learning rate, increase entropy regulation or add environmental rewards. It seems that the network manages to fool the opponent's network faster than it learns.

5_05_1224

Trainer: PPO

batch size	1024	beta	8.0e-3
buffer size	8192	epsilon	0.3
hidden units	256	**learning rate**	**2.0e-3**
learning rate schedule	linear	max steps	6.0e5
normalize	false	num epoch	5
num layers	1	summary freq	1000
time horizon	64		
pretraining strength	**0.8**	pretraining steps	600000
extrinsic strength	**1.0**	**extrinsic gamma**	**0.99**
curiosity strength	**0.1**	**curiosity gamma**	**0.99**
curiosity encoding size	**128**		
gail strength	1.0	gail gamma	0.99
gail encoding size	128		

Demo path: demos/LongStdDDemo.demo

Render Target Sensor: 64x64

Grayscale: false

Observation space size = 25

Action space size = 2

Episode steps = 1000

Total steps = 50000

Time = 1300s

Standard rewards restored. Imitation has been given more strength but has less influence on the model's reward.

I have interrupted the model to train a reference one, with the values of the one that was successful but the new rewards to see if it is still correct. The model has managed to raise the rewards several steps, although I should analyze issues such as entropy or losses to adjust more parameters. The goal is to get a faster workout than the benchmark, so that it can be scaled.

5_05_1300

Trainer: PPO

batch size	1024	beta	8.0e-3
buffer size	8192	epsilon	0.3
hidden units	256	learning rate	2.0e-3
learning rate schedule	linear	max steps	6.0e5
normalize	false	num epoch	5
num layers	1	summary freq	1000
time horizon	64		
extrinsic strength	1.0	**extrinsic gamma**	**0.8**
curiosity strength	0.1	**curiosity gamma**	**0.8**
curiosity encoding size	128		

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 25
Action space size = 2

Episode steps = 1000
Total steps = 600000
Time = 14100s

Training with the previous values (except time horizon) of the one that came closest to success. Quite correct model, serves as a reference for rewards (training time should be improved). However, it doesn't handle impulse linking or the ones on the right side very well. From the demo a reward of 0.25 can be expected as a maximum optimistic (this would be quite correct behaviour).

5_05_1715

Trainer: PPO

batch size	1024	beta	8.0e-3
buffer size	8192	epsilon	0.3
hidden units	256	learning rate	2.0e-3
learning rate schedule	linear	max steps	6.0e5
normalize	false	num epoch	5
num layers	1	summary freq	1000
time horizon	64		
pretraining strength	**0.5**	**pretraining steps**	**600000**
extrinsic strength	1.0	extrinsic gamma	0.8
curiosity strength	0.1	curiosity gamma	0.8
curiosity encoding size	128		
gail strength	**0.02**	**gail gamma**	**0.99**
gail encoding size	**256**		

Demo path: demos/LongStdDDemo.demo

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 25
Action space size = 2

Episode steps = 1000
Total steps = 77000
Time = 2000s

Test with the 3 parameters (inspired by Pyramids), following the indications of the documentation. The aim is to achieve a faster or more rewarding workout than the reference one (5_05_1300).

It has started well but ends up being between -60 and -80. In the following I will add other optional parameters to see if the learning is stabilized.

5_05_1752

Trainer: PPO

batch size	1024	beta	8.0e-3
buffer size	8192	epsilon	0.3
hidden units	256	learning rate	2.0e-3
learning rate schedule	linear	max steps	6.0e5
normalize	false	num epoch	5
num layers	1	summary freq	1000
time horizon	64		
pretraining strength	0.5	pretraining steps	600000
extrinsic strength	1.0	extrinsic gamma	0.8
curiosity strength	0.1	curiosity gamma	0.8
curiosity encoding size	128		
gail strength	0.02	gail gamma	0.99
gail encoding size	256	**gail use vail**	**true**

Demo path: demos/LongStdDDemo.demo

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 25
Action space size = 2

Episode steps = 1000
Total steps = 88000
Time = 2300s

Vail has been added to the training (bottleneck). It should be more stable, but, decrease curiosity, add use actions (to imitate, in theory) and/or add learning rate of GAIL.

Very similar result.

5_05_1834

Trainer: PPO

batch size	1024	beta	8.0e-3
buffer size	8192	epsilon	0.3
hidden units	256	learning rate	2.0e-3
learning rate schedule	linear	max steps	6.0e5
normalize	false	num epoch	5
num layers	1	summary freq	1000
time horizon	64		
pretraining strength	0.5	pretraining steps	600000
extrinsic strength	1.0	extrinsic gamma	0.8
curiosity strength	0.1	curiosity gamma	0.8
curiosity encoding size	128		
gail strength	0.02	gail gamma	0.99
gail encoding size	256	**gail use vail**	**false**
gail use actions	**true**		

Demo path: demos/LongStdDDemo.demo

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 25
Action space size = 2

Episode steps = 1000
Total steps = 110000
Time = 2900s

Added use actions to true, and removed the use vail.

It oscillates on the same values, although a little higher, but still not improving on the original.

5_05_1925

Trainer: PPO

batch size	1024	beta	8.0e-3
buffer size	8192	epsilon	0.3
hidden units	256	learning rate	2.0e-3
learning rate schedule	linear	max steps	6.0e5
normalize	false	num epoch	5
num layers	1	summary freq	1000
time horizon	64		
pretraining strength	0.5	pretraining steps	600000
extrinsic strength	1.0	extrinsic gamma	0.8
curiosity strength	0.1	curiosity gamma	0.8
curiosity encoding size	128		
gail strength	**0.01**	**gail gamma**	**0.95**
gail learning rate	**0.00005**	**gail encoding size**	**64**
gail use vail	**true**	gail use actions	true

Demo path: demos/LongStdDDemo.demo

Render Target Sensor: 64x64

Grayscale: false

Observation space size = 25

Action space size = 2

Episode steps = 1000

Total steps = 98000

Time = 2500s

Changes in a few variables, and added learning rate. It shows the same trend.

5_05_2017

Trainer: PPO

batch size	1024	beta	8.0e-3
buffer size	8192	epsilon	0.3
hidden units	256	learning rate	2.0e-3
learning rate schedule	linear	max steps	6.0e5
normalize	false	num epoch	5
num layers	1	summary freq	1000
time horizon	64		
extrinsic strength	1.0	extrinsic gamma	0.8
curiosity strength	0.1	curiosity gamma	0.8
curiosity encoding size	128		
gail strength	0.01	gail gamma	0.95
gail learning rate	**0.0005**	gail encoding size	64
gail use vail	true	gail use actions	true

Demo path: demos/LongStdDDemo.demo

Render Target Sensor: 64x64

Grayscale: false

Observation space size = 25

Action space size = 2

Episode steps = 1000

Total steps = 600000

Time = 15200s

Pretraining removed. The result was practically the same as in the reference, and has reached the same reward. However, it does not seem that the GAIL rewards have influenced anything, maybe the curiosity is too high, besides, it has taken 20 minutes more to do the same steps. It might be worth trying out new rewards.

5_06_1036

Trainer: PPO

batch size	1024	beta	8.0e-3
buffer size	8192	epsilon	0.3
hidden units	256	learning rate	2.0e-3
learning rate schedule	linear	max steps	6.0e5
normalize	false	num epoch	5
num layers	1	summary freq	1000
time horizon	64		
extrinsic strength	1.0	**extrinsic gamma**	**0.9**
curiosity strength	**0.05**	**curiosity gamma**	**0.9**
curiosity encoding size	128		

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 25
Action space size = 2

Episode steps = 1000
Total steps = 527000
Time = 12500s

Benchmark training with the new linear rewards. A jump may have to be added to the penalties depending on the outcome.

The reward goes up in parallel to the previous reference, but it doesn't take much to detect impulses. It will be necessary to add a jump to the penalties.

5_06_1415

Trainer: PPO

batch size	1024	beta	8.0e-3
buffer size	8192	epsilon	0.3
hidden units	256	learning rate	2.0e-3
learning rate schedule	linear	max steps	6.0e5
normalize	false	num epoch	5
num layers	1	summary freq	1000
time horizon	64		
extrinsic strength	1.0	extrinsic gamma	0.9
curiosity strength	0.05	curiosity gamma	0.9
curiosity encoding size	128		

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 25

Action space size = 2

Episode steps = 1000
Total steps = 242000
Time = 5200s

Added a jump in the rewards (locking everything up penalizes the same as failing by 0.00001 units). It has a very similar trend, so it doesn't improve the previous one.

5_06_1848

Trainer: PPO

batch size	1024	beta	8.0e-3
buffer size	8192	epsilon	0.3
hidden units	256	learning rate	2.0e-3
learning rate schedule	linear	max steps	6.0e5
normalize	false	num epoch	5
num layers	1	summary freq	1000
time horizon	64		
extrinsic strength	1.0	extrinsic gamma	0.9
curiosity strength	0.05	curiosity gamma	0.9
curiosity encoding size	128		

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 25
Action space size = 2

Episode steps = 1000
Total steps = -
Time = -

Parameter tests. None of them work as well as they should. The next test with SAC.

5_07_0956

Trainer: SAC

batch size	128	buffer size	50000
buffer init steps	0	hidden units	128
init entcoef	1.0	learning rate	1.0e-3
learning rate schedule	constant	max steps	6.0e5
memory size	256	normalize	true
num update	1	train interval	5
num layers	2	time horizon	1024
sequence length	64	summary freq	1000
tau	0.01	use recurrent	false
vis encode type	simple		
pretraining strength	0.5	pretraining steps	10000
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.01	curiosity gamma	0.99
curiosity encoding size	128		
gail strength	0.01	gail gamma	0.99
gail encoding size	128		

Demo path: demos/LongStdDDemo.demo

Render Target Sensor: 64x64

Grayscale: false

Observation space size = 25

Action space size = 2

Episode steps = 1000

Total steps = 72000

Time = 4600s

Training with SAC. It is much slower than PPO but requires less samples, has risen fast but has diverged by 70k.

5_07_1142

Trainer: SAC

batch size	128	buffer size	50000
buffer init steps	0	hidden units	128
init entcoef	1.0	**learning rate**	**8.0e-4**
learning rate schedule	**linear**	max steps	6.0e5
memory size	256	normalize	true
num update	1	train interval	5
num layers	2	**time horizon**	**64**
sequence length	64	summary freq	1000
tau	**0.007**	use recurrent	false
vis encode type	simple		
pretraining strength	**0.7**	**pretraining steps**	**20000**
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.01	curiosity gamma	0.99
curiosity encoding size	128		
gail strength	0.01	gail gamma	0.99
gail encoding size	128	**use actions**	**true**

Demo path: demos/LongStdDDemo.demo

Render Target Sensor: 64x64

Grayscale: false

Observation space size = 25

Action space size = 2

Episode steps = 1000

Total steps = 194000

Time = 11900s

Back to previous rewards and changes in some parameters. The network adapts quite well to impulses although it has intermittent drops in rewards (smaller each time, it may be part of how the model works). Better results could be obtained with this model, but the training is much slower in proportion.

5_07_1512

Trainer: SAC

batch size	128	buffer size	50000
buffer init steps	**2000**	**hidden units**	**256**
init entcoef	**1.5**	**learning rate**	**1.0e-3**
learning rate schedule	**constant**	max steps	6.0e5
memory size	256	normalize	true
num update	1	train interval	5
num layers	**1**	time horizon	64
sequence length	64	summary freq	1000
tau	**0.005**	use recurrent	false
vis encode type	simple		
pretraining strength	0.7	pretraining steps	20000
extrinsic strength	**2.0**	extrinsic gamma	0.99
curiosity strength	**0.02**	curiosity gamma	0.99
curiosity encoding size	128		
gail strength	**0.02**	gail gamma	0.99
gail encoding size	128	use actions	true

Demo path: demos/LongStdDDemo.demo

Render Target Sensor: 64x64

Grayscale: false

Observation space size = 25

Action space size = 2

Episode steps = 1000

Total steps = -

Time = -

Changes in layer structure. Cancelled.

5_07_1730

Trainer: SAC

batch size	128	buffer size	50000
buffer init steps	2000	hidden units	256
init entcoef	1.5	learning rate	1.0e-3
learning rate schedule	constant	max steps	6.0e5
memory size	256	normalize	true
num update	1	train interval	5
num layers	1	time horizon	64
sequence length	64	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	simple		
pretraining strength	0.7	pretraining steps	20000
extrinsic strength	2.0	extrinsic gamma	0.99
curiosity strength	0.02	curiosity gamma	0.99
curiosity encoding size	128		
gail strength	0.02	gail gamma	0.99
gail encoding size	128	use actions	true

Demo path: demos/LongStdDDemo.demo

Render Target Sensor: 64x64

Grayscale: false

Observation space size = 25

Action space size = 2

Episode steps = 1000

Total steps = 215000

Time = 4600s

GPU training tests, I don't know if it really works or not

-The time scale of the training cannot be changed.

-Adding twice as many environments causes twice as many steps to be taken, but the curve increases by half as fast (more or less).

In this case, the agent ends up farther away from the reward. The next, train with less learning rate.

5_07_2039

Trainer: SAC

batch size	128	**buffer size**	**200000**
buffer init steps	**5000**	hidden units	256
init entcoef	**0.7**	**learning rate**	**4.0e-4**
learning rate schedule	constant	max steps	6.0e5
memory size	256	normalize	true
num update	1	train interval	5
num layers	1	time horizon	64
sequence length	**128**	summary freq	1000
tau	0.005	**use recurrent**	**true**
vis encode type	simple		
pretraining strength	**0.4**	pretraining steps	20000
extrinsic strength	**1.5**	extrinsic gamma	0.99
curiosity strength	**0.03**	curiosity gamma	0.99
curiosity encoding size	128		
gail strength	**0.03**	gail gamma	0.99
gail encoding size	128	use actions	true

Demo path: demos/NoObsDemo.demo

Render Target Sensor: 64x64

Grayscale: false

Observation space size = 0

Action space size = 2

Episode steps = 1000

Total steps = 57000

Time = 3500s

The 25 previous observations have been changed to “use recurrent”. The CNN encoder has also been changed, so it may take a little longer to train (it shouldn’t be as relevant as before, when inferring by GPU). It seems that increasing the buffer causes it to go much slower: contrast with the final learning result (if it doesn’t improve, check the same without buffer, if it doesn’t change it will be that the 25 observations are essential). It has not learned anything.

5_07_2209

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	**1.0**	learning rate	4.0e-4
learning rate schedule	constant	max steps	6.0e5
memory size	256	normalize	true
num update	1	train interval	5
num layers	1	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	**use recurrent**	**false**
vis encode type	simple		
pretraining strength	0.4	pretraining steps	20000
extrinsic strength	1.5	extrinsic gamma	0.99
curiosity strength	0.03	curiosity gamma	0.99
curiosity encoding size	128		
gail strength	0.03	gail gamma	0.99
gail encoding size	128	use actions	true

Demo path: demos/NoObsDemo.demo

Render Target Sensor: 64x64

Grayscale: false

Observation space size = 0

Action space size = 2

Episode steps = 1000

Total steps = 600000

Time = 14000s

Use recurrent is what causes training to slow down considerably.

