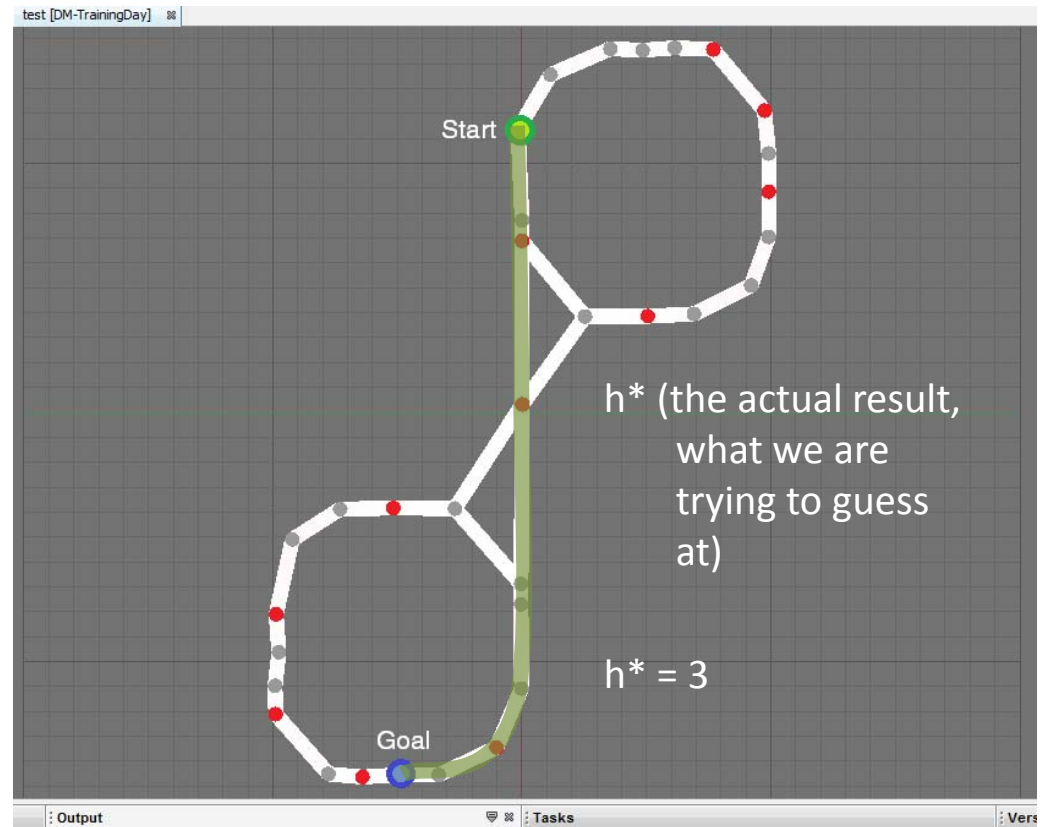
