DYNAMIC GAP · TRAJECTORY EVALUATOR

Two gap cost functions

What was broken, what replaced it, and what to expect when you tune it

Gap width clearance

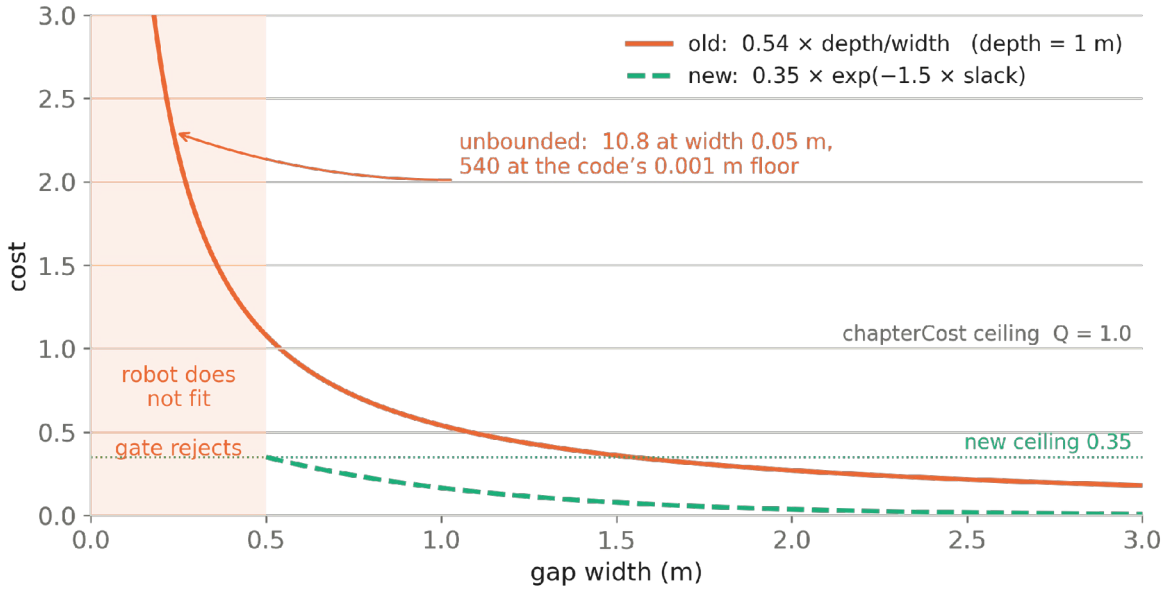
Gap endpoint proximity

All figures measured from a 50-episode run of gru_learn_scenario_11. Worked numbers assume r_inscr = 0.25 m with inf_ratio = 1.0.

The old score had no upper limit

The old rule scored a gap by depth ÷ width, and charged some weight (0.54) × that.

- The direction was right — narrow, deep gaps did cost more.
- But nothing capped it. At a 1 mm width the score reaches 540.
- Every other term in the evaluator stops somewhere. This one did not.
- Depth sat on top of the fraction, so a deeper gap always cost more.



Summary: Penalizing Depth:

Anything on top of a fraction makes the number bigger. So more depth always meant more cost — there was no combination of numbers where growing the depth made a gap cheaper.

Same doorway width, same tightness, same danger — but the deep one is punished 1.35 more than the shallow one.

Measure the leftover room, then cap the penalty

1

Work out the width the robot needs

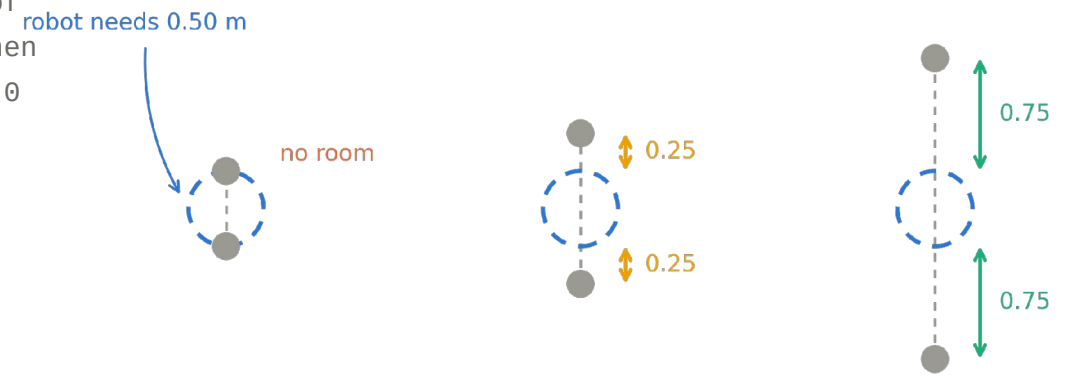
```
requiredGapWidth = 2 × (r_inscr × inf_ratio) + margin  =  0.50 m
where: r_inscr = radius of the biggest circle that fits inside of
the robot's body i.e. 2 x robot's width = safe clearance. And then
inf_ratio = Safety multiplier which is set at 1.0 and margin is 0
```