Without memory parameters it still seems to adapt correctly, and has a speed similar to ppo. However, it performs a kind of “counterpulses”, which perhaps would be corrected with more training time.

The rewards of each standard deviation could be separated (by giving 1 of each + the composite), and if noise were added, the actual previous steps could also be used (in SAC they may work better).

5_09_1222

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	4.0e-4
learning rate schedule	constant	max steps	6.0e5
memory size	256	normalize	true
num update	1	train interval	5
num layers	1	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	simple		
pretraining strength	0.4	pretraining steps	20000
extrinsic strength	1.5	extrinsic gamma	0.99
curiosity strength	0.03	curiosity gamma	0.99
curiosity encoding size	128		
gail strength	0.03	gail gamma	0.99
gail encoding size	128	use actions	true

Demo path: demos/IndDemo.demo

Render Target Sensor: 64x64

Grayscale: false

Observation space size = 0

Action space size = 2

Episode steps = 1000

Total steps = 600000

Time = 14000s

Individual rewards have been added for each line (they are currently only linear, and with less weight). The demo is changed so that the rewards are consistent (maximum is 3). It is saved as a checkpoint for further training. It adapts surprisingly well to impulses, although I think there is still room for further noise reduction.

5_09_1643

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	4.0e-4
learning rate schedule	constant	max steps	6.0e5
memory size	256	normalize	true
num update	1	train interval	5
num layers	1	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	simple		
pretraining strength	0.4	pretraining steps	20000
extrinsic strength	1.5	extrinsic gamma	0.99
curiosity strength	0.03	curiosity gamma	0.99
curiosity encoding size	128		
gail strength	0.03	gail gamma	0.99
gail encoding size	128	use actions	true

Demo path: demos/IndDemo.demo

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 0
Action space size = 2

Episode steps = 1000
Total steps = 600000
Time = 14000s + 7700s

Same model as the last one but woth more training steps. Max steps has been changed to leave it indefinitely. It has not had any major performance improvements.

Chapter 2

Reaction time

5_09_1855

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	8192	epsilon	0.2
hidden units	256	learning rate	2.0e-3
learning rate schedule	linear	max steps	6.0e5
normalize	true	num epoch	6
num layers	1	summary freq	1000
time horizon	64		
pretraining strength	0.8	pretraining steps	20000
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.01	curiosity gamma	0.99
curiosity encoding size	128		
gail strength	0.01	gail gamma	0.99
gail encoding size	128		

Demo path: demos/IndDemo.demo

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 0
Action space size = 2

Episode steps = 1000

Total steps = 524000
Time = 5580s

Speed test using ppo with 3 envs and GPU, same rewards as in 5_09_1643
It works twice as fast, so it can be good for testing (and using SAC for final versions). However, it often gives errors like Out of memory (and other stranger ones that may be caused by overheating).

5_12_2000

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	4.0e-4
learning rate schedule	constant	max steps	6.0e5
memory size	256	normalize	true
num update	1	train interval	5
num layers	1	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	simple		
extrinsic strength	1.5	extrinsic gamma	0.99
curiosity strength	0.03	curiosity gamma	0.99
curiosity encoding size	128		

Frames [1]: 0, 5, 10, 12, 15, 17, 20, 22, 25, 27, 30, 32, 35, 37, 40, 45, 50, 55, 60, 70, 80

Render Target Sensor: 21 * (20x20)
Grayscale: true
Observation space size = 0
Action space size = 2

Episode steps = 1000
Total steps = 278000
Time = 17650s

Bot with 20 frames of memory (up to 1.4 seconds), and motion memory. No reactions are used, it serves to check efficiency with SAC and 3x envs. At the moment it gives negative dimension error when making convolution.

[1] Frame 0 is the current frame, which is used in every training, the other frames are the n-th previous frame.

Using 2 images of 16x16 fails, but with 2 of 64x64 it works. This last one also goes with grayscale.

With 3 16x16 images grayscale doesn't work. 16 of 64x64 fails.

With 3 images 64x64 grayscale does work. With 4 images it also works.

20x20 seems to be the minimum (or close) to be able to convolute without errors. However, it is necessary that all images are the same (I could check this later, but it seems to be the case). It gives error later. THIS ERROR was due to the fact that the demo was made with 16x16 render targets, so I remove GAIL at this moment.

Personally, I don't like very much the "21x20x20" thing.

The training becomes about 3 times slower despite using grayscale, but the results are not bad. It might be better to use PPO in these test cases.

GAIL may be quite relevant, as it has improved the reward but has not adapted to the idle movement too closely.

5_13_1118

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	4.0e-4
learning rate schedule	constant	max steps	6.0e5
memory size	256	normalize	true
num update	1	train interval	5
num layers	1	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	simple		
pretraining strength	**0.4**	**pretraining steps**	**20000**
extrinsic strength	1.5	extrinsic gamma	0.99
curiosity strength	0.03	curiosity gamma	0.99
curiosity encoding size	128		
gail strength	**0.03**	**gail gamma**	**0.99**
gail encoding size	**128**	**gail use actions**	**true**

Demo path: demos/Frames20Demo.demo

Frames: 0, 5, 10, 12, 15, 17, 20, 22, 25, 27, 30, 32, 35, 40, 45, 50, 55, 60, 70, 80

Render Target Sensor: 20 * (20x20)

Grayscale: true

Observation space size = 0

Action space size = 2

Episode steps = 1000
Total steps = 254000
Time = 20750s

Removed the frame 37. Training with a bot with noise and delay (BotDelayNoised, before it was BotNoised). Now the impulses have a different shape.

The curve is similar to the previous training. GAIL makes it even slower (from 3h 20m to 4h 20m at 200k steps). The average and deviations are better being more invariant, I should restore the previous settings (and improve the demo bot for this model).

5_13_1819

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	4.0e-4
learning rate schedule	constant	max steps	6.0e5
memory size	256	normalize	true
num update	1	train interval	5
num layers	1	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	simple		
pretraining strength	0.4	pretraining steps	20000
extrinsic strength	1.5	extrinsic gamma	0.99
curiosity strength	0.03	curiosity gamma	0.99
curiosity encoding size	128		
gail strength	0.03	gail gamma	0.99
gail encoding size	128	gail use actions	true

Demo path: demos/Frames20Demo.demo
Frames: 0, 5, 10, 12, 15, 17, 20, 22, 25, 27, 30, 32, 35, 40, 45, 50, 55, 60, 70, 80

Render Target Sensor: 20 * (20x20)
Grayscale: true
Observation space size = 0
Action space size = 2

Episode steps = 1000
Total steps = 308000
Time = 25000s

Improved the trajectory of the averages, also in the demo.

The training is slow but with better proportion than the previous one (probably because the demo occupies less space). It manages to learn certain things but it is necessary to refine more how it adapts to the movements (both impulses and coherent).

5_14_0952

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	8192	epsilon	0.2
hidden units	256	learning rate	2.0e-3
learning rate schedule	linear	max steps	6.0e5
normalize	true	num epoch	6
num layers	1	summary freq	1000
time horizon	64		
pretraining strength	**0.5**	pretraining steps	20000
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.01	curiosity gamma	0.99
curiosity encoding size	128		
gail strength	0.01	gail gamma	0.99
gail encoding size	128		

Demo path: demos/Frames20Demo.demo
Frames: 0, 5, 10, 12, 15, 17, 20, 22, 25, 27, 30, 32, 35, 40, 45, 50, 55, 60, 70, 80

Render Target Sensor: 20 * (20x20)
Grayscale: true
Observation space size = 0
Action space size = 2

Episode steps = 1000
Total steps = 600000
Time = 18450s

It greatly improves the reward curve, perhaps because of the simplification of the frames (individually). Adapts to various reactions at the right time but has too much noise. It may be better to change the reward system by adapting it to the time dimension and have the agent take only one action again (and not discount it by how far it is, but by the average). It also greatly improves the SAC times, although the latter could have better results in the very long term. It is considered as good but the method will be changed. On the other hand, it should be noted that this model does not handle well the targets that appear on the right side.

After changing back to the previous 1 action method, train the model with pure behavioral cloning.

5_14_2010

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	8192	epsilon	0.2
hidden units	256	learning rate	2.0e-3
learning rate schedule	linear	max steps	6.0e5
normalize	true	num epoch	6
num layers	1	summary freq	1000
time horizon	64		
pretraining strength	0.5	pretraining steps	20000
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.01	curiosity gamma	0.99
curiosity encoding size	128		
gail strength	0.01	gail gamma	0.99
gail encoding size	128		

Demo path: demos/OneLineMem.demo
Frames: 0, 5, 10, 12, 15, 17, 20, 22, 25, 27, 30, 32, 35, 40, 45, 50, 55, 60, 70, 80

Render Target Sensor: 20 * (20x20)
Grayscale: true
Observation space size = 25
Action space size = 1

Episode steps = 1000
Total steps = 378000
Time = 12000s

Simple training based on tolerable range of 0.05.

As expected, it works the same as before: it adapts to the average but does not take impulses. The reward should be modified to accommodate the time difference (and the parameter of future rewards).

5_15_1019

Trainer: BC

batches per epoch	10	batch size	64
hidden units	128	learning rate	3.0e-4
max steps	5.0e4	memory size	256
num layers	2	sequence length	32
summary freq	1000	use recurrent	false

Demo path: demos/BCDemo.demo
Frames: 0, 5, 10, 12, 15, 17, 20, 22, 25, 27, 30, 32, 35, 40, 45, 50, 55, 60, 70, 80

Render Target Sensor: 20 * (20x20)
Grayscale: true
Observation space size = 25
Action space size = 1

Episode steps = 1000
Total steps = 9000
Time = 3200s

Behavioral cloning training. It begins with very good rewards but the model becomes unbalanced after a few iterations. Well trained one could say that it needs very few steps, but it is very slow in number of steps per minute.

5_15_1108

Trainer: BC

batches per epoch	10	batch size	64
hidden units	**256**	**learning rate**	**5.0e-5**
max steps	5.0e4	memory size	256
num layers	**1**	sequence length	32
summary freq	1000	use recurrent	false

Demo path: demos/BCDemo.demo
Frames: 0, 5, 10, 12, 15, 17, 20, 22, 25, 27, 30, 32, 35, 40, 45, 50, 55, 60, 70, 80

Render Target Sensor: 20 * (20x20)
Grayscale: true
Observation space size = 25
Action space size = 1

Episode steps = 1000
Total steps = 16000
Time = 5000s

Changes in architecture and learning rate. The reward decreases but more slowly. Next, train at a lower learning rate and with the previous layers.

5_15_1237

Trainer: BC

batches per epoch	10	batch size	64
hidden units	**128**	**learning rate**	**1.0e-5**
max steps	5.0e4	memory size	256
num layers	**2**	sequence length	32
summary freq	1000	use recurrent	false

Demo path: demos/BCDemo.demo
Frames: 0, 5, 10, 12, 15, 17, 20, 22, 25, 27, 30, 32, 35, 40, 45, 50, 55, 60, 70, 80

Render Target Sensor: 20 * (20x20)
Grayscale: true
Observation space size = 25
Action space size = 1

Episode steps = 1000
Total steps = 30000
Time = 10500s

Training with even less learning rate. I have noticed that in the beginning he does not do impulses anymore, it would be good to record some of the previous models at the beginning of the training to illustrate the memory (when he takes impulses).

It is considered good but still fails on lateral impulses.

5_15_1703

Trainer: BC

batches per epoch	10	batch size	64
hidden units	128	learning rate	1.0e-5
max steps	5.0e4	memory size	256
num layers	2	sequence length	32
summary freq	1000	use recurrent	false

Demo path: demos/BCDemo.demo
Frames: 0, 5, 10, 12, 15, 17, 20, 22, 25, 27, 30, 32, 35, 40, 45, 50, 55, 60, 70, 80

Render Target Sensor: 20 * (20x20)
Grayscale: true
Observation space size = 25
Action space size = 1

Episode steps = 1000
Total steps = 25000
Time = 8700s

Training with Online BC. The results are slightly better, since they do not depend on an external dataset but on live action. Training times are equivalent. It still takes a long time to learn the impulses on the right side.

5_15_1932

Trainer: BC

batches per epoch	10	batch size	64
hidden units	128	learning rate	1.0e-5
max steps	5.0e4	memory size	256
num layers	2	sequence length	32
summary freq	1000	use recurrent	false

Demo path: demos/BCDemo.demo

Frames: 0, 5, 10, 12, 15, 17, 20, 22, 25, 27, 30, 32, 35, 40, 45, 50, 55, 60, 70, 80

Render Target Sensor: 20 * (20x20)

Grayscale: true

Observation space size = 25

Action space size = 1

Episode steps = 1000

Total steps = 36000

Time = 12000s

Online training with the learning rate of 5_15_1237. It follows the same curve and ends up being chaotic.

5_21_1008

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	8192	epsilon	0.2
hidden units	256	learning rate	2.0e-3
learning rate schedule	linear	max steps	6.0e5
normalize	true	num epoch	6
num layers	1	summary freq	1000
time horizon	64		
pretraining strength	0.5	pretraining steps	20000
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.01	curiosity gamma	0.99
curiosity encoding size	128		
gail strength	0.01	gail gamma	0.99
gail encoding size	128		

Demo path: demos/QueueDemo.demo

Frames: 0, 5, 10, 12, 15, 17, 20, 22, 25, 27, 30, 32, 35, 40, 45, 50, 55, 60, 70, 80

Render Target Sensor: 20 * (20x20)

Grayscale: true

Observation space size = 25

Action space size = 1

Episode steps = 1000

Total steps = 180000

Time = 5900s

Training with a movement queue. It is important that the parameter of future rewards (gamma) is high. Training may be flawed by how the average and deviation is calculated right now.

Goes up slowly. It has failed at 180000 steps.

5_21_1242

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	8192	epsilon	0.2
hidden units	256	learning rate	2.0e-3
learning rate schedule	linear	max steps	6.0e5
normalize	true	num epoch	6
num layers	1	summary freq	1000
time horizon	64		
pretraining strength	0.5	pretraining steps	20000
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.01	curiosity gamma	0.99
curiosity encoding size	128		
gail strength	0.01	gail gamma	0.99
gail encoding size	128		

Demo path: demos/QueueDemo.demo

Frames: 0, 5, 10, 12, 15, 17, 20, 22, 25, 27, 30, 32, 35, 40, 45, 50, 55, 60, 70, 80

Render Target Sensor: 20 * (20x20)

Grayscale: true

Observation space size = 25

Action space size = 1

Episode steps = 1000

Total steps = 493000

Time = 15700s

Changes: normal movement does not score, any failed or overdone impulses discount, and successful impulses add up (more than failures).