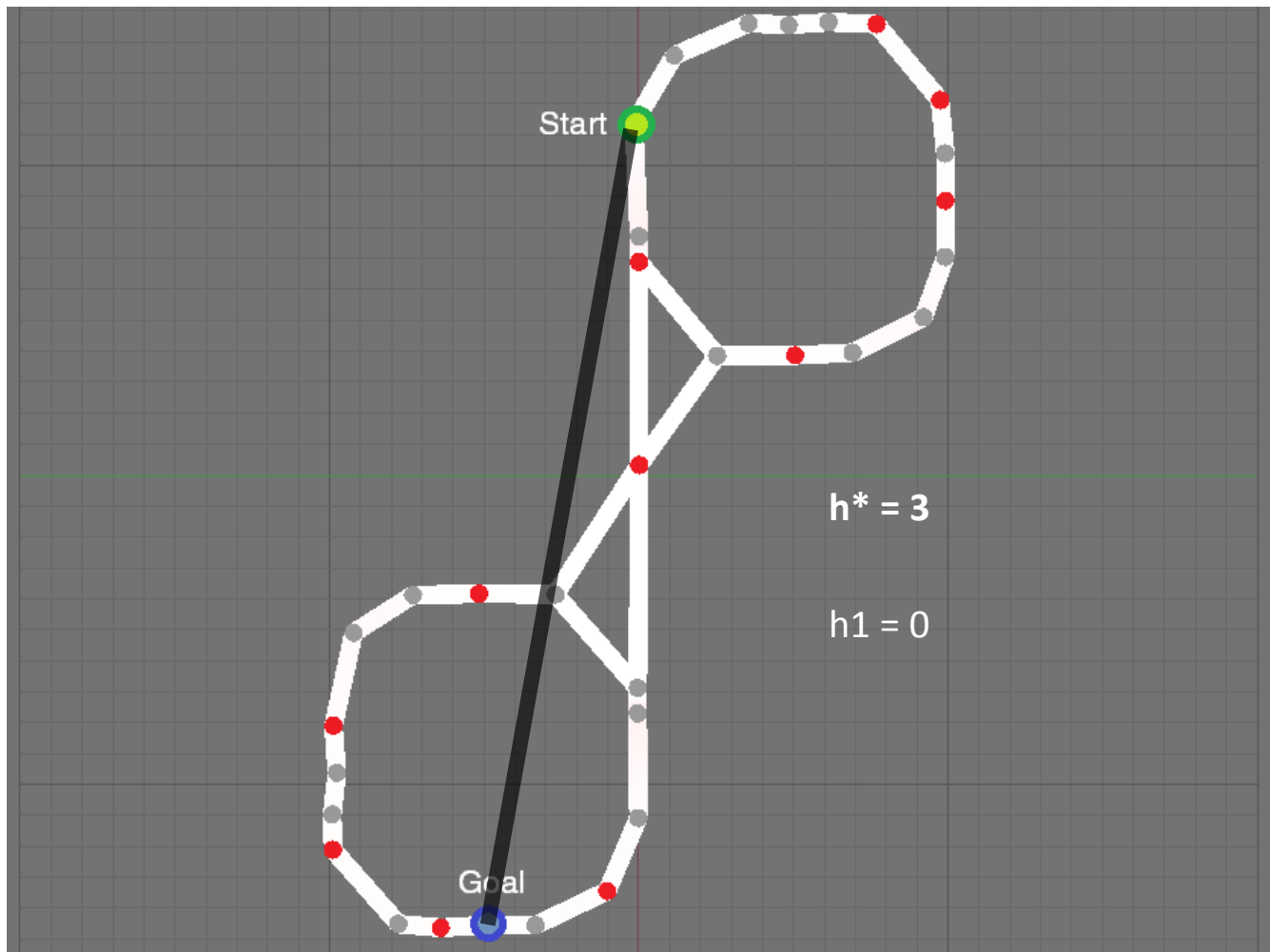