2

Subtract it from the gap

```
slack = gapWidth − requiredGapWidth
```

slack = gapWidth − requiredGapWidth



3

No room? Reject it. Some room? Price it.

```
slack ≤ 0  →  +∞  (gate)
slack > 0  →  C_w = 0.35 × exp(−1.5 × slack) i.e. Gap width
clearance cost. C_w comes from the same idea of terminal cost i.e.
```

chapterCost: Q * exp(-pen_exp_weight * inflRbtToScanDist)

Every 0.67 m of extra room divides the penalty by e ≈ 2.7.

0.75 m → 0.241 1.00 m → 0.165 1.50 m → 0.078 2.00 m → 0.037

4

Reject gaps where measured width < required_gap_width

```
the gap is rejected outright instead of being given a price.
Previously nothing anywhere checked whether the robot physically
fit through a gap.
```

1. Depth is gone! Was already being done by the goal term: Deeper gap → trajectory ends further along → closer to the waypoint → lower `Q_f` cost

What it should buy us

1

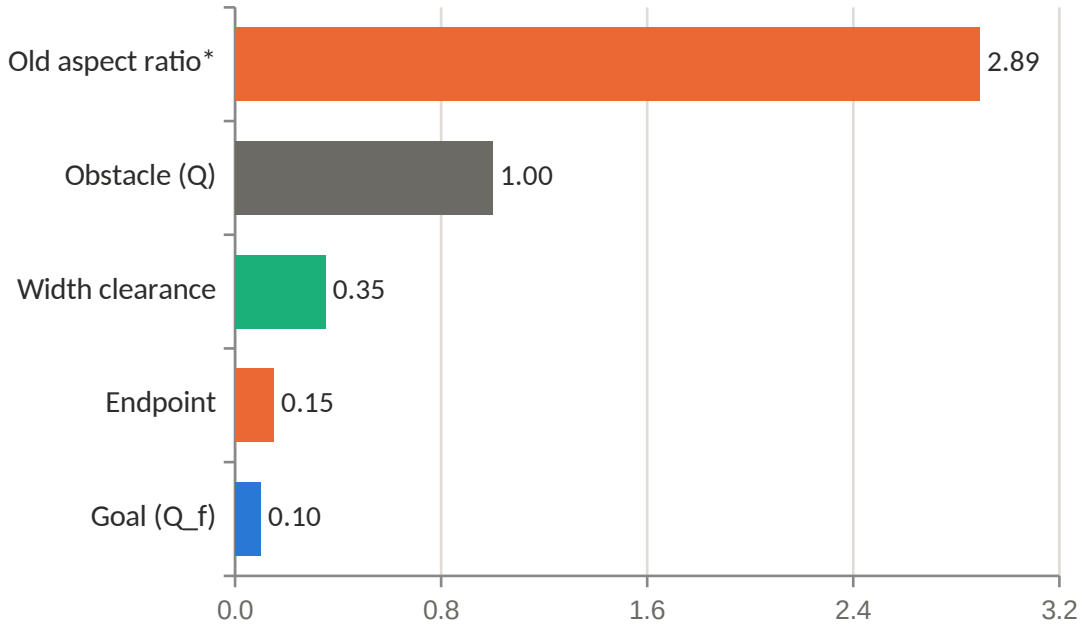
Never look at a gap the robot cannot fit through

Nothing rejected on width before. chapterCost only fires at contact when inf_ratio = 1.0.

2

Reward room without punishing progress

Depth is out of the cost entirely. It survives only as the RViz AR= label.