Similar to previous models trained with PPO, it tends to average out but does not learn the impulses.

The score could be improved if impulses could be treated separately (or movements in general).

5_21_1714

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	4.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	256	normalize	true
num update	1	train interval	5
num layers	1	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	simple		
pretraining strength	0.4	pretraining steps	20000
extrinsic strength	1.5	extrinsic gamma	0.99
curiosity strength	0.03	curiosity gamma	0.99
curiosity encoding size	128		
gail strength	0.03	gail gamma	0.99
gail encoding size	128	gail use actions	true

Demo path: demos/QueueDemo.demo

Frames: 0, 5, 10, 12, 15, 17, 20, 22, 25, 27, 30, 32, 35, 40, 45, 50, 55, 60, 70, 80

Render Target Sensor: 20 * (20x20)

Grayscale: true

Observation space size = 25

Action space size = 1

Episode steps = 1000

Total steps = 254000

Time = 21000s

Training with SAC, in the same build as the previous one. Factors may need to be increased as it rarely receives rewards.

The graph converges quite well to the average, retaining some noise, but not reacting to any impulse. The reward is -0.34. Curiously, almost all of the score increase has been between 190-200k.

Penalties need to be given much more weight, and perhaps even more weight to the extra impulses from the net.

5_22_1045

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	4.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	256	normalize	true
num update	1	train interval	5
num layers	1	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	simple		
pretraining strength	0.4	pretraining steps	20000
extrinsic strength	1.5	extrinsic gamma	0.99
curiosity strength	0.03	curiosity gamma	0.99
curiosity encoding size	128		
gail strength	0.03	gail gamma	0.99
gail encoding size	128	gail use actions	true

Demo path: demos/QueueDemo.demo

Frames: 0, 5, 10, 12, 15, 17, 20, 22, 25, 27, 30, 32, 35, 40, 45, 50, 55, 60, 70, 80

Render Target Sensor: 20 * (20x20)

Grayscale: true

Observation space size = 25

Action space size = 1

Episode steps = 1000

Total steps = 255000

Time = 20750s

Rewards multiplied.

The rewards follow strange patterns, and the movement forms a kind of frequency. I think what it does is it accumulates little impulses to add up to a real one. I should better redefine how to score the impulses and what is considered impulse or not (and the standard deviation should be increased or made fixed).

In the end it stabilizes at the average, like most previous models.

5_23_1036

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	4.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	256	normalize	true
num update	1	train interval	5
num layers	1	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	simple		
pretraining strength	0.4	pretraining steps	20000
extrinsic strength	1.5	extrinsic gamma	0.99
curiosity strength	0.03	curiosity gamma	0.99
curiosity encoding size	128		
gail strength	0.03	gail gamma	0.99
gail encoding size	128	gail use actions	true

Demo path: demos/QueueDemo.demo

Frames: 0, 5, 10, 12, 15, 17, 20, 22, 25, 27, 30, 32, 35, 40, 45, 50, 55, 60, 70, 80

Render Target Sensor: 20 * (20x20)

Grayscale: true

Observation space size = 25

Action space size = 1

Episode steps = 1000

Total steps = 320000

Time = 27500s

Changes in the reward, now eliminates whole impulses if it fails at first. It doesn't react to impulses after many hours. He may not have enough incentive to try.

The factors are: 1500 reward and 750 penalty (in general).

5_23_1825

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	4.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	256	normalize	true
num update	1	train interval	5
num layers	1	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	simple		
pretraining strength	0.4	pretraining steps	20000
extrinsic strength	1.5	extrinsic gamma	0.99
curiosity strength	0.03	curiosity gamma	0.99
curiosity encoding size	128		
gail strength	0.03	gail gamma	0.99
gail encoding size	128	gail use actions	true

Demo path: demos/QueueDemo.demo

Frames: 0, 5, 10, 12, 15, 17, 20, 22, 25, 27, 30, 32, 35, 40, 45, 50, 55, 60, 70, 80

Render Target Sensor: 20 * (20x20)

Grayscale: true

Observation space size = 25

Action space size = 1

Episode steps = 1000

Total steps = 203000

Time = 17000s

Reward=2000, fail=-500, miss=-1500. It has been corrected that it penalizes the final part of an impulse (if it succeeds, it does not continue to penalize).

It still doesn't have enough incentive to make impulses. Next time try with PPO and new parameters, and maybe also improve the reward system.

5_24_1625

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	8192	epsilon	0.2
hidden units	256	learning rate	2.0e-3
learning rate schedule	linear	max steps	6.0e5
normalize	true	num epoch	6
num layers	1	summary freq	1000
time horizon	64		
pretraining strength	0.5	pretraining steps	20000
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.01	curiosity gamma	0.99
curiosity encoding size	128		
gail strength	0.01	gail gamma	0.99
gail encoding size	128		

Demo path: demos/QueueDemo.demo

Frames: 0, 5, 10, 12, 15, 17, 20, 22, 25, 27, 30, 32, 35, 40, 45, 50, 55, 60, 70, 80

Render Target Sensor: 20 * (20x20)

Grayscale: true

Observation space size = 25

Action space size = 1

Episode steps = 1000

Total steps = 260000

Time = 8300s

PPO training. Reward=500, fail=-150, miss=-1500. Higher learning rate.

5_24_1850

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	8192	epsilon	0.2
hidden units	256	learning rate	2.0e-3
learning rate schedule	linear	max steps	6.0e5
normalize	true	num epoch	6
num layers	1	summary freq	1000
time horizon	64		
pretraining strength	0.5	pretraining steps	20000
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.01	curiosity gamma	0.99
curiosity encoding size	128		
gail strength	0.01	gail gamma	0.99
gail encoding size	128		

Demo path: demos/QueueDemo.demo
Frames: 0, 5, 10, 12, 15, 17, 20, 22, 25, 27, 30, 32, 35, 40, 45, 50, 55, 60, 70, 80

Render Target Sensor: 20 * (20x20)
Grayscale: true
Observation space size = 25
Action space size = 1

Episode steps = 1000
Total steps = 323000
Time = 10750s

PPO training. Reward=750, fail=-250, miss=-1500. It does not perform impulses.

5_24_2154

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	8192	epsilon	0.2
hidden units	256	learning rate	2.0e-3
learning rate schedule	linear	max steps	6.0e5
normalize	true	num epoch	6
num layers	1	summary freq	1000
time horizon	64		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.01	curiosity gamma	0.99
curiosity encoding size	128		

Frames: 0, 5, 10, 12, 15, 17, 20, 22, 25, 27, 30, 32, 35, 40, 45, 50, 55, 60, 70, 80

Render Target Sensor: 20 * (20x20)
Grayscale: true
Observation space size = 25
Action space size = 1

Episode steps = 1000
Total steps = 213000
Time = 5800s

PPO training. Reward=250, fail=-250, miss=-5000. Penalty for not performing impulses taken to the extreme. The program failed but the behavior didn't improve either.

5_25_1607

Trainer: BC

batches per epoch	10	batch size	64
hidden units	128	learning rate	5.0e-5
max steps	5.0e4	memory size	256
num layers	2	sequence length	32
summary freq	1000	use recurrent	false

Demo path: demos/BCDemo.demo
Frames: 0, 5, 10, 12, 15, 17, 20, 22, 25, 27, 30, 32, 35, 40, 45, 50, 55, 60, 70, 80

Render Target Sensor: 20 * (20x20)
Grayscale: true
Observation space size = 25
Action space size = 1

Episode steps = 1000
Total steps = 13000
Time = 4500s

Online training. A deterministic bot is used without any noise (constant movement at 0.3 and impulses at 0.2 seconds). It gets worse with time, although it doesn't start badly (if it catches just one impulse).

5_25_1730

Trainer: BC

batches per epoch	10	batch size	64
hidden units	128	**learning rate**	**1.0e-5**
max steps	5.0e4	memory size	256
num layers	2	sequence length	32
summary freq	1000	use recurrent	false

Demo path: demos/UniformDemo.demo
Frames: 0, 5, 10, 12, 15, 17, 20, 22, 25, 27, 30, 32, 35, 40, 45, 50, 55, 60, 70, 80

Render Target Sensor: 20 * (20x20)
Grayscale: true
Observation space size = 25
Action space size = 1

Episode steps = 1000
Total steps = 27000
Time = 8000s

Changed the demo and the learning rate.
The reward decreases constantly, and the behavior gets worse, although in some specific moments it gave good results.

5_25_1958

Trainer: BC

batches per epoch	10	batch size	64
hidden units	128	learning rate	1.0e-5
max steps	5.0e4	memory size	256
num layers	2	sequence length	32
summary freq	1000	use recurrent	false

Demo path: demos/UniformDemo.demo
Frames: 0, 5, 10, 12, 15, 17, 20, 22, 25, 27, 30, 32, 35, 40, 45, 50, 55, 60, 70, 80

Render Target Sensor: 20 * (20x20)
Grayscale: true
Observation space size = 25
Action space size = 1

Episode steps = 1000
Total steps = 15000
Time = 4500s

Now checks if there's a non-null target. There's been a bot failure, it's repeated.

5_25_2157

Trainer: BC

batches per epoch	10	batch size	64
hidden units	128	learning rate	1.0e-5
max steps	5.0e4	memory size	256
num layers	2	sequence length	32
summary freq	1000	use recurrent	false

Demo path: demos/UniformDemo.demo
Frames: 0, 5, 10, 12, 15, 17, 20, 22, 25, 27, 30, 32, 35, 40, 45, 50, 55, 60, 70, 80

Render Target Sensor: 20 * (20x20)
Grayscale: true
Observation space size = 25
Action space size = 1

Episode steps = 1000
Total steps = 46000
Time = 13300s

Several corrections to the bot's behavior (at some point it stopped reacting to targets, because the closest variable didn't restart).

It gets worse over time, making very strange effects after pulling several impulses. It is saved to test it in execution.

It does a good continuous movement, but when there's a target between the previous frames it oscillates in strange ways, although it keeps moving in the direction it should (as a whole). PPO, or more complex layers with imitation learning, should be tried.

5_26_1059

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	8192	epsilon	0.2
hidden units	256	learning rate	2.0e-3
learning rate schedule	linear	max steps	6.0e5
normalize	true	num epoch	6
num layers	1	summary freq	1000
time horizon	64		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.01	curiosity gamma	0.99
curiosity encoding size	128		

Frames: 0, 5, 10, 12, 15, 17, 20, 22, 25, 27, 30, 32, 35, 40, 45, 50, 55, 60, 70, 80

Render Target Sensor: 20 * (20x20)
Grayscale: true
Observation space size = 25
Action space size = 1

Episode steps = 1000
Total steps = 122000
Time = 4600s

Training with a single PPO thread (not multi) It should get better results since the movement is deterministic. The real bot does strange behaviors so it is stopped.

5_26_1225

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	8192	epsilon	0.2
hidden units	256	learning rate	2.0e-3
learning rate schedule	linear	max steps	6.0e5
normalize	true	num epoch	6
num layers	1	summary freq	1000
time horizon	64		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.01	curiosity gamma	0.99
curiosity encoding size	128		

Frames: 0, 5, 10, 12, 15, 17, 20, 22, 25, 27, 30, 32, 35, 40, 45, 50, 55, 60, 70, 80

Render Target Sensor: 20 * (20x20)
Grayscale: true
Observation space size = 25
Action space size = 1

Episode steps = 1000
Total steps = 200000
Time = 7750s

By setting the distance of closest to 2.0f, it reacted earlier (corrected). It does not seem to converge with PPO.

5_26_1507

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	8192	epsilon	0.2
hidden units	256	learning rate	2.0e-3
learning rate schedule	linear	max steps	6.0e5
normalize	true	num epoch	6
num layers	1	summary freq	1000
time horizon	64		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.01	curiosity gamma	0.99
curiosity encoding size	128		

Frames: 0, 5, 10, 12, 15, 17, 20, 22, 25, 27, 30, 32, 35, 40, 45, 50, 55, 60, 70, 80

Render Target Sensor: 20 * (20x20)
Grayscale: true
Observation space size = 25
Action space size = 1

Episode steps = 1000
Total steps = 460000
Time = 17200s

Use of simpler linear rewards, assuming the movement is deterministic and has no noise.

The reward grows slowly, and catches the impulses more or less well but may lack the complexity of the network structure to learn well.

5_26_1958

Trainer: BC

batches per epoch	10	batch size	64
hidden units	128	learning rate	1.0e-5
max steps	5.0e4	**memory size**	**32**
num layers	**3**	**sequence length**	**128**
summary freq	1000	**use recurrent**	**true**

Demo path: demos/UniformDemo.demo
Frames: 0, 5, 10, 12, 15, 17, 20, 22, 25, 27, 30, 32, 35, 40, 45, 50, 55, 60, 70, 80

Render Target Sensor: 20 * (20x20)
Grayscale: true
Observation space size = 25
Action space size = 1

Episode steps = 1000
Total steps = 12000
Time = 13000s

More complex network with imitation learning. The training time is very slow, but the results are good. However, the model loads the CPU a lot when inferring, decreasing the fps by half (25-30). When using recurrent memory, it may not be necessary to save previous movements and images.

As a bad part, from time to time random impulses appear without any target in sight, they may disappear with more training or they may occur due to lack of complexity (react time is 0.25).

In other experiments, different architectures could be tested using recurrent (1, 2 or 4 layers, with more or less hidden units), and some without more inputs than the current image (and maybe the previous motion, depending on how the recurrent memory works).

5_27_0953

Trainer: BC

batches per epoch	10	batch size	64
hidden units	128	learning rate	1.0e-5
max steps	5.0e4	memory size	32
num layers	3	sequence length	128
summary freq	1000	use recurrent	true

Demo path: demos/UniformNoRTDemo.demo

Render Target Sensor: 1 * (20x20)
Grayscale: true
Observation space size = 25
Action space size = 1

Episode steps = 1000
Total steps = 18000
Time = 16500s

From tests carried out, it seems that BC cannot be used with a vector observation of size 0 (I have left it at 25 as it was). It's also important to point out that it is necessary to turn off the demonstration recorder to train, and that it is possible to train online with PPO and SAC (I suppose that in a single thread).

Compare the training speed (and results) with the previous model, and the FPS when running it.

The result looks good, at the expense of testing it. It has the typical random impulses, we will have to see if it is more efficient too.

In the editor it goes crazy after a few seconds, maybe because it receives the empty previous movement vector. Try to do it with less movements.

Reaction 0.25.

5_27_1656

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	8192	epsilon	0.2
hidden units	256	learning rate	2.0e-3
learning rate schedule	linear	max steps	6.0e5
memory size	**32**	normalize	true
num epoch	6	num layers	1
sequence length	**128**	summary freq	1000
time horizon	64	**use recurrent**	**true**
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.01	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 1 * (20x20)
Grayscale: true
Observation space size = 1
Action space size = 1

Episode steps = 1000
Total steps = 70000
Time = 1550s

An enumeration has been created to support more types of observations, in this case it only receives the last movement. Attempting to train with imitation learning does not work using only one observation (with 25).

