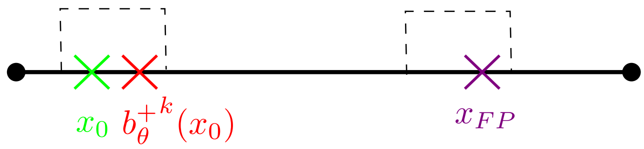
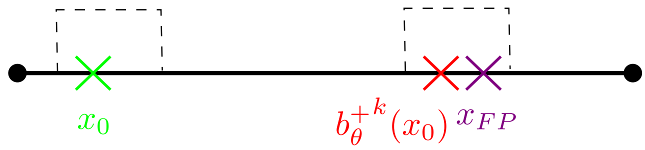
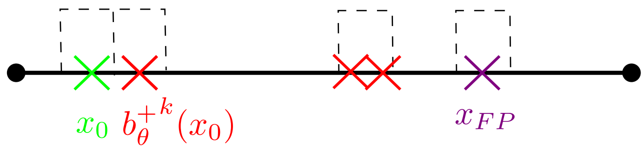


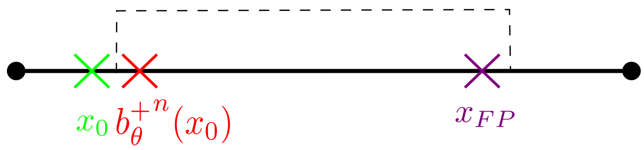
(a) Finds a cycle that is not a periodic orbit.



(b) Finds the real periodic orbit.



(c) Increasing resolution.



(d) Decreasing resolution.

Figure 1: Resolution analysis.

In Figure 1 we show an edge length l for a given n-gon (polygon with n sides). We assume that the robot hits that edge for the first time at point x_0 and that there exists a stable limit cycle of period n that hits the edge at point x_{FP}. Let $b_\theta^{+k}(x_0)$ be the hit point on the polygon boundary after iterating k times the bouncing map $b_\theta^{+}(x_0)$, starting from x_0. Finally, we show as dotted squares the cells for a given discretization.

In simulations that consider regular polygons with fixed bouncing policy, for the right bouncing angle intervals, only one stable limit cycle shall appear for each θ. When the algorithm finds more than one periodic orbit, only one of them can be the sought limit cycle. The other orbits appear as an artefact of the current discretization resolution. For certain values of θ, the convergence rate towards x_{FP} is quite small, resulting in hit points $b_\theta^{+k}(x_0)$ that slowly move away from x_0. For example, consider Figure 1a and assume that $b_\theta^{+k}(x_0)$ has already returned to the n-gon edge e. For a given resolution and a certain θ, x_0 and $b_\theta^{+k}(x_0)$ can fall in the same cell z, which would result into an apparent periodic orbit, but z is indeed a transient cell. Conversely, in Figure 1b, the algorithm has iterated enough such that $b_\theta^{+k}(x_0)$ and x_{FP} are in the same cell z, so any $b_\theta^{+k+jn}(x_0)$ would stay in z, hence, z is truly a persistent cell.

To avoid the issue present in Figure 1a, the discretization resolution could be increased such that x_0 and $b_\theta^{+k}(x_0)$ fall into different cells z and w, just as seen in Figure 1c, therefore, z and w would be classified as transient cells. Nonetheless, with this finer discretization there is no guarantee that as $b_\theta^{+k}(x_0)$ keeps iterating, two consecutive hit points $b_\theta^{+jn}(x_0)$ and $b_\theta^{+(j+1)n}(x_0)$, which are getting closer to x_{FP}, fall in the same cell y yielding again a spurious periodic cycle. The resolution could again be increased but this also has the inconvenience that as the cells become smaller, the algorithm will need more iterations in order to find the real stable limit cycle.

Based on the last analysis it is desirable to have large enough cells to avoid over iterating the algorithm. Following this vein, setting $k = n$ is a suitable election, as it is the minimum k that guarantees the robot to return to the edge where x_0 is located, opening the possibility of closing a cycle. Inequality (1) can be used to compute an actual bound for the distance between $b_\theta^{+k}(x_0)$ and x_{FP} as a function of k; again, setting $k = n$, yields a bound that can be used to select the actual size of the cells. For instance, suppose that in Figure 1b, the point $b_\theta^{+k}(x_0)$ has $k = n$. If that is the case, it can be seen that the shown cell size is enough to contain distance $d(b_\theta^{+n}(x_0), x_{FP})$, so the algorithm would not identify spurious periodic orbits and would rapidly find the persistent cell that contains x_{FP}. On the other hand, if the distance between $b_\theta^{+n}(x_0)$ and x_{FP} is larger, it would require bigger cells just as shown in Figure 1d. In either case, selecting the cell size according to distance $d(b_\theta^{+n}(x_0), x_{FP})$ (actually according to its bound $c(\theta)^n l$), makes the algorithm identify a single limit cycle with fewer iterations.

Theorem 1 *Assume an adjacent edges fixed bouncing scheme, and assume that the robot starts over δP at distance $x_0 \in (0, l)$ from vertex v_i. When the robot encounters δP for the k^{th} time, then*

$$d(b_\theta^{+k}(x_0), x_{FP}) < c(\theta)^k l. \tag{1}$$

Corollary 1 *If the initial distance x_0 is known, then $c(\theta)^k d(x_0, x_{FP})$ gives the exact bound to $d(b_\theta^{+k}(x_0), x_{FP})$.*