** worst observed in the run; the formula has no ceiling at all*

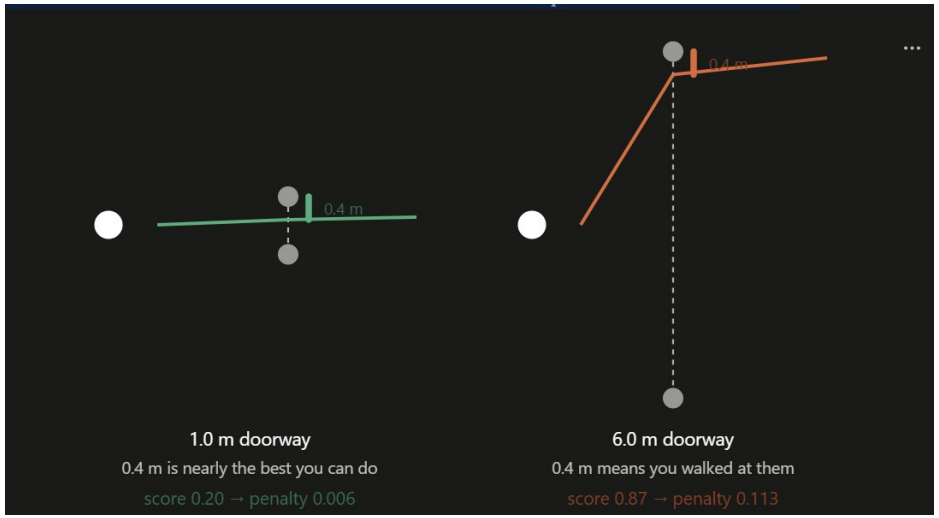
Explained Intuitively

Two people are standing in a corridor and you have to walk between them.

You can walk **straight down the middle**, or you can **brush past one of them**. Everyone agrees brushing past is worse. The job is to turn that feeling into a number. The obvious idea is to measure centimetres to the nearest person. That doesn't work:

Both orange and teal bars are the same length — the same 0.4 m. But in the small doorway that's almost

the best you could manage, and in the big one you've walked nearly straight at somebody.



Centimetres can't tell those apart. So don't use centimetres.

Use a score out of 1 instead

Ask a different question: *how much of the room that was available to you did you actually use?*

- Dead centre → you used all of it → score **0** → no penalty
- Halfway out → score **0.5**
- Touching someone → you used none of it → score **1** → full penalty

That's the whole idea. Everything else is arithmetic to produce that score.

To get it, you take how close you came, divide by half the doorway (because half the doorway is the most room you could possibly have kept), and subtract from 1:

- Small doorway: `0.4 ÷ 0.5 = 0.8` → you kept 80 % of the room → score `1 − 0.8 = 0.20`
- Big doorway: `0.4 ÷ 3.0 = 0.13` → you kept 13 % → score `1 − 0.13 = 0.87`

Same 0.4 m, completely different scores. That's step 2 doing its job.

Then: Square the score. A tiny wobble shouldn't be nagged at, a near-miss should. Squaring turns 0.2 into 0.04 and leaves 0.8 at 0.64.

Multiply by 0.15. That just sets the biggest fine you're ever willing to hand out.

Measure the closest pass, as a share of the doorway

1 **Closest approach, over the whole path**

```
d_min = min over poses i ≥ 1 of distance to p_l or p_r
```

2 **Divide by half the width**

```
offset = clamp(1 − d_min ÷ (gapWidth / 2), 0, 1)
```

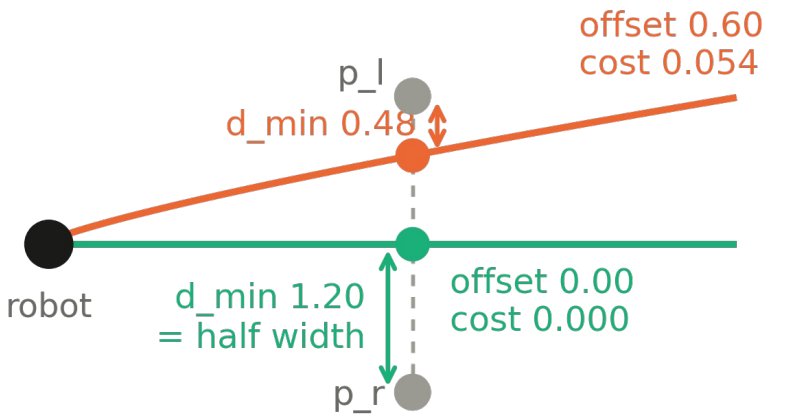
3 **Apply a capped penalty**

```
C_ep = 0.15 × offset²      always between 0 and 0.15
```

4 **Width cancels out**

```
offset = 1 − 2 × min(k, 1−k)      no gapWidth left
```

new: closest pass ÷ half the width



a centred pass scores 0 at ANY width