The Heuristic Question

- Imagine a bot is standing on the node marked as Start, and his goal is the node marked Goal.
- To improve A* performance, since our "shortest-path" takes into account **not** only distance but also features, we need a heuristic to find out how many such features the bot will pass on his journey, **without** knowing the exact path he's going to take.
- In other words, some function h , which has access to what the bot knows about the world, at **the time before the path is calculated**, and returns an integer, which is the "best guess" regarding how many features he'll pass on his way.





First idea – “Air-distance”

When A* uses just regular path-planning, h is usually the shortest distance. This is good guess cause we know the actual distance must be higher than this value.

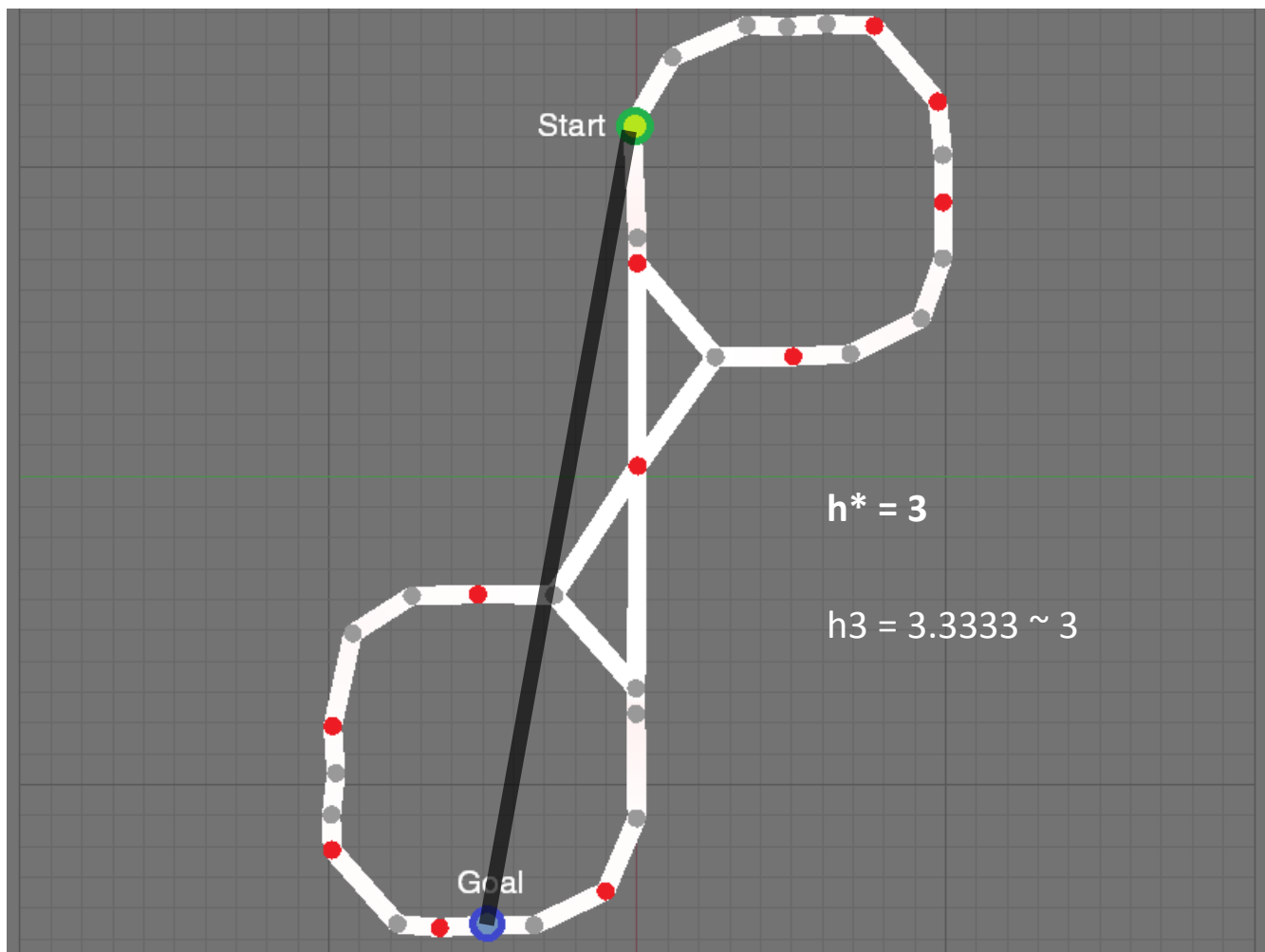
Does the shortest line to the goal pass through packs?

Answer: In the above case, we would get $h=0$, where the result we're searching for is $h=3$.



This is problematic for a number of reasons:

