1 Nomenclature

Symbol	Description
C^i	Cartesian coordinates at time step i, $C^i = (x^t, y^i, z^i)$, C^{goal} represents goal coordinates
R	Cumulative reward, $R = \sum r$
r^t	Reward at time step t, $r^0 = 0$
D^i	Distance to goal at time step i, $D^i = \lvert C^{goal} - C^i \rvert$
D^i_n	Normalised distance to goal at time step i, $D^i_n = \lvert C^{goal} - C^i \rvert \div \lvert C^{goal} - C^0 \rvert$
k_{exp}	Exponential reward scaling constant, user variable
r_{reach}	Bonus reward upon reaching target, user variable

2 User Variables

Symbol	Variable Name	Description
k_{exp}	exp_rew_scaling	Exponential reward scaling constant
r_{reach}	reach_target_bonus_reward	Bonus reward upon reaching target

3 Reward

Reward, r is given by the sum of 4 components, r_b, r_n, r_e and r_{reach} which are base reward, normalised reward, exponential reward, and reward upon reaching respectively.

$$r = r_b + r_n + r_e + r_{reach} \tag{1}$$

3.1 Base Reward, r_b

At time t, Base reward, r_b, is calculated using changes in cartesian distance

$$r_b^t = |C^t - C^{t-1}| \tag{2}$$

3.2 Normalised Reward, r_n

Reward is normalised with the aim of making the robot getting the same reward when reaching different goal coordinates. The normalisation approach is described as follows.

1. Determine the magnitude of the changes between initial and goal coordinates, $K = |C^{goal} - C^0|$
2. Calculate normalised reward by dividing base reward with the magnitude K

Therefore,

$$r_n^t = r_b^t \div K = \frac{|C^t - C^{t-1}|}{|C^{goal} - C^0|} \tag{3}$$

3.2.1 Plots

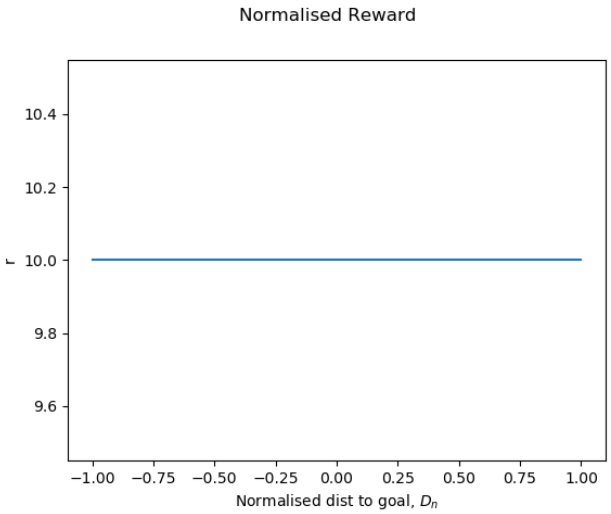

Figure 1: Plot of time step reward, r_n), scale=10

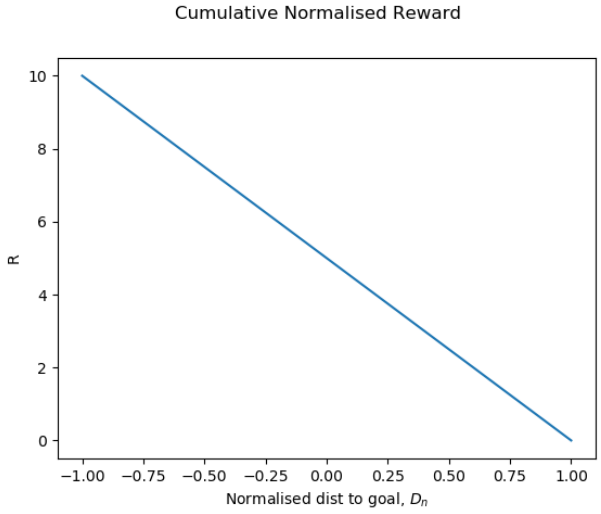

Figure 2: Plot of cumulative reward, R_n, scale=10

3.3 Exponential Reward, r_e

This is to provide additional incentive for the robot to complete the final bit of the task, by providing more reward towards the end. An example of such problem is when the robot is doing only 80% of the task but never 100%, even after prolonged periods of training.