In this training we use PPO with recurrent memory, and the last movement as visual input. It is done online. However, it has less layers than the previous ones with imitation learning.

I cancel it to try with SAC, it doesn't seem to improve the previous PPO results.

5_27_1727

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	**hidden units**	**128**
init entcoef	1.0	learning rate	4.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	**32**	normalize	true
num update	1	train interval	5
num layers	**2**	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	**use recurrent**	**true**
vis encode type	simple		
pretraining strength	0.4	pretraining steps	20000
extrinsic strength	1.5	extrinsic gamma	0.99
curiosity strength	0.03	curiosity gamma	0.99
curiosity encoding size	128		
gail strength	0.03	gail gamma	0.99
gail encoding size	128	gail use actions	true

Demo path: demos/UniformNoRT1Demo.demo

Render Target Sensor: 1 * (20x20)

Grayscale: true

Observation space size = 1

Action space size = 1

Episode steps = 1000

Total steps = 235000

Time = 24000s

Online training with SAC and 2 layers in the network.

It looked like it was going to converge to the average, but in the end it has started to make very irregular impulses. It may work with more network complexity.

5_28_1240

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	128
init entcoef	1.0	learning rate	4.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	true
vis encode type	simple		
pretraining strength	0.4	pretraining steps	20000
extrinsic strength	1.5	extrinsic gamma	0.99
curiosity strength	0.03	curiosity gamma	0.99
curiosity encoding size	128		
gail strength	0.03	gail gamma	0.99
gail encoding size	128	gail use actions	true

Demo path: demos/UniformNoRT1Demo.demo

Render Target Sensor: 1 * (20x20)
Grayscale: true
Observation space size = 1
Action space size = 1

Episode steps = 1000
Total steps = -
Time = -

–Tests
Training with higher requirements in rewards and more layers.

Does not work with 1 observation and 2 or 3 layers. The error seems to have to do with sequence length, I don't know if it should not be greater than the hidden layer units or if the relationship between memory size and sequence length should be kept to a minimum. I'm going to do some tests (with sequence length=128, like the hidden units, it works with an observation). The curious thing is that the failures were in step 5000.

sl=256, hu=128, ms=32: FAILS
sl=256, hu=256, ms=32: FAILS, must not be for equality with hidden units
sl=256, hu=128, ms=64: FAILS
sl=128, hu=128, ms=16: DOES NOT FAIL

For some reason it always fails when sequence length is greater than or equal to 256, the buffer size is also not the key. Therefore, the most that can be used is 128.

Could the encoding size of curiosity and gail be the cause, since both were 128? Test it later.

5_28_1316

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	128
init entcoef	1.0	learning rate	4.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	**3**	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	true
vis encode type	simple		
pretraining strength	0.4	pretraining steps	20000
extrinsic strength	1.5	extrinsic gamma	0.99
curiosity strength	0.03	curiosity gamma	0.99
curiosity encoding size	128		
gail strength	0.03	gail gamma	0.99
gail encoding size	128	gail use actions	true

Demo path: demos/UniformNoRT1Demo.demo

Render Target Sensor: 1 * (20x20)
Grayscale: true
Observation space size = 1
Action space size = 1

Episode steps = 1000
Total steps = 272000
Time = 28000s

It didn't look bad after about 150000+ steps, it went from converging to average to start making impulses (with some noise), but then it started to go crazy. However, the reward has increased, so I think there may be a problem that makes the rewards not very well balanced. Either that, or it may require some additional layer.

I've checked that sequence length also fails with equal encoding sizes, so it can't be 256 in general.

5_28_2108

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	128
init entcoef	1.0	learning rate	4.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	**4**	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	true
vis encode type	simple		
pretraining strength	0.4	pretraining steps	20000
extrinsic strength	1.5	extrinsic gamma	0.99
curiosity strength	0.03	curiosity gamma	0.99
curiosity encoding size	128		
gail strength	0.03	gail gamma	0.99
gail encoding size	128	gail use actions	true

Demo path: demos/UniformNoRT1Demo.demo

Render Target Sensor: 1 * (20x20)
Grayscale: true
Observation space size = 1
Action space size = 1

Episode steps = 1000
Total steps = 100000
Time = 10300s

One more layer in the network. I stop the training but it seems to be following the same path as the previous one. I could try again by combining the use of recurrent and the previous images.

Despite having one more layer, it seems to have no added cost. It would be nice to continue with this same training (–load)

5_29_0926

Trainer: BC

batches per epoch	10	batch size	64
hidden units	128	learning rate	1.0e-5
max steps	5.0e4	memory size	32
num layers	3	sequence length	128
summary freq	1000	use recurrent	true

Demo path: demos/UniformNoRT1Demo.demo

Render Target Sensor: 1 * (20x20)
Grayscale: true
Observation space size = 25
Action space size = 1

Episode steps = 1000
Total steps = 18000
Time = 17200s

BC training using an input. The rest of the parameters are the same as in the last training that worked.

Testing if it works well in the editor, the result seemed correct (at least with the ones on the left).

In editor it doesn't work well after encountering a left-hand pulse: when it stands still it enters untrained cases.

5_29_1504

Trainer: BC

batches per epoch	10	batch size	64
hidden units	128	learning rate	1.0e-5
max steps	5.0e4	memory size	32
num layers	**2**	sequence length	128
summary freq	1000	use recurrent	true

Demo path: demos/UniformNoRT1Demo.demo

Render Target Sensor: 1 * (20x20)
Grayscale: true
Observation space size = 25
Action space size = 1

Episode steps = 1000
Total steps = 23000
Time = 21500s

BC training and one less layer. It's very similar to the previous one.

In the game it stops working in about 2 seconds, when it starts spinning uncontrollably.

5_29_2126

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	128
init entcoef	1.0	learning rate	4.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	**2**	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	true
vis encode type	simple		
pretraining strength	0.4	pretraining steps	20000
extrinsic strength	1.5	extrinsic gamma	0.99
curiosity strength	0.03	curiosity gamma	0.99
curiosity encoding size	128		
gail strength	0.03	gail gamma	0.99
gail encoding size	128	gail use actions	true

Demo path: demos/UniformNoRT1Demo.demo

Render Target Sensor: 1 * (20x20)

Grayscale: true

Observation space size = 1

Action space size = 1

Episode steps = 1000

Total steps = 90000

Time = 9000s

Training with adjusted rewards (squared). Although the demo has not been changed, so it will have old rewards there. In this first test we only use 2 layers.

It's close to average, but it doesn't make any impulses.

5_31_1929

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	128
init entcoef	1.0	**learning rate**	**7.5e-4**
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	**3**	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	true
vis encode type	simple		
pretraining strength	0.4	pretraining steps	20000
extrinsic strength	1.5	extrinsic gamma	0.99
curiosity strength	0.03	curiosity gamma	0.99
curiosity encoding size	128		
gail strength	0.03	gail gamma	0.99
gail encoding size	128	gail use actions	true

Demo path: demos/UniformNoRT1Demo.demo

Render Target Sensor: 1 * (20x20)

Grayscale: true

Observation space size = 1

Action space size = 1

Episode steps = 1000

Total steps = 153000

Time = 15700s

Higher learning rate. In following trainings it would be good to base the rewards on coherence (adapted to a single input). It has grown faster but falls back after 90k steps. It doesn't get to make impulses as far as I've seen, it would be convenient to search among previous models to readapt them.

6_01_1050

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	128
init entcoef	1.0	**learning rate**	**5.0e-4**
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	3	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	true
vis encode type	simple		
pretraining strength	0.4	pretraining steps	20000
extrinsic strength	**1.0**	extrinsic gamma	0.99
curiosity strength	0.03	curiosity gamma	0.99
curiosity encoding size	128		
gail strength	0.03	gail gamma	0.99
gail encoding size	128	gail use actions	true

Demo path: demos/UniformNoRT1Demo.demo

Render Target Sensor: 1 * (20x20)

Grayscale: true

Observation space size = 1

Action space size = 1

Episode steps = 1000

Total steps = 190000

Time = 19700s

Training with higher gail and curiosity weight (in ratio) and reward factor of 300 (against 10). Seems to be stagnating on the same behaviors.

6_01_1623

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	128
init entcoef	1.0	learning rate	5.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	3	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	true
vis encode type	simple		
pretraining strength	0.4	pretraining steps	20000
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	**0.1**	curiosity gamma	0.99
curiosity encoding size	128		
gail strength	0.03	gail gamma	0.99
gail encoding size	128	gail use actions	true

Demo path: demos/UniformNoRT1Demo.demo

Render Target Sensor: 1 * (20x20)

Grayscale: true

Observation space size = 1

Action space size = 1

Episode steps = 1000

Total steps = 305000

Time = 32000s

It goes up until it adapts to the average (without impulses) but then it goes down again, it is not useful to use recurrent memory in these cases without other information (it might work but with a much more perfected system of rewards). Next time try the combination of render targets and recurrent but with SAC (and a single motion input).

6_02_1023

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	128
init entcoef	1.0	learning rate	5.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	3	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	true
vis encode type	simple		
pretraining strength	0.4	pretraining steps	20000
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		
gail strength	0.03	gail gamma	0.99
gail encoding size	128	gail use actions	true

Frames: 0, 5, 10, 12, 15, 17, 20, 22, 25, 27, 30, 32, 35, 40, 45, 50, 55, 60, 70, 80

Demo path: demos/Frames20Demo.demo

Render Target Sensor: 20 * (20x20)

Grayscale: true

Observation space size = 1

Action space size = 1

Episode steps = 1000

Total steps = 90000

Time = 18000s

Training with 20 render targets, the demo has also been updated. The objective is to see if it improves the performance with respect to the one trained with SAC. It is still only adjusted to the average, it doesn't seem to have any incentive to make impulses. In the next training I will use as input the movements of the bot to see how long it takes to learn (if it does), if it does not learn, change reward system (even to TR) and if it learns, check the reward in which it behaves decently to incorporate learning curriculum.

6_02_1545

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	128
init entcoef	1.0	learning rate	5.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	3	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	true
vis encode type	simple		
pretraining strength	0.4	pretraining steps	20000
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		
gail strength	0.03	gail gamma	0.99
gail encoding size	128	gail use actions	true

Frames: 0, 5, 10, 12, 15, 17, 20, 22, 25, 27, 30, 32, 35, 40, 45, 50, 55, 60, 70, 80

Demo path: demos/Frames20Demo.demo

Render Target Sensor: 20 * (20x20)

Grayscale: true

Observation space size = 1

Action space size = 1

Episode steps = 1000

Total steps = 86000

Time = 17600s

Receives actual values from the bot. Doesn't react anyway.

Add noise in next tests.

6_02_2054

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	128
init entcoef	1.0	learning rate	5.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	3	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	true
vis encode type	simple		
pretraining strength	0.4	pretraining steps	20000
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		
gail strength	0.03	gail gamma	0.99
gail encoding size	128	gail use actions	true

Frames: 0, 5, 10, 12, 15, 17, 20, 22, 25, 27, 30, 32, 35, 40, 45, 50, 55, 60, 70, 80

Demo path: demos/R0N02.demo

Render Target Sensor: 20 * (20x20)

Grayscale: true

Observation space size = 1

Action space size = 1

Episode steps = 1000

Total steps = 55000

Time = 9500s

Parameters in the bot (reaction=0, noise=0.2) and rewards with tolerable range (TR=0.05, MR=1), as a reference point to see if it works. The demo has an identifying name. This model doesn't work because there was an error in the tolerable range rewards that made it always give 0 as a punishment.

6__02__2341

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	128
init entcoef	1.0	learning rate	5.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	3	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	true
vis encode type	simple		
pretraining strength	0.4	pretraining steps	20000
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		
gail strength	0.03	gail gamma	0.99
gail encoding size	128	gail use actions	true

Frames: 0, 5, 10, 12, 15, 17, 20, 22, 25, 27, 30, 32, 35, 40, 45, 50, 55, 60, 70, 80

Demo path: demos/R0N02.demo

Render Target Sensor: 20 * (20x20)

Grayscale: true

Observation space size = 1

Action space size = 1

Episode steps = 1000

Total steps = 100000

Time = 7300s + 11000s

Same parameters but corrected. The demo is redone because the previous one has incorrect rewards. Also the noise has been reduced to 0.05 to enter the tolerable range.

Training interrupted, continue.

Next models could be trained with PPO, as it should be faster in reaching a certain level. It doesn't seem to go too high, and it may be caused by the resolution of the render targets (by discard). In the next test PPO.

6_03_1241

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	8192	epsilon	0.2
hidden units	256	learning rate	2.0e-3
learning rate schedule	linear	max steps	6.0e5
memory size	32	normalize	true
num epoch	6	num layers	1
sequence length	128	summary freq	1000
time horizon	64	use recurrent	true
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.01	curiosity gamma	0.99
curiosity encoding size	128		

Frames: 0, 5, 10, 12, 15, 17, 20, 22, 25, 27, 30, 32, 35, 40, 45, 50, 55, 60, 70, 80

Render Target Sensor: 20 * (20x20)
Grayscale: true
Observation space size = 1
Action space size = 1

Episode steps = 1000
Total steps = 154000
Time = 7000s

PPO training with the parameters that were. No demo included.
Converge to the average, no impulses. Next I try with 64x64 render targets.

6_03_1454

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	8192	epsilon	0.2
hidden units	256	**learning rate**	**2.5e-4**
learning rate schedule	linear	max steps	6.0e5
memory size	32	**normalize**	**false**
num epoch	6	num layers	1
sequence length	128	summary freq	1000
time horizon	64	use recurrent	true
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.01	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 64*64

Grayscale: true
Observation space size = 1
Action space size = 1

Episode steps = 1000
Total steps = 600000
Time = 14000s

Change in render target, normalize and learning rate. If it works well it serves as a reference, if not there is a problem with another parameter or the reward system.

It didn't work. Next time I'll try with rewards with coherence (that I should create).

6_05_1205

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	128
init entcoef	1.0	learning rate	5.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	3	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	true
vis encode type	simple		
pretraining strength	0.4	pretraining steps	20000
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		
gail strength	0.03	gail gamma	0.99
gail encoding size	128	gail use actions	true

Frames: 0, 5, 10, 12, 15, 17, 20, 22, 25, 27, 30, 32, 35, 40, 45, 50, 55, 60, 70, 80
Demo path: demos/Frames20Demo.demo

Render Target Sensor: 20 * (20x20)
Grayscale: true
Observation space size = 1
Action space size = 1

Episode steps = 1000
Total steps = 82000
Time = 15900s

First training after migrating repository. It has given some error at the beginning for not finding references (to something of python or ml agents).

