Online Motion Planning, SS 16
Exercise sheet 5
University of Bonn, Inst. for Computer Science, Dpt. I

- *You can hand in your written solutions until Wednesday,* 25.5., 14:15, *postbox in front of room E.01 LBH.*
- *We allow (and recommend) fixed groups of 2 students.*
- *Please subscribe to our mailing list:*
  https://lists.iai.uni-bonn.de/mailman/listinfo.cgi/vl-online

Exercise 13: Proof detail ccw-order turn (4 points)

In the lecture there was a proof that shows that a curve from $\mathcal{K}$ cannot have a self-intersection. We considered the case of a clockwise turn. Now consider the case of the counterclockwise turn (as in Figure 1) and analogously give the proof.

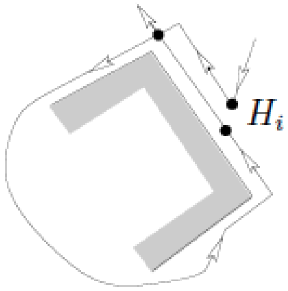

Figure 1: A counterclockwise loop and an intersection!

Exercise 14: Detect vertical edges (4 points)

We introduced δ-pseudo orthogonal scenes with sensor error ρ and provide some sufficient condition in a **Corollary** for detecting a horizontal edge.

a) By the same arguments and conditions, show that we can also correctly detect a vertical edge.

b) Analogously, for moving along the boundary give sufficient conditions for detecting the case that two adjacent edges are *both* vertical edges (in principle).

Exercise 15: Bug leaving from closest vertex (4 points)

We consider a modification to the *BUG* algorithm, given that the obstacles are simple polygons in the plane. The bug starts at its starting point s. In order to reach destination point t, the bug moves in direction of t, until an obstacle O hinders its movements. As usual, the bug walks along the boundary of O and keeps track of the distance to t.

The modification is as follows. Instead of leaving O at a point closest to t, the bug leaves O at a *vertex* v of O's boundary which is closest to t. Then, the bug continues in direction of t, until it encounters another obstacle.

Prove or disprove that the modified *BUG* algorithm will eventually reach the target point t, although possibly not as quickly as the unmodified algorithm.

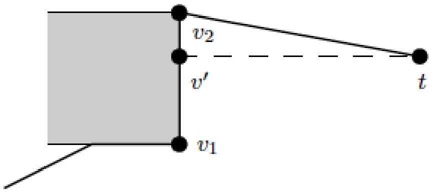

Figure 2: Bug-Variant: Leaving from the closest vertex v_2!