3.3.1 Derivation

Since the x-variable is distance to goal, D_n, it can be a bit inconvenient because we want the maximum value of R_e to be when D_n is close to 0. As such, we transform the x-variable from D_n to some arbitrary variable y, such that $y = 1 - D_n$.

This will produce the results we want, as we get maximum reward when $D_n = 0$, and a reward of 0 when $D_n = 1$, which represents the starting position.

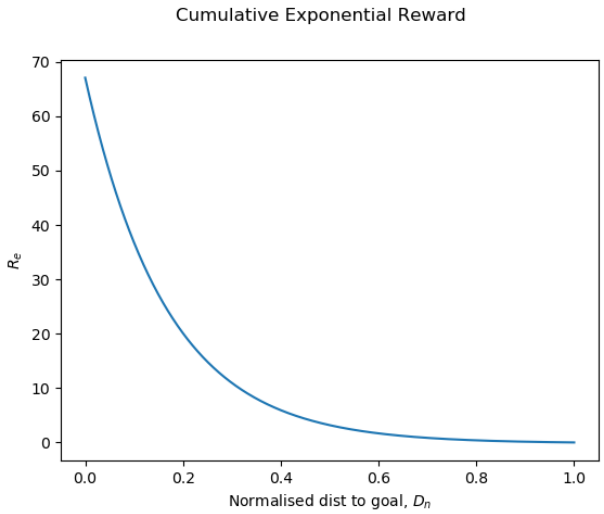

Figure 3: Plot of cumulative exponential reward, R_e, k_{exp}=6, $D_n = [0, 1.0]$

Based on Figure 3, we can see that the if $D_n > 1$, the rewards will only become insignificant and never going negative.

However, we want to penalise the robot for moving away from the goal, which meant that we need to give negative rewards for $D_n > 1$. As such, R_e is made into a piecewise function, with a linear negative function at $D_n > 1$. A further modification is made to the negative region function such that even if the robot keeps moving away from the goal, the reward will not get too negative. This is by introducing another piecewise linear function at $D_n > 9$ with a much lower scaling.

Therefore, our formula for R_e is given by:

$$R_e(D_n) = \begin{cases} \frac{1}{k_{exp}} e^{k_{exp}(1-D_n)} & 0 \leq D_n \leq 1 \\ 5(1-D_n) & 1 < D_n \leq 9 \\ 0.5(1-D_n) & 9 < D_n \end{cases} \tag{4}$$

And the formula of r_e is given by:

$$r_e^t = R_e(D_n^t) - R_e(D_n^{t-1}) \tag{5}$$

An illustration of the piecewise function of R_e is shown below.

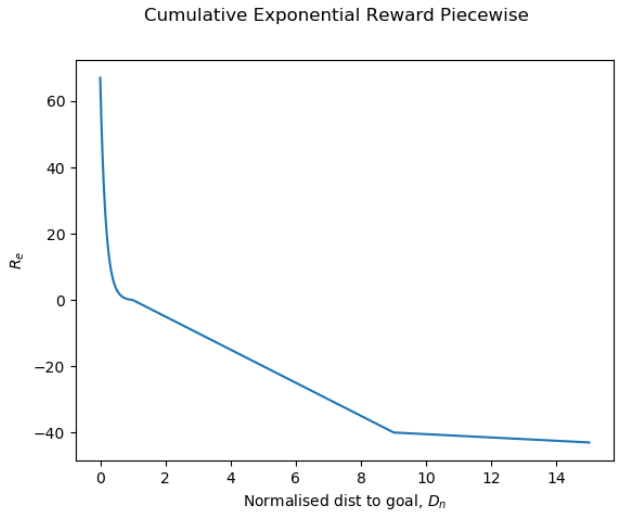

Figure 4: Plot of full cumulative exponential reward, R_e, k_{exp}=6, $D_n = [0, 1.0]$

3.4 Reach Reward, r_{reach}

This is user defined, and is added onto the reward before any penalties are applied. This means that penalty multipliers will be applied to this value.

4 Penalties

Penalties are handled after all rewards are calculated. Penalty multipliers are applied before fixed penalties are applied.

4.1 Orientation Penalty

4.2 Height Penalty