Now I use rewards inside range (they are similar to TR, but they give maximum reward as long as it is within range).

It gets close to the average as expected.

6_05_1659

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	128
init entcoef	1.0	learning rate	5.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	3	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	true
vis encode type	simple		
pretraining strength	0.4	pretraining steps	20000
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		
gail strength	0.03	gail gamma	0.99
gail encoding size	128	gail use actions	true

Frames: 0, 5, 10, 12, 15, 17, 20, 22, 25, 27, 30, 32, 35, 40, 45, 50, 55, 60, 70, 80
Demo path: demos/Frames20Demo.demo

Render Target Sensor: 20 * (20x20)
Grayscale: true
Observation space size = 1
Action space size = 1

Episode steps = 1000
Total steps = 117000
Time = 25200s

Change in Rewards (StdCoherenceIndividual): Failing coherent moves penalizes, and succeeding inconsistent moves gives reward (penalizes only if done in the opposite direction). The average is a bit unstable but should be fine. In future models it would be good to use dynamic averaging and deviation.

It doesn't converge.

6_06_1433

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	8192	epsilon	0.2
hidden units	256	learning rate	2.5e-4
learning rate schedule	linear	max steps	6.0e5
memory size	32	normalize	false
num epoch	6	num layers	1
sequence length	128	summary freq	1000
time horizon	64	use recurrent	true
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.01	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 20 * (20x20)

Grayscale: true

Observation space size = 1

Action space size = 1

Episode steps = 1000

Total steps = 137000

Time = 6200s

Dynamic average and deviation implemented. The previous reward is used but now takes into account half of the deviation to start penalizing (that's how far it usually expands).

It tends to exploit the rewards by staying in the margin where the impulses usually appear. Needs more penalty.

6_06_1630

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	8192	epsilon	0.2
hidden units	256	learning rate	2.5e-4
learning rate schedule	linear	max steps	6.0e5
memory size	32	normalize	false
num epoch	6	num layers	1
sequence length	128	summary freq	1000
time horizon	64	use recurrent	true
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.01	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 1 * (20x20)
Grayscale: false
Observation space size = 1
Action space size = 1

Episode steps = 1000
Total steps = 487000
Time = 10000s

Slight changes in the linearity of rewards and factors Now I go back to NOT using grayscale in the rendering. In next tests I may put the previous 25 moves back.

Failed training by leaving the reaction time at 0.2 by mistake.

6_06_1933

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	8192	epsilon	0.2
hidden units	256	learning rate	2.5e-4
learning rate schedule	linear	max steps	6.0e5
memory size	32	normalize	false
num epoch	6	num layers	1
sequence length	128	summary freq	1000
time horizon	64	use recurrent	true
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.01	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 1 * (20x20)

Grayscale: false
Observation space size = 1
Action space size = 1

Episode steps = 1000
Total steps = 411000
Time = 9000s

Training with react time = 0, in theory it should converge, unless the rewards are not yet balanced.

It doesn't converge at all.

6_06_2319

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	8192	epsilon	0.2
hidden units	256	learning rate	2.5e-4
learning rate schedule	linear	max steps	6.0e5
memory size	32	normalize	false
num epoch	6	num layers	1
sequence length	128	summary freq	1000
time horizon	64	**use recurrent**	**false**
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.01	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 1 * (64x64)
Grayscale: false
Observation space size = 1
Action space size = 1

Episode steps = 1000
Total steps = 432000
Time = 9400s

This time I take away recurrent and double the reward factor (50 against 100).

It has failed a training (it has stopped in 8000 steps, but without giving error).

It has managed to follow impulses with great precision but it lacks a jump in the penalties so that it tends to the average in the coherent movements.

In addition, the buffer size may have to be decreased to avoid errors like the previous one (check how it was in previous successful models). Keep in mind that using the model as it is will probably cause it to diverge (since it has only trained with real values). In

the next training the result of using the bot values instead of the real ones could be compared.

6_07_1000

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	8192	epsilon	0.2
hidden units	256	learning rate	2.5e-4
learning rate schedule	linear	max steps	6.0e5
memory size	32	normalize	false
num epoch	6	num layers	1
sequence length	128	summary freq	1000
time horizon	64	use recurrent	false
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.01	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 1 * (64x64)

Grayscale: false

Observation space size = 1

Action space size = 1

Episode steps = 1000

Total steps = 235000

Time = 5000s

In this training it receives the observations of the bot itself. This allows you to compare the learning curve of both.

Remember to change the linearity of the rewards after this training, and the buffer size.

It follows exactly the same path as the previous model.

6_07_1139

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	**4096**	epsilon	0.2
hidden units	256	learning rate	2.5e-4
learning rate schedule	linear	max steps	6.0e5
memory size	32	normalize	false
num epoch	6	num layers	1
sequence length	128	summary freq	1000
time horizon	64	use recurrent	false
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.01	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 1 * (64x64)
Grayscale: false
Observation space size = 1
Action space size = 1

Episode steps = 1000
Total steps = 588000
Time = 12700s

More pressure has been added to the rewards, and 2 jumps in value. The parameters are still 50 reward and 100 punish, and the agent's observations. The buffer has also been changed. The result seems similar to the previous 2.

6_07_1515

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	4096	epsilon	0.2
hidden units	256	learning rate	2.5e-4
learning rate schedule	linear	max steps	6.0e5
memory size	32	normalize	false
num epoch	6	num layers	1
sequence length	128	summary freq	1000
time horizon	64	use recurrent	false
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.01	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 1 * (64x64)

Grayscale: false
Observation space size = 1
Action space size = 1

Episode steps = 1000
Total steps = 418000
Time = 9000s

Even more penalty pressure has been added (it now always penalizes inconsistent movement, unless it is exactly the same, and linearly increasing with distance). If it doesn't work, the reward/penalty factors will have to be adjusted. The other parameters are the same.

It works the same as the previous ones, that's why it is not saved.

6_07_1747

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	4096	epsilon	0.2
hidden units	256	learning rate	2.5e-4
learning rate schedule	linear	max steps	6.0e5
memory size	32	normalize	false
num epoch	6	num layers	1
sequence length	128	summary freq	1000
time horizon	64	use recurrent	false
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.01	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 1 * (64x64)
Grayscale: false
Observation space size = 1
Action space size = 1

Episode steps = 1000
Total steps = -
Time = -

Penalty factor at 150 and reward factor at 50.

Up to 230000 steps (-20 reward) follows all similar (makes perfect impulses but has a lot of noise in normal movement), so I'm going to increase the penalty to 250 at this point. The reward goes down to -42, if it evolves well from now on it may work as a learning curriculum, if it follows the same trend it will be better to change the form of

the penalties and/or use SAC (Also keep in mind that the learning rate has already gone down).

It does not change.

6_07_1947

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	128
init entcoef	1.0	learning rate	5.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	3	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	**use recurrent**	**false**
vis encode type	simple		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 1 * (64x64)
Grayscale: false
Observation space size = 1
Action space size = 1

Episode steps = 1000
Total steps = 316000
Time = 11000s

Training with SAC (without GAIL) and the same initial parameters as in the previous one (also without recurrent and with observations of the agent).

It has reached a good reward but after 107k steps it has fallen (it has remained flat). I think the problem is that the reward is also unbalanced, it should have minimal rewards for getting very close in coherent movements, so that a positive result can be achieved.

6_08_1054

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	128
init entcoef	1.0	learning rate	5.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	3	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	simple		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 1 * (64x64)

Grayscale: false

Observation space size = 1

Action space size = 1

Episode steps = 1000

Total steps = 161000

Time = 5200s

Training until I fail, to see the results of before.

It's much worse with SAC, I think that because the coherent movement has no positive reward, it doesn't distinguish when an impulse fails from when a coherent movement succeeds and ends up being fun. In the next one I'll add low rewards to see if it doesn't diverge.

6_08_1232

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	128
init entcoef	1.0	learning rate	5.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	3	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	simple		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 1 * (64x64)
Grayscale: false
Observation space size = 1
Action space size = 1

Episode steps = 1000
Total steps = 92000
Time = 3200s

Minimal rewards added. Still not converging, the next time I test PPO but with 3 layers.

6_08_1328

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	4096	epsilon	0.2
hidden units	256	learning rate	2.5e-4
learning rate schedule	linear	max steps	6.0e5
memory size	32	normalize	false
num epoch	6	**num layers**	**3**
sequence length	128	summary freq	1000
time horizon	64	use recurrent	false
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.01	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 1 * (64x64)
Grayscale: false
Observation space size = 1
Action space size = 1

Episode steps = 1000
Total steps = 600000
Time = 13000s

PPO with the new minimum rewards and 3 layers. It gets the same results as before.

6_08_1725

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	4096	epsilon	0.2
hidden units	256	**learning rate**	**1.5e-4**
learning rate schedule	**constant**	max steps	6.0e5
memory size	32	normalize	false
num epoch	6	**num layers**	**3**
sequence length	128	summary freq	1000
time horizon	64	use recurrent	false
extrinsic strength	1.0	**extrinsic gamma**	**0.8**
curiosity strength	**0.02**	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 1 * (64x64)
Grayscale: false
Observation space size = 1
Action space size = 1

Episode steps = 1000
Total steps = 76000
Time = 1650s

Very unbalanced exponential rewards.

6_08_1759

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	4096	epsilon	0.2
hidden units	256	learning rate	1.5e-4
learning rate schedule	constant	max steps	6.0e5
memory size	32	normalize	false
num epoch	6	num layers	3
sequence length	128	summary freq	1000
time horizon	64	use recurrent	false
extrinsic strength	1.0	extrinsic gamma	0.8
curiosity strength	0.02	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 1 * (64x64)
Grayscale: false
Observation space size = 1
Action space size = 1

Episode steps = 1000
Total steps = 600000
Time = 13000s

Rewards with decay from x^5 to 1 distance, then linear. It seems that the weight of the penalties stays very high now. The result is still similar, but the rewards are much lower.

6_08_2152

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	4096	epsilon	0.2
hidden units	256	learning rate	1.5e-4
learning rate schedule	constant	max steps	6.0e5
memory size	32	normalize	false
num epoch	6	num layers	3
sequence length	128	summary freq	1000
time horizon	64	use recurrent	false
extrinsic strength	1.0	extrinsic gamma	0.8
curiosity strength	0.02	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 1 * (64x64)

Grayscale: false
Observation space size = 1
Action space size = 1

Episode steps = 1000
Total steps = 600000
Time = 13000s

Reward=100, penalty=150. In addition, it receives observations from the bot only. Change decay if it still doesn't work.

Although it still has a lot of noise, the impulses are traced millimetrically (both when the network moves and when it imitates the bot), although it may be because it receives the bot's values as input (they are uniform) in spite of not really being working. If it works well we could start testing with some fixed delay frames (receiving all the following ones as observation) and then increase them.

6_09_1119

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	4096	epsilon	0.2
hidden units	256	learning rate	1.5e-4
learning rate schedule	constant	max steps	6.0e5
memory size	32	normalize	false
num epoch	6	num layers	3
sequence length	128	summary freq	1000
time horizon	64	use recurrent	false
extrinsic strength	1.0	extrinsic gamma	0.8
curiosity strength	0.02	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 1 * (64x64)
Grayscale: false
Observation space size = 1
Action space size = 1

Episode steps = 1000
Total steps = 585000
Time = 12750s

Noise from 0.05 to 0.15, and now receives observations from the agent instead of the bot.

I could try to put the average as a target, instead of the real value (in coherent movements).

The bot adapts to a level similar to the previous bots, the standard deviation is similar in spite of having more noise. You could smooth out the deviation somewhat to make it better suited to the movement (even though it's not exactly a standard deviation).

6_09_1503

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	4096	epsilon	0.2
hidden units	256	learning rate	1.5e-4
learning rate schedule	constant	max steps	6.0e5
memory size	32	normalize	false
num epoch	6	num layers	3
sequence length	128	summary freq	1000
time horizon	64	use recurrent	false
extrinsic strength	1.0	extrinsic gamma	0.8
curiosity strength	0.02	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 1 * (64x64)
Grayscale: false
Observation space size = 1
Action space size = 1

Episode steps = 1000
Total steps = 356000
Time = 7700s

Same parameters as before, but using a reward system that appreciates being closer to the average more than the real movement (it receives rewFactor/10, in a fixed way). It might cause the bot not to be motivated to follow impulses, but I hope it will be more adapted to the average, even if it's too much.

It seems that there is no change, I will increase the amount of reward for successful coherent movement.

6_09_1715

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	4096	epsilon	0.2
hidden units	256	learning rate	1.5e-4
learning rate schedule	constant	max steps	6.0e5
memory size	32	normalize	false
num epoch	6	num layers	3
sequence length	128	summary freq	1000
time horizon	64	use recurrent	false
extrinsic strength	1.0	extrinsic gamma	0.8
curiosity strength	0.02	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 1 * (64x64)
Grayscale: false
Observation space size = 1
Action space size = 1

Episode steps = 1000
Total steps = 600000
Time = 13000s

It now receives the same factor for the coherent as the incoherent (the coherent may be even greater), and the reward and penalty factors are the same (100).

In spite of the changes, it continues to do well on impulses (although it gets worse on the right) and the coherent movement has not changed much.

6_09_2142

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	4096	epsilon	0.2
hidden units	256	learning rate	1.5e-4
learning rate schedule	constant	max steps	6.0e5
memory size	32	normalize	false
num epoch	6	num layers	3
sequence length	128	summary freq	1000
time horizon	64	use recurrent	false
extrinsic strength	1.0	extrinsic gamma	0.8
curiosity strength	0.02	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 1 * (64x64)
Grayscale: false
Observation space size = 1
Action space size = 1

Episode steps = 1000
Total steps = 297000
Time = 6400s

Doubled the reward factor to 200 but the proportion that receives the consistent reward movement has been halved (0.5).

With these values it does not catch the right impulses well, the results are similar to the previous checkpoint.

Although the impulses are not perfectly modeled, in future tests we could start using bots with delays.

6_10_1148

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	4096	epsilon	0.2
hidden units	256	**learning rate**	**7.5e-4**
learning rate schedule	constant	max steps	6.0e5
memory size	32	normalize	false
num epoch	6	num layers	3
sequence length	128	summary freq	1000
time horizon	64	use recurrent	false
extrinsic strength	1.0	extrinsic gamma	0.8
curiosity strength	0.02	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 1 * (64x64)
Grayscale: false
Observation space size = 1
Action space size = 1

Episode steps = 1000
Total steps = 340000
Time = 7400s

I've added a consistency factor to the rewards within the average. The factors are 150 of penalty and 50 of reward. It also has a smoothing of deviations, and a higher learning rate.

The result was quite good and rising fast, but diverges at 320k steps (the learning rate schedule was constant, so it was to be expected). Although the right hand impulses were not always right.