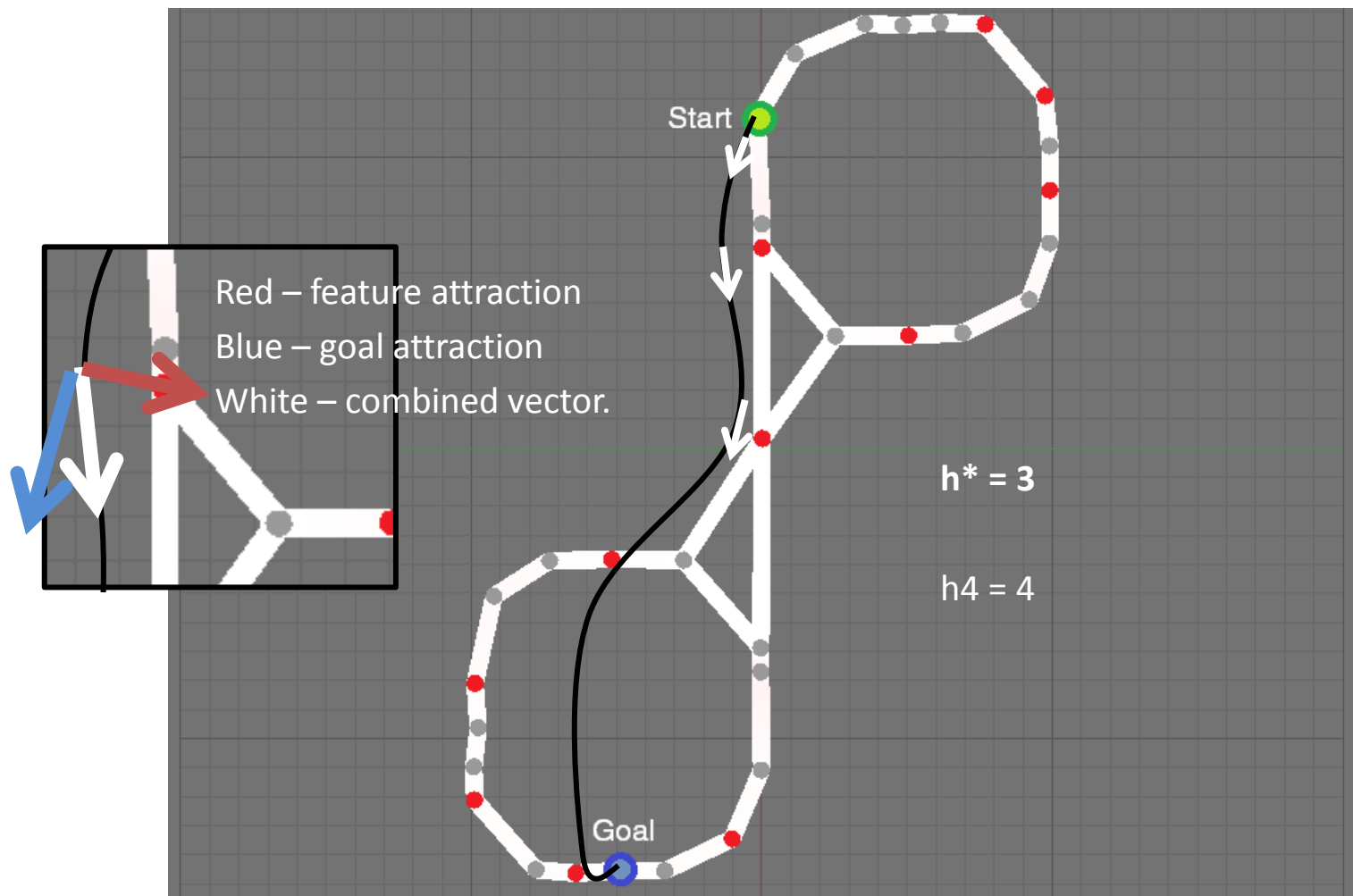
- 1) This really has little to do with the real path the bot is going to take, and therefore detached from the true answer.
- 2) Guessing over in A^* is actually a lot worse than guessing under, cause it's makes the final path less optimal.



Third idea – Probability “Air-distance”

- We take our best guess at how much **distance** the bot is going to traverse.
- We precalculate how many of each feature there is per square-foot.
- If the bot travels say 100 ft, and we know there is a healthpack per 33 square ft, $h=100/33 \sim 3$.

This seems to work the best, but still can over estimate if the actual path is sparser than the “average” spread.



Fourth idea – Potential Fields

- Imagine a particle sets off from the start node with velocity v , pointed towards the goal. Each feature has a “gravity effect” which attracts the particle towards it, but not as much as the particle is attracted towards the goal.
- When the particle finally reaches the goal, how many features did it pass?

Seems complicated and perhaps computationally horrendous. On the other hand, exact and might even give a decent guess towards the actual path or even just a better distance guess.

Can you think of a better one?

