**Initial Distance Minimizer Script**

**Structure**

* Randomly choses an action to move NESW
* When action brings it closer to target, increases likelihood of that action in the future, otherwise decreases
* Run within SITL simulation

**TESTS**

* Drone takes off to 2 m then goes 20 m N, 25 m E
* Tests run on returning home
* Considered done when drone is 3 m total from home location

**Basic Action Weight Shifting**

* Start equal (weight list is (25, 25, 25, 25))
* If moving north brings you closer to the target, shift a constant weight from S to N
* Works sometimes for constant weight shift of 10
  + Gets back to home in a couple minutes ideally
  + Can often overshoot target, BIG PROBLEM
  + Note: Weight list doesn’t have to add to 100, they are just relative weights. Weights can be negative.
* Countering Overshoot issue
  + Cap weights
  + Change action distance according to proximity

Basic Action Weight Shifting with Capped Weights

* 1 for 1 successful returns to home with weight range (10, 40)
  + 340 seconds
  + Effective range is (15, 35) due to starting weights of 25 and jump size of 10
* 3 for 3 successful returns to home with weight range (5, 45)
  + 280, 277, 262 seconds
  + Correctly handled overshoot!
* Often slower than an ideal non-capped run because there is always a chance of backtracking, but far more reliable

Basic Action Weight Shifting with Capped Weights and Proximity-Based Action Distance

* Action distance = distance from target \* proximity factor
* 2 for 2 successful runs with proximity factor 10% minimum action distance 1 m
  + 184, 175 seconds
* 1 for 1 successful run with proximity factor 20% min dist 1
  + 280 seconds
  + Big initial movements might take drone out of range of the beacon, not ideal
* 1 for 1 successful run with proximity factor 15% min dist 1
  + 200, 284, 137 (!), 157, 135 (!), 235 seconds
    - 284 sec run was stuck due to move altitude bug for a while, maybe ignore for comparative purposes

**Scaled Action Weight Shifting**

* Weight shift scaled based on some impact factor
* Impact factor can be something like change in distance / last action distance
  + 1 if action was directly towards/away from target
  + ~0 if action was perpendicular to target direction
* 4 for 4 successful runs with impact factor = change in distance / last action distance
  + Weight shift is 20 \* impact factor
  + 133, 181, 130, 183 seconds
    - These were runs with float comparison bug (target weight was rarely in range, as it was a float)
  + WITHOUT BUG
    - 132, 118, 118, 118 seconds
  + After Impact Factor changed to use real travel distance instead of target travel distance (so that it is always between 0 and 1)
    - 250 seconds
* 1 for 1 successful run with weight shift of 40 \* impact factor
  + 136, 195 seconds

**Additional Improvements for This Algorithm**

* Shifting weights among all choices (not just opposites)
  + Can cause shift according to impact factor
    - Impact factor of 1 indicates action was directly towards target, can reduce likelihood of perpendicular actions greatly
    - Impact factor of 0 indicates action was perpendicular to target, can increase likelihood of perpendicular actions greatly

**REVISITED: IMPACT FACTOR**

* Basically, the same concept as ML gradient
  + “How important is that direction for gradient decent?”
* Can use this to determine weight redistribution between N,S,E,W
* Example
  + Moves north, impact factor 75%
    - Now south impact factor would be -75%
    - Leave remaining 25% to E/W (maybe tweak this number?)
    - We know N/S movements are more important by a decent margin
    - Decrease S,E,W weights (South the most), increase N weight by sum

**NEXT ALGORITHM TO TEST: Velocity Control**

* Kinda a 2-axis PID system
* Change N and E velocities according to distance from target and rate of change of distance (and maybe cumulation of distance, but that seems unnecessary until doing a fine sweep which is likely to be done by the human)
* Might be easiest and most reliable to do distance minimization with N/S movement then E/W movement