6_10_1404

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	4096	epsilon	0.2
hidden units	256	learning rate	7.5e-4
learning rate schedule	**linear**	max steps	6.0e5
memory size	32	normalize	false
num epoch	6	num layers	3
sequence length	128	summary freq	1000
time horizon	64	use recurrent	false
extrinsic strength	1.0	extrinsic gamma	0.8
curiosity strength	0.02	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 1 * (64x64)

Grayscale: false

Observation space size = 1

Action space size = 1

Episode steps = 1000

Total steps = 600000

Time = 13000s

Noise at 0.3, same parameters.

Pretty good result, it fits well to the 2 movements, although it makes some impulse from the right loosely.

6_10_1854

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	4096	epsilon	0.2
hidden units	256	learning rate	7.5e-4
learning rate schedule	linear	max steps	6.0e5
memory size	32	normalize	false
num epoch	6	num layers	3
sequence length	128	summary freq	1000
time horizon	64	use recurrent	false
extrinsic strength	1.0	extrinsic gamma	0.8
curiosity strength	0.02	curiosity gamma	0.99
curiosity encoding size	128		

Frames: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Render Target Sensor: 10 * (32x32)

Grayscale: false

Observation space size = 1

Action space size = 1

Episode steps = 1000

Total steps = 600000

Time = 20000s

0.1 of react time and 0.2 of noise. All other parameters same as above. Now it receives as input the frames 1-2-3-4-5-6-7-8-9 (apart from the current one). The reaction should occur in frame 5-6. It may take longer to train, as it now receives 10 visual inputs instead of 1, and that can generate a lot of noise (it may also be better to give it the last 10 actions as input).

Using 64x64 inputs gives "Cuda out of memory" error, so I have decreased the size to 32 to redo the training.

The bot adapts well to most movements, both coherent and incoherent. However, some rightward impulses are not caught, and some consecutive movements are not as good either.

Training time has increased by 50 percent. Taking into account the amount of processed pixels: 64*64 = 4096, 32*32 * 10 = 10240. That's slightly more than double.

6_11_1247

Trainer: PPO

batch size	1024	beta	5.0e-3
buffer size	4096	epsilon	0.2
hidden units	256	learning rate	7.5e-4
learning rate schedule	linear	max steps	6.0e5
memory size	32	normalize	false
num epoch	6	num layers	3
sequence length	128	summary freq	1000
time horizon	64	use recurrent	false
extrinsic strength	1.0	extrinsic gamma	0.8
curiosity strength	0.02	curiosity gamma	0.99
curiosity encoding size	128		

Frames: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Render Target Sensor: 10 * (32x32)
Grayscale: false
Observation space size = 1
Action space size = 1

Episode steps = 1000
Total steps = 173000
Time = 8000s

0.12 of react time and 0.05 of noise. The render queue has also been optimized. There have been some out of memory failures, so I should see how to increase it.

Failure after several steps (sync error), probably because of the same thing of the memory.

6_11_1540

Trainer: PPO

batch size	**512**	beta	5.0e-3
buffer size	**2048**	epsilon	0.2
hidden units	256	learning rate	7.5e-4
learning rate schedule	linear	max steps	6.0e5
memory size	32	normalize	false
num epoch	**4**	num layers	3
sequence length	128	summary freq	1000
time horizon	64	use recurrent	false
extrinsic strength	1.0	extrinsic gamma	0.8
curiosity strength	0.02	curiosity gamma	0.99
curiosity encoding size	128		

Frames: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Render Target Sensor: 10 * (32x32)
Grayscale: false
Observation space size = 1
Action space size = 1

Episode steps = 1000
Total steps = 291000
Time = 10000s

The batch size and buffer size have been decreased to check if they consume less memory. Training may be less stable (although the num epoch has been lowered, which should make it more stable at the cost of being slower).

For some reason Unity fails and closes. It happens after optimizing the render targets.

6_12_1040

Trainer: PPO

batch size	512	beta	5.0e-3
buffer size	2048	epsilon	0.2
hidden units	256	learning rate	7.5e-4
learning rate schedule	linear	max steps	6.0e5
memory size	32	normalize	false
num epoch	4	num layers	3
sequence length	128	summary freq	1000
time horizon	64	use recurrent	false
extrinsic strength	1.0	extrinsic gamma	0.8
curiosity strength	0.02	curiosity gamma	0.99
curiosity encoding size	128		

Frames: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Render Target Sensor: 10 * (32x32)
Grayscale: false
Observation space size = 1
Action space size = 1

Episode steps = 1000
Total steps = 600000
Time = 20200s

The error seems to have its origin in the fact that the render targets created are not deleted/destroyed (the garbage collector doesn't collect them). Now I've put the destroy sequence, but if it still fails we'll have to call GC.Collect or Resources.UnloadUnusedAssets

too. The training is the same as the previous one. It reaches the same score but more slowly.

In this model, the standard deviation is a little bit higher, and the bot tends to adapt to it, so it has more noise than the original one.

6_12_1812

Trainer: PPO

batch size	512	beta	5.0e-3
buffer size	2048	epsilon	0.2
hidden units	256	learning rate	7.5e-4
learning rate schedule	linear	max steps	6.0e5
memory size	32	normalize	false
num epoch	4	num layers	3
sequence length	128	summary freq	1000
time horizon	64	use recurrent	false
extrinsic strength	1.0	extrinsic gamma	0.8
curiosity strength	0.02	curiosity gamma	0.99
curiosity encoding size	128		

Frames: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Render Target Sensor: 10 * (32x32)
Grayscale: false
Observation space size = 1
Action space size = 1

Episode steps = 1000
Total steps = 600000
Time = 19700s

React time of 0.08 and noise of 0.15. In this bot the spawner has been changed to balance the number of times each case is encountered, for the moment the same amount each of the 4 (chosen at random). Theoretically it should train faster and get a better result.

The spawns on the right keep failing.

6_13_1047

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	**hidden units**	**256**
init entcoef	1.0	learning rate	5.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	3	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	simple		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Frames: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Render Target Sensor: 10 * (32x32)

Grayscale: false

Observation space size = 1

Action space size = 1

Episode steps = 1000

Total steps = 74000

Time = 5000s

Training with SAC. The proportion of cases has been changed (more or less 50% left, and 24% consecutive).

It doesn't work, it stays in the average. It may occur because of receiving rewards in the center.

6_13_1224

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	5.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	3	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	simple		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Frames: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Render Target Sensor: 10 * (32x32)

Grayscale: false

Observation space size = 1

Action space size = 1

Episode steps = 1000

Total steps = 52000

Time = 3500s

I have removed the consistency factor on rewards within the average (this will apply to subsequent PPO trainings). In this one it does not receive any reward for getting the coherent movement right (prop. coherent = 0).

Now it doesn't seem to converge. It is kept as it is, the agent serves as an example when the problem is too complex for the number of layers used.

6_13_1327

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	5.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	**4**	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	simple		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Frames: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Render Target Sensor: 10 * (32x32)
Grayscale: false
Observation space size = 1
Action space size = 1

Episode steps = 1000
Total steps = 78000
Time = 5500s

Same parameters but one more layer in the network. If it doesn't improve the result, the problem may be in the visual encoder.

It follows the same trends but takes longer to get there. I don't know if it could end up converging but it doesn't seem to.

6_13_1506

Trainer: BC

batches per epoch	10	batch size	64
hidden units	**256**	learning rate	1.0e-5
max steps	5.0e4	memory size	32
num layers	**3**	sequence length	128
summary freq	1000	**use recurrent**	**false**

Demo path: demos/R008N015.demo
Frames: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Render Target Sensor: 10 * (32x32)

Grayscale: false
Observation space size = 1
Action space size = 1

Episode steps = 1000
Total steps = 11000
Time = 2500s

Because of an initial error, I believe that recurrent memory does not go well when it receives an observation. At the end it diverges, I'm going to try 4 layers.

6_13_1555

Trainer: BC

batches per epoch	10	batch size	64
hidden units	256	learning rate	1.0e-5
max steps	5.0e4	memory size	32
num layers	**4**	sequence length	128
summary freq	1000	use recurrent	false

Demo path: demos/R008N015.demo
Frames: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Render Target Sensor: 10 * (32x32)
Grayscale: false
Observation space size = 1
Action space size = 1

Episode steps = 1000
Total steps = 17000
Time = 4200s

It has a very erratic behavior, it doesn't get any impulse right. Special cases make the problem more complex to solve (and I think we will have to change the visual encoder).

6_13_1720

Trainer: BC

batches per epoch	10	batch size	64
hidden units	256	learning rate	1.0e-5
max steps	5.0e4	memory size	32
num layers	4	sequence length	128
summary freq	1000	use recurrent	false

Demo path: demos/R01N015T005.demo
Frames: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Render Target Sensor: 10 * (32x32)
Grayscale: false
Observation space size = 1
Action space size = 1

Episode steps = 1000
Total steps = 7000
Time = 1500s

A time deviation variable has been added (to 0.05). The noise is now 0.15 and the reaction time is 0.1. Only the common case is taken into account in this training, for simplicity.

It seems to work decently, it removes the noise and hits the impulses. Note** it had been trained with the bot's observations instead of its own, strange cases occur when it reacts earlier (but receives observations otherwise).

6_13_1800

Trainer: BC

batches per epoch	10	batch size	64
hidden units	256	learning rate	1.0e-5
max steps	5.0e4	memory size	32
num layers	4	sequence length	128
summary freq	1000	use recurrent	false

Demo path: demos/R001N015T014.demo
Frames: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Render Target Sensor: 10 * (32x32)
Grayscale: false
Observation space size = 1
Action space size = 1

Episode steps = 1000
Total steps = 7000
Time = 2000s

Extreme case: it can react in any of the frames it receives (from 0.01 to 0.15). Reaction time at 0.01 (temporary noise 0.14) and noise at 0.15.

When left longer it loses precision.

6_13_1843

Trainer: BC

batches per epoch	10	batch size	64
hidden units	256	learning rate	1.0e-5
max steps	5.0e4	memory size	32
num layers	4	sequence length	128
summary freq	1000	use recurrent	false

Demo path: demos/R001N015T014.demo
Frames: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Render Target Sensor: 10 * (32x32)
Grayscale: false
Observation space size = 1
Action space size = 1

Episode steps = 1000
Total steps = 2000
Time = 500s

Training with the same parameters as above but shorter. It has a much better result than the previous ones, from which conclusions could be drawn about how well it models behaviour. The next steps would be to apply it to bots with a higher range (and where the agent does not receive all the frames), probably with a higher number of observations of previous movements, to test other encoders (along with more cases), to check the performance of BC networks against PPOs (SAC did not work out well), etc.

Chapter 3

Discrete actions

6_17_1757

Trainer: PPO

batch size	512	beta	5.0e-3
buffer size	2048	epsilon	0.2
hidden units	256	learning rate	7.5e-4
learning rate schedule	linear	max steps	6.0e5
memory size	32	normalize	false
num epoch	4	num layers	3
sequence length	128	summary freq	1000
time horizon	64	use recurrent	false
extrinsic strength	1.0	**extrinsic gamma**	**0.99**
curiosity strength	0.02	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 20
Action space size = 1

Episode steps = 1000
Total steps = 294000
Time = 6800s

Click training (test) with cheat sheet: TR=0.1, pun and rew=10 For some reason, the bot doesn't click on compiled trainings, but it does in the editor.

Some incorrect impulses disappear over time, but the rewards seem very unbalanced.

6_17_1954

Trainer: PPO

batch size	512	beta	5.0e-3
buffer size	2048	epsilon	0.2
hidden units	256	learning rate	7.5e-4
learning rate schedule	linear	max steps	6.0e5
memory size	32	normalize	false
num epoch	4	num layers	3
sequence length	128	summary freq	1000
time horizon	64	use recurrent	false
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.02	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 64x64

Grayscale: false

Observation space size = 20

Action space size = 1

Episode steps = 1000

Total steps = 600000

Time = 13800s

15 of punish factor, 100 of reward factor and 0.2 of tolerable range. It improves quite a bit but has no penalty for leaving clicks, so it doesn't hit all of them (and gets a high reward).

6_18_1100

Trainer: PPO

batch size	512	beta	5.0e-3
buffer size	2048	epsilon	0.2
hidden units	256	learning rate	7.5e-4
learning rate schedule	linear	max steps	6.0e5
memory size	32	normalize	false
num epoch	4	num layers	3
sequence length	128	summary freq	1000
time horizon	64	use recurrent	false
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.02	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 64x64

Grayscale: false

Observation space size = 20

Action space size = 1

Episode steps = 1000

Total steps = 570000

Time = 12200s

Training with penalties for failing impulses (-4 * punFactor) added. I don't like very much that it relies on the interpolation variable to see the accuracy because in very short movements it's very difficult to get it right. In next trainings try with SAC or BC, and then think about improving the reward system.

I think the penalty was too high, now it makes too many impulses. Next time I try SAC and less penalty.

6_18_1426

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	5.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	**2**	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	simple		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 20
Action space size = 1

Episode steps = 1000
Total steps = 488000
Time = 17300s

Training with SAC, 2*punish for leaving impulses and the same parameters. I have also reduced the amount of layers (although maybe one would be enough). It hits quite a few impulses, except for very specific (and maybe confusing) cases. It reaches the rewards of 6_17_1954, but with even stricter conditions.

6_19_1045

Trainer: BC

batches per epoch	10	batch size	64
hidden units	128	learning rate	1.0e-5
max steps	5.0e4	memory size	32
num layers	4	sequence length	128
summary freq	1000	use recurrent	false

Demo path: demos/Clicker09.demo

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 20

Action space size = 1

Episode steps = 1000
Total steps = 50000
Time = 5200s

I had several ValueErrors because the heuristics created a size 2 action space (not compatible with the real one).

The model, although it gets some impulses right, is very imprecise. It doesn't seem to improve the SAC or PPO trainings at all.

6_19_1311

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	5.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	simple		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 20
Action space size = 1

Episode steps = 1000
Total steps = 348000
Time = 12800s

Tolerable range reduced to 0.1. The reward should be lower than the previous model with SAC, but it will be necessary to check if it is better in practice.

It seems to work well. Comparison with previous model with range 0.2 is missing.

6_19_1959

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	5.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	simple		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 64x64

Grayscale: false

Observation space size = 20

Action space size = 1

Episode steps = 1000

Total steps = 452000

Time = 16000s

Noise factor of 2 added (calculated differently from other noises). However, I can't get the movement to be stable in Y (it goes in both directions).

At 390k steps it diverges. Then it seems that the graph stays flat.

6_22_1213

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	5.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	simple		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 64x64

Grayscale: false

Observation space size = 20

Action space size = 1

Episode steps = 1000

Total steps = -

Time = -

New bot with multiple inaccuracies (BotPrecision): inaccuracy=1.2, noise in X = 0.8, noise in Y = 0.1, timeClic (average) = 0.9, time click variance = 0.1. Targets are now also detected while in idle. The penalties are exaggeratedly large (around 10,000), so I should balance them out in subsequent trainings.

6_22_1243

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	5.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	simple		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 64x64

Grayscale: false

Observation space size = 20

Action space size = 1

Episode steps = 1000

Total steps = 126000

Time = 4500s

The error was that the agent was getting punishments from the bot that didn't move instead of the new one (it has to be changed in the function of the rewards, since it's a getComponent).

The result is decent as almost any of the previous ones (even if this one has more noise), but sometimes the bot clicks without coming to mind.

6_22_1402

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	5.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	**resnet**		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 20
Action space size = 1

Episode steps = 1000
Total steps = 118000
Time = 6200s

Use of resnet as a visual encoder. It is expected to be slower, but it should also work better.

I think it always acts in front of the target as it destroys it, but it has the potential to act (it does it in the middle). It could end the training being the network that makes the clicks instead of the bot (if I can get the bot not to go crazy by not hitting the target).

I've activated the neural network for a few steps, but it has the problem that if it's the network that determines when to click, the agent's shots are not shown (I don't know if it penalizes or not). It works like the previous ones, in the next trainings I will change to half agent control (working well).

The performance of the game when using the net is as good as with other visual encoders (the training is a bit slower), but the result seems to be a bit better.

It causes failures when it doesn't click when playing the neural network, the bot stands pointing at a target but doesn't move. That problem is tried to be solved in the next training.

6_22_1603

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	5.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 64x64

Grayscale: false

Observation space size = 20

Action space size = 1

Episode steps = 1000

Total steps = 65000

Time = 3600s

Fixed some bugs where the clicks were not displayed as the runtime was programmed. Now the agent can be trained as the main player (although it's not recommended to do it from the beginning).

Switch to neural network control at 40,000 steps.

Be careful to avoid getting stuck on the site if it doesn't respond: penalize strongly and continue after X failures.

The problem of switching is to get stuck if it doesn't respond, maybe if it is trained from the beginning with agent control it will have better results.

6_22_1725

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	5.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 64x64

Grayscale: false

Observation space size = 20

Action space size = 1

Episode steps = 1000

Total steps = 16000

Time = 700s

Train the bot from the beginning, the number of simultaneous planes has been limited again. Increased penalty for failing to click to 3 times punFactor (now it has worse consequences to leave a click than to fail it).

After several changes it stays at *5 for not clicking and *2 for clicking in idle or too late.

Try PPO in next trainings.

Maybe I could put as an input parameter the last value of the network (to know if it has clicked or not).

6_22_1740

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	5.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 64x64

Grayscale: false

Observation space size = 21

Action space size = 1

Episode steps = 1000

Total steps = 71000

Time = 3700s

Added the last click value as a parameter (if it's higher than 0, it should force the bot to go negative so the next frame can be clicked). I think that in previous trainings it got stuck because when it received only the last image, it considered that it should stay in positive values.

Clicking too much should give a small penalty (sometimes it makes 3 clicks in a row, but it gets 0 points). The result is still much better.

6_22_1847

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	5.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 64x64

Grayscale: false

Observation space size = 21

Action space size = 1

Episode steps = 1000

Total steps = 245000

Time = 6500s

PunFactor/3 punishment has been added for over-clicking (it may not have much effect, since the bot has no click memory, or it may force it to perform better).

It has had a crash for unknown reasons (it doesn't show up in the log editor either). However, the reward already seemed to be stagnant for a long time. The model works well.

6_23_1013

Trainer: PPO

batch size	512	beta	5.0e-3
buffer size	2048	epsilon	0.2
hidden units	256	learning rate	7.5e-4
learning rate schedule	linear	max steps	6.0e5
memory size	32	normalize	false
num epoch	4	num layers	3
sequence length	128	summary freq	1000
time horizon	64	use recurrent	false
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.02	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 21
Action space size = 1

Episode steps = 1000
Total steps = 33000
Time = 770s

Training with DISCRETE space of action (1 branch of size 2). The rest of the training works the same way. In this case it has a simple encoder, I stop the training to compare it with resnet.

6_23_1028

Trainer: PPO

batch size	512	beta	5.0e-3
buffer size	2048	epsilon	0.2
hidden units	256	learning rate	7.5e-4
learning rate schedule	linear	max steps	6.0e5
memory size	32	normalize	false
num epoch	4	num layers	3
sequence length	128	summary freq	1000
time horizon	64	use recurrent	false
vis encode type	**resnet**		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.02	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 21
Action space size = 1

Episode steps = 1000
Total steps = 109000
Time = 2600s

Change on the encoder to compare performance. Use SAC in discrete space for the next workout.

The reward curve increases but after a number of steps it stays flat (in the graph the agent is stuck at 1, but without clicking).

6_23_1118

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	5.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 21
Action space size = 1

Episode steps = 1000
Total steps = 54000
Time = 2750s

Training with discrete action space and SAC. Before 10000 steps the reward already falls and the agent stagnates and does not improve again.

6_23_1206

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	5.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 64x64

Grayscale: false

Observation space size = 21

Action space size = 1

Episode steps = 1000

Total steps = 38000

Time = 1800s

Same training but lastClick is saved as a random number (instead of 0 or 1). Same result.

6_23_1305

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	5.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 64x64

Grayscale: false

Observation space size = 21

Action space size = 1

Episode steps = 1000

Total steps = 13000

Time = 600s

Training with continuous values again. A factor has been added whereby movements with short reaction times discount less and give more reward. I don't know how the model can get better or worse with these values. It has learned but somewhat worse, it's not valid because of a failure in the calculation of the new factors.

6_23_1317

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	5.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 64x64

Grayscale: false

Observation space size = 21

Action space size = 1

Episode steps = 1000

Total steps = 91000

Time = 4600s

Error corrected. Correct results are obtained, but they are not very different in quality from those that did not take time into account. However, there are times when it acts differently (it stays up while it doesn't have to click, and in a while it goes back down).

6_23_1855

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	5.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 64x64

Grayscale: false

Observation space size = 21

Action space size = 1

Episode steps = 1000

Total steps = 7000

Time = 250s

Training with discrete action space but considering that it clicks whenever iClick is 1 (in each frame independently, without depending on whether the previous one was 0).

The initial rewards are very low due to the large amount of clicks it makes, I hope it won't diverge because of that.

In a matter of 5000 steps it has already learned not to make clicks, since not acting is more profitable than constantly failing.

6_23_1905

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	5.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 21
Action space size = 1

Episode steps = 1000
Total steps = 66000
Time = 2800s

PunFactor of 2, RewFactor of 30, and ignoring a click is penalized *20. It works well but not better than its continuous motion counterparts, it has more uncounted clicks than the other models (although it usually gets it right when it should).

6_24_1306

Trainer: BC

batches per epoch	10	batch size	64
hidden units	128	learning rate	1.0e-5
max steps	5.0e4	memory size	32
num layers	4	sequence length	128
summary freq	1000	use recurrent	false

Demo path: demos/ClickerDiscrete.demo

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 21
Action space size = 1

Episode steps = 1000
Total steps = 22000
Time = 2680s

Behavioral cloning with discrete movements. It has the problem of learning to act in the next frame (it should be commented in a section) because the demos are 1 frame late, when the target has already been destroyed.

Chapter 4

Complex movements

6_25_1134

Trainer: PPO

batch size	512	beta	5.0e-3
buffer size	2048	epsilon	0.2
hidden units	256	learning rate	7.5e-4
learning rate schedule	linear	max steps	6.0e5
memory size	32	normalize	false
num epoch	4	num layers	3
sequence length	128	summary freq	1000
time horizon	64	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.02	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 20
Action space size = 2

Episode steps = 1000
Total steps = 82000
Time = 2700s

Noiseless and accurate bot (and movement curve with increasing speed), rewards with tolerable range of 0.15 and factors of 30 (reward and penalty).

At the beginning it shows a cuda out of memory error that in the last ones didn't show up, I don't know if it could be because the bot is bigger, because of the encoder or because of using the 2 debugs at the same time.

At the end it has failed (GPU sync error). The learning has followed very strange patterns, it always gave values ending in 11 and it has been jumping between 3 concrete rewards: 7211, 7811, 8411 (and once 9011).

6_25_1246

Trainer: PPO

batch size	512	beta	5.0e-3
buffer size	2048	epsilon	0.2
hidden units	256	learning rate	7.5e-4
learning rate schedule	linear	max steps	6.0e5
memory size	32	normalize	false
num epoch	4	num layers	3
sequence length	128	summary freq	1000
time horizon	64	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.02	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 20
Action space size = 2

Episode steps = 1000
Total steps = 20000
Time = 700s

The distance function doesn't seem to be going wrong, now punFactor=10, rewFactor=30 and maxRange=2 (of tolerable range). Out of memory errors still appear.

For some reason it follows the same patterns as before, even with the same times. It fails again as before.

6_25_1302

Trainer: PPO

batch size	512	beta	5.0e-3
buffer size	2048	epsilon	0.2
hidden units	256	learning rate	7.5e-4
learning rate schedule	linear	max steps	6.0e5
memory size	32	normalize	false
num epoch	4	num layers	3
sequence length	128	summary freq	1000
time horizon	64	use recurrent	false
vis encode type	**simple**		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.02	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 64x64
Grayscale: false
Observation space size = 20
Action space size = 2

Episode steps = 1000
Total steps = 29000
Time = 650s

Change in the encoder. No out of memory errors at first. In the next one I'm going to put resnet again, but with a network with less internal layers.

6_25_1322

Trainer: PPO

batch size	512	beta	5.0e-3
buffer size	2048	epsilon	0.2
hidden units	256	learning rate	7.5e-4
learning rate schedule	linear	max steps	6.0e5
memory size	32	normalize	false
num epoch	4	**num layers**	**2**
sequence length	128	summary freq	1000
time horizon	**32**	use recurrent	false
vis encode type	**resnet**		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.02	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 32x32
Grayscale: false
Observation space size = 20
Action space size = 2

Episode steps = 1000
Total steps = 49000
Time = 1000s

The error was that the cheat rewards missed were added (this always happened).

Training with the same parameters but simpler (and with less time horizon, which could decrease even more in theory).

Image size has been decreased to see if the problem is there (resnet should need then less memory).

It seems recommended to decrease the batch (and buffer) size to improve the performance, I will try it in next trainings.

For some reason, the debug axis lines are "destroyed" (the object disappears) and that causes an error in Unity, we'll have to see where the bug is.

On the other hand, the rewards are very small (even being multiplied by 30, but at the end you divide everything by 1000).

In the end it started to rise, but the axis debug was destroyed. The mistake was that the debug's sphere had a collider, and when the bot shot at it, the parent object (the whole axis debug) was destroyed. This shouldn't be a problem now.

6_25_1403

Trainer: PPO

batch size	512	beta	5.0e-3
buffer size	2048	epsilon	0.2
hidden units	256	learning rate	7.5e-4
learning rate schedule	linear	max steps	6.0e5
memory size	32	normalize	false
num epoch	4	num layers	2
sequence length	128	summary freq	1000
time horizon	32	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.02	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 32x32
Grayscale: false

Observation space size = 20
Action space size = 2

Episode steps = 1000
Total steps = 291000
Time = 5900s

Fixed bugs and Rew-Pun factors multiplied by 10 (100-300).
It goes up slowly but steadily, it may be more optimal to force the bot to have less reward (with the same factors, as before). In the graph it seems to be tending to the same side but with a lot of noise, in this case it would also be convenient to reduce the tolerable range.

6_25_1546

Trainer: PPO

batch size	512	beta	5.0e-3
buffer size	2048	epsilon	0.2
hidden units	256	learning rate	7.5e-4
learning rate schedule	linear	max steps	6.0e5
memory size	32	normalize	false
num epoch	4	num layers	2
sequence length	128	summary freq	1000
time horizon	32	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.02	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 32x32
Grayscale: false
Observation space size = 20
Action space size = 2

Episode steps = 1000
Total steps = 480000
Time = 11200s

RewFactor=PunFactor=300, TR=0.1. It does improve the reward but not so much the behavior. Check how it goes with 2 debug axes.

6_25_1859

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	5.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 32x32

Grayscale: false

Observation space size = 20

Action space size = 2

Episode steps = 1000

Total steps = 248000

Time = 8900s

It ends up converging to a kind of medium value, but it achieves positive rewards. The result is not very good also because the observations were of the agent (at first almost random, and then very uniform). In the next one reduce the tolerable range and pass pure observations of the bot.

6_25_2132

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	5.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 32x32

Grayscale: false

Observation space size = 20

Action space size = 2

Episode steps = 1000

Total steps = 270000

Time = 9700s

0.05 of TR and observations of the bot. It has managed to adapt well in a moment, but over 250,000 steps it has started to become unstable. The noise in Y is much higher than the noise in X.

It is considered good because it has been close to a good result (although it does not work well in the editor, since it used bot observations).

6_26_1159

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	5.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 32x32

Grayscale: false

Observation space size = 20

Action space size = 2

Episode steps = 1000

Total steps = 88000

Time = 3100s

Training with angle-magnitude tolerable range rewards TRAngle = 0.05, TRMagnitude = 0.05, MR = 0.5, punFactor=200, rewFactor=300.

I think there's some failure in the rewards because it's moving almost the opposite way and getting less negative rewards each time, I'll let it go to see how it goes.

6_26_1329

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	5.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 32x32

Grayscale: false

Observation space size = 20

Action space size = 2

Episode steps = 1000

Total steps = 260000

Time = 9000s

The problem was in how the arc tangents were applied, so that the opposite one gave the same result as the positive one (-X/-Y==X/Y). Now I use Vector2.Angle to calculate it directly (in decimal).

The final result isn't bad at all but it is quite noisy and slow (and still not out of the bot's observations). I should add sensitivity to the movement in the Y axis (and maybe in the X axis) because the bot has more trouble adjusting to very small numbers.

6_26_1612

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	5.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 32x32

Grayscale: false

Observation space size = 20

Action space size = 2

Episode steps = 1000

Total steps = 187000

Time = 6400s

Same training as before but with sensitivity of 10 on the Y-axis (has more range of accuracy). Already from the beginning it can be seen that the movements in Y are flatter than in X, I hope that it can adapt to larger ranges of movement in spite of the sensitivity. In next trainings we have to improve the reward system to penalize the distance in magnitude.

The reward curve is quite improved, even though we haven't included magnitude penalties yet.

In the editor, it needs the agent's observations to work well, although it does not become chaotic if left alone (sometimes it can end up looking at the zenith or the ground, but if the agent's observations are put back in, it recovers its position). It is also true that these cases have not yet been trained.

6_26_2000

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	5.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 32x32
Grayscale: false
Observation space size = 20
Action space size = 2

Episode steps = 1000
Total steps = 220000
Time = 8100s

A penalty has been added for failing the magnitude even if the angle is correct (it is 3 times less than punFactor, although it may have less influence on how the penalty is calculated).

The result is better than the previous ones, but taking into account that we still start from only bot observations. However, it doesn't seem to carry the Y-impulses very well (maybe because it doesn't see many, or because it has too much sensitivity).

Despite adding a penalty, it has achieved much higher rewards.

6_27_1116

Trainer: BC

batches per epoch	10	batch size	64
hidden units	128	learning rate	1.0e-5
max steps	5.0e4	memory size	32
num layers	4	sequence length	128
summary freq	1000	use recurrent	false

Demo path: demos/ComplexDemo.demo

Render Target Sensor: 32x32
Grayscale: false
Observation space size = 20
Action space size = 2

Episode steps = 1000
Total steps = 37000
Time = 3400s

BC training and bot observations. It quickly adapts to the lines.

I should lower the sensitivity parameter in Y (10), and make new demos in which the position of the agent changes randomly (also modify Spawn Cases, so that it does not create planes while looking up/down). It doesn't take very well the impulses to the left at the end.

6_27_1240

Trainer: BC

batches per epoch	10	batch size	64
hidden units	128	learning rate	1.0e-5
max steps	5.0e4	memory size	32
num layers	4	sequence length	128
summary freq	1000	use recurrent	false

Demo path: demos/ComplexDemo.demo

Render Target Sensor: 32x32
Grayscale: false
Observation space size = 20
Action space size = 2

Episode steps = 1000
Total steps = 9000
Time = 900s

Changing the observations from bot to agent at 6000 steps has stopped working well. Transitioning from one side to the other, it can be seen that the curve starts to flatten out.

6_27_1257

Trainer: BC

batches per epoch	10	batch size	64
hidden units	128	learning rate	1.0e-5
max steps	5.0e4	memory size	32
num layers	4	sequence length	128
summary freq	1000	use recurrent	false

Demo path: demos/ComplexDemo.demo

Render Target Sensor: 32x32
Grayscale: false
Observation space size = 20
Action space size = 2

Episode steps = 1000
Total steps = 13000
Time = 1200s

Sensitivity in Y to 5 (also changed in the demo). Now start from the beginning with observations from the same agent.

It has been adapting to a kind of midpoint (although it was doing some impulse) but at 9000 steps it already loses control.

6_27_1331

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	5.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		
gail strength	**0.1**	**gail gamma**	**0.99**
gail encoding size	**128**		

Demo path: ./demos/ComplexDemo.demo

Render Target Sensor: 32x32
Grayscale: false
Observation space size = 20
Action space size = 2

Episode steps = 1000
Total steps = 470000
Time = 18800s

Gail with agent's remarks. It takes it quite a few steps to get a good result, in the end it adapts to some of the curves but not with much precision. However, it usually gets the angle of the movements right (not so much the magnitude).

6_27_1846

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	5.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		
gail strength	**0.05**	gail gamma	0.99
gail encoding size	128		

Demo path: ./demos/ComplexDemo.demo

Render Target Sensor: 32x32
Grayscale: false
Observation space size = 20
Action space size = 2

Episode steps = 1000
Total steps = 135000
Time = 5400s

Training with less GAIL, in this case with bot observations. It does not improve previous results, even with these observations.

6_27_2054

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	5.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 32x32
Grayscale: false
Observation space size = 20
Action space size = 2

Episode steps = 500
Total steps = 389000
Time = 13600s

In this training I will add the agent restart: at the end of each episode (which will be shorter than 1000).

I think I'll also have to restart the bot's movement with the agent reset to make it work.

Maybe the impulses created by restarting the agent are too big for the Y-sensitivity it has right now.

There are parts of the movement line to which it adapts very well, however it tends to "anticipate" the next impulse too soon. If I could get it to not finish them so soon it would be a very good result (even taking into account that we still have bot observations).

6_29_1528

Trainer: PPO

batch size	512	beta	5.0e-3
buffer size	2048	epsilon	0.2
hidden units	256	learning rate	7.5e-4
learning rate schedule	linear	max steps	6.0e5
memory size	32	normalize	false
num epoch	4	num layers	2
sequence length	128	summary freq	1000
time horizon	32	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.02	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 32x32

Grayscale: false

Observation space size = 20

Action space size = 2

Episode steps = 500

Total steps = 337000

Time = 8000s

Training with cheat shaped tolerable range, the curves have experimental forms yet. RewFactor = PunFactor = 400. In this case it only creates targets on the right side (half in and half out). I haven't changed the Y-sensitivity (still at 5) and that might affect the training somewhat.

It may be necessary to add a way that if it misses the shape at the end (anticipates too much the next move) it won't be rewarded later. Also remove the random reset.

It doesn't mislead the angle of the move, but the reward stagnates at 100 thousand steps.

6_29_1748

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	5.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 32x32

Grayscale: false

Observation space size = 20

Action space size = 2

Episode steps = 1000

Total steps = 100000

Time = 3600s

Random reboot and Y-sensitivity have been removed (although this may not have been necessary), and training is also done with SAC. The curves and factors are the same as before.

It tends to stay static in a straight line (as in some previous model), it hits the angle quite well but not the magnitude. I think it needs more penalty factor to be somewhat more effective.

6_29_1853

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	5.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 32x32

Grayscale: false

Observation space size = 20

Action space size = 2

Episode steps = 1000

Total steps = 124000

Time = 4500s

600 of punish factor and 250 of reward factor, the other parameters remain the same. It's not much better than before, probably because it's easier to get the initial impulse right than to try to gain something from the others. I'm going to try to make sure that if you don't get the last move right, you don't get rewarded on the next one.

6_29_2018

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	5.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 32x32

Grayscale: false

Observation space size = 20

Action space size = 2

Episode steps = 1000

Total steps = 498000

Time = 17600s

Corrections in the curves, it should work better now, although I haven't done the thing where I don't get rewarded on the first move if I miss the last one on the previous one.

Learn the angle correctly but not so much the magnitude, it might be necessary to add more exigency there (or a different curve). The amount of observations it receives could be reduced, since 10 previous movements are not necessary to have a clear idea of the movement (3-5 would be enough).

6_30_0842

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	5.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 32x32

Grayscale: false

Observation space size = 6

Action space size = 2

Episode steps = 1000

Total steps = 145000

Time = 5000s

Observations reduced to the last 3 of each axis. The way to detect targets has also been changed to isVisible.

Momentum variable added, increases the reward if several are combined in a row and decreases it if not. PunFactor = 300 and RewFactor = 125, same curves.

It still matches the angle but not the magnitude.

6_30_1013

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	5.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: 32x32
Grayscale: false
Observation space size = 6
Action space size = 2

Episode steps = 1000
Total steps = 71000
Time = 2400s

I removed a Mathf.Min that limited the penalties by magnitude. It should keep the spawner within the limits of the camera angle, because I'm seeing a lot of repeat shots.

6_30_1043

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	5.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: -
Grayscale: -
Observation space size = 6
Action space size = 2

Episode steps = 1000
Total steps = 87000
Time = 1700s

Training without spawns and without a camera, the only objective is to imitate movement. All other parameters are the same as before. We will also have to review how the penalty is calculated with magnitude and think about increasing it. It follows the same curve, the next time I try to remove the factor that decreases the magnitude penalties.

6_30_1132

Trainer: PPO

batch size	512	beta	5.0e-3
buffer size	2048	epsilon	0.2
hidden units	256	learning rate	7.5e-4
learning rate schedule	linear	max steps	6.0e5
memory size	32	normalize	false
num epoch	4	num layers	2
sequence length	128	summary freq	1000
time horizon	32	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.02	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: -
Grayscale: -
Observation space size = 6
Action space size = 2

Episode steps = 1000
Total steps = 214000
Time = 4300s

Now there's no discount for failing the magnitude factor. I've also removed the momentum. It doesn't seem to be getting any better.

6_30_1246

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	5.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: -

Grayscale: -

Observation space size = 6

Action space size = 2

Episode steps = 1000

Total steps = 123000

Time = 3000s

Same training as before but using SAC. If it doesn't improve we will have to rethink the shape of the curves or how to give more importance to the magnitude than to the angle. Same result.

6_30_1337

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	5.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: -
Grayscale: -
Observation space size = 6
Action space size = 2

Episode steps = 1000
Total steps = 110000
Time = 2700s

More pronounced penalty curve. If it does not change, the tolerable range curve will probably have to be modified.

6_30_1425

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	**learning rate**	**7.0e-4**
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: -
Grayscale: -
Observation space size = 6
Action space size = 2

Episode steps = 1000
Total steps = 191000
Time = 4500s

Higher learning rate, 400 punFactor and 100 rewFactor.

6_30_1542

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	7.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: -
Grayscale: -
Observation space size = 6
Action space size = 2

Episode steps = 1000
Total steps = 433000
Time = 11000s

Modified tolerable range curve, and some other modifications to the other 2. RewFactor=100, PunFactor=500.
Improved quite a bit from the previous ones.

6_30_1853

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	7.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: -
Grayscale: -
Observation space size = 6
Action space size = 2

Episode steps = 1000
Total steps = 260000
Time = 6500s

Tolerable range curve flatter, small changes in the other 2. Rew=90, Pun=500.

6_30_2044

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	7.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: -
Grayscale: -
Observation space size = 6
Action space size = 2

Episode steps = 1000
Total steps = 339000
Time = 8000s

Good result like the previous ones, has to anticipate the next movements yet.

6_30_2300

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	7.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: -
Grayscale: -
Observation space size = 6
Action space size = 2

Episode steps = 1000
Total steps = 223000
Time = 5600s

More reward at the end of the moves, the rest of the curves and parameters remain the same. It manages to make some pretty good moves, but it's far from perfect. In next practice I could try to reverse the reward curve to see how it acts, and use PPO with these rewards.

7_01_0903

Trainer: PPO

batch size	512	beta	5.0e-3
buffer size	2048	epsilon	0.2
hidden units	256	learning rate	7.5e-4
learning rate schedule	linear	max steps	6.0e5
memory size	32	normalize	false
num epoch	4	num layers	2
sequence length	128	summary freq	1000
time horizon	32	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.02	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: -

Grayscale: -

Observation space size = 6

Action space size = 2

Episode steps = 1000

Total steps = 309000

Time = 6300s

Same training as 6_30_2300, but using PPO instead of SAC. Compare both models at the end.

The result is much worse than with SAC. It is saved for comparison.

7_01_1053

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	7.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: -

Grayscale: -

Observation space size = 6

Action space size = 2

Episode steps = 1000

Total steps = 78000

Time = 1900s

Reward factor curve with higher value at the end (in addition to the beginning). The factors remain at: rew=90, pun=500.

It tends to stay flat at the top (although it does some concrete moves well).

7_01_1133

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	7.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: -

Grayscale: -

Observation space size = 6

Action space size = 2

Episode steps = 1000

Total steps = 63000

Time = 1500s

Flatter tolerable range curve, earlier shape rewards curve and slightly longer penalty curve. Still using momentum, now I've put a variable to control the relationship between the magnitude penalties (the higher it is, the more its importance is reduced, and if it's lower than 1 it's given more value than the angle, for now it's 1).

It seems to be worse than the cases with similar curves.

7_01_1201

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	7.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: -

Grayscale: -

Observation space size = 6

Action space size = 2

Episode steps = 1000

Total steps = 221000

Time = 5500s

Tolerable range curve scaled up (as before), and the penalty curve restored to the previous shape. The magnitude factor is now 0.75, so it penalizes more.

Pretty good result.

In the next practices I will use 3 more curves to save the best ones.

7_01_1341

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	7.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: -

Grayscale: -

Observation space size = 6

Action space size = 2

Episode steps = 1000

Total steps = 245000

Time = 6200s

Tolerable range curve with more value at the end, sharper reward curve (only the first move with more value counts practically) and less lengthy penalty curve. Pun=500 and Rew=75.

The result has improved, the next one I'll add more pressure to the penalties.

7_01_1529

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	7.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: -
Grayscale: -
Observation space size = 6
Action space size = 2

Episode steps = 1000
Total steps = 118000
Time = 3000s

Rew=70, Pun=600. Same curves as in previous practice.
Looks slightly worse than the last one.

7_01_1623

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	7.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: -
Grayscale: -
Observation space size = 6
Action space size = 2

Episode steps = 1000
Total steps = 195000
Time = 5000s

Tolerable reduced range, but also the penalty factor: PunFactor=400, RewFactor=100.

Correct result, similar to the previous ones (not saved).

7_01_1748

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	7.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: -
Grayscale: -
Observation space size = 6
Action space size = 2

Episode steps = 1000
Total steps = 95000
Time = 2400s

Training with the tolerable range demanding of the last one, but factors of 500-75. In addition, the magnitude ratio is 0.5 so it penalizes even more. It does not improve the previous results, the best model so far seems to be 7_01_1341.

7_01_1851

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	7.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: -

Grayscale: -

Observation space size = 6

Action space size = 2

Episode steps = 1000

Total steps = 73000

Time = 1800s

Same parameters as in 7_01_1341 (Pun=500, Rew=75, useMomentum, magnitude=0.75) but using as space actions the angle of the movement and its magnitude (normalized). I could create a vector of observations that had the last angles and magnitudes (instead of x and y).

The result seems to improve at times, in future trainings use angle and magnitude observations.

7_01_1931

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	7.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: -

Grayscale: -

Observation space size = 6

Action space size = 2

Episode steps = 1000

Total steps = 143000

Time = 3400s

Now the observations are **angle-magnitude** (which are also size 6). In theory the problem should be rather more trivial for the neural network, even if the numbers saved from observation are not the exact actions (they are saved already processed and normalized).

It makes some good angles, but only underneath. This may be because at angle 0 the action angles -1 and +1 coincide.

7_01_2044

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	7.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: -

Grayscale: -

Observation space size = 6

Action space size = 2

Episode steps = 1000

Total steps = 62000

Time = 1500s

Flat curves. The direction of movement in this bot has been reversed (now it goes to the left). The observations have been modified to save the action before processing (keep in mind when testing the agent 7_01_1931).

The movement remains flat, these curves are not useful.

7_01_2114

Trainer: SAC

batch size	128	buffer size	200000
buffer init steps	5000	hidden units	256
init entcoef	1.0	learning rate	7.0e-4
learning rate schedule	constant	max steps	6.0e6
memory size	32	normalize	true
num update	1	train interval	5
num layers	2	time horizon	64
sequence length	128	summary freq	1000
tau	0.005	use recurrent	false
vis encode type	resnet		
extrinsic strength	1.0	extrinsic gamma	0.99
curiosity strength	0.1	curiosity gamma	0.99
curiosity encoding size	128		

Render Target Sensor: -
Grayscale: -
Observation space size = 6
Action space size = 2

Episode steps = 1000
Total steps = 94000
Time = 2400s

Modified curves similar to the first ones, pun=600 and rew=75.
The curve stays flat on the average, it doesn't work.