BeachCrab: An Autonomous Beach-Cleaning Robot — Simulation Design and Results

Yashwanth Gowda

September 16, 2025

Abstract

A MATLAB-based small-scale simulation of an autonomous beach-cleaning robot (*BeachCrab*). The agent patrols a two-dimensional grid, detects metallic and non-metallic debris, avoids obstacles via reactive turning, and enters a high-speed *leap* mode when long clear corridors are sensed. Using provided runs (n=5), we report collection rates and counts, give precise governing equations for kinematics and mode switching, and include representative figures produced by the simulator. Across runs, the mean final collection rate was 17.20% (std 4.50%), with averages of 10.00 metallic and 18.00 non-metallic items collected per run (total 28.00).

1 Introduction

Marine litter on beaches threatens ecosystems and safety. Mechanical beach-cleaners exist, but small, autonomous agents could operate in narrow corridors and fragile habitats. *BeachCrab* is a minimal agent: a point robot with short-range sensing and simple control that nevertheless exhibits effective local coverage. This paper documents the environment model, the robot's states and dynamics, sensing and control policy, and the evaluation metrics used to analyze performance.

2 Methods

2.1 Environment

The world is a rectangular grid of width W and height H. Each cell encodes one of four states: sand (0), metallic debris (1), non-metallic debris (2), or obstacle (3). Debris and obstacles are randomly placed without overlap. The robot starts at (x_0, y_0) with heading θ_0 (degrees).

2.2 Robot State and Kinematics

The robot state at time step t is (x_t, y_t, θ_t). With step speed v_t, the (clipped) kinematics are

$$x_{t+1} = \mathrm{clip}\big(x_t + v_t \cos(\theta_t),\, 1,\, W\big), \tag{1}$$

$$y_{t+1} = \mathrm{clip}\big(y_t + v_t \sin(\theta_t),\, 1,\, H\big), \tag{2}$$

$$\theta_{t+1} = \theta_t + u_{\theta,t}, \tag{3}$$

where $u_{\theta,t}$ is an angular increment determined by the control mode (defined below). Two speeds are used: a nominal v_{norm} and a faster $v_{\mathrm{leap}} \gg v_{\mathrm{norm}}$.

2.3 Sensing

Within a square (or circular) window of radius R around (x_t, y_t), the agent detects obstacles and debris. If any obstacle is observed in this window, a turning behavior is activated. If debris is observed and no obstacle preempts the choice, the heading is biased toward the detected debris cell (nearest-first). When neither is observed, the agent evaluates an extended window of radius R_{leap} along the current heading to test for long, obstacle-free corridors.

2.4 Control Policy

At each step, one of the following modes is active:

1. **Obstacle avoidance**: turn in the direction $s \in \{-1, +1\}$ using a fixed rate ω_{turn}; i.e., $u_{\theta,t} = s\,\omega_{\text{turn}}$.
2. **Debris pursuit**: orient toward the nearest debris cell's bearing ϕ_t, with $u_{\theta,t} = \text{wrap}(\phi_t - \theta_t)$.
3. **Exploration**: small random walk bias $u_{\theta,t} \sim \mathcal{U}(-\Delta, \Delta)$.

Speed selection:

$$v_t = \begin{cases} v_{\text{leap}}, & \text{if the corridor of length } R_{\text{leap}} \text{ is free of obstacles,} \\ v_{\text{norm}}, & \text{otherwise.} \end{cases} \tag{4}$$

2.5 Collection and Termination

When the rounded position coincides with a debris cell, the item is removed and the respective counter is incremented. A run terminates after a fixed horizon $T_{\max}$ or when all debris is collected. Let M and N be the counts of metallic and non-metallic items collected, respectively, and $T = M + N$. If T_{total} is the number of debris items initially placed, then the final collection rate is

$$\text{rate} = \frac{T}{T_{\text{total}}}, \qquad (\%\ \text{collected}) = 100 \times \frac{T}{T_{\text{total}}}. \tag{5}$$

3 Calculations and Summary Statistics

Across n =5 runs, we compute the empirical mean, standard deviation, and range of the final collection rate, as well as mean per-class counts:

Metric	Value
Mean final rate (%)	17.20
Std. dev. final rate (%)	4.50
Min–Max final rate (%)	10.30 – 21.00
Mean metallic (count)	10.00
Mean non-metallic (count)	18.00
Mean total (count)	28.00

Table 1: Aggregate statistics across runs.

4 Results

We include representative simulator outputs (bar/pie compositions) and in-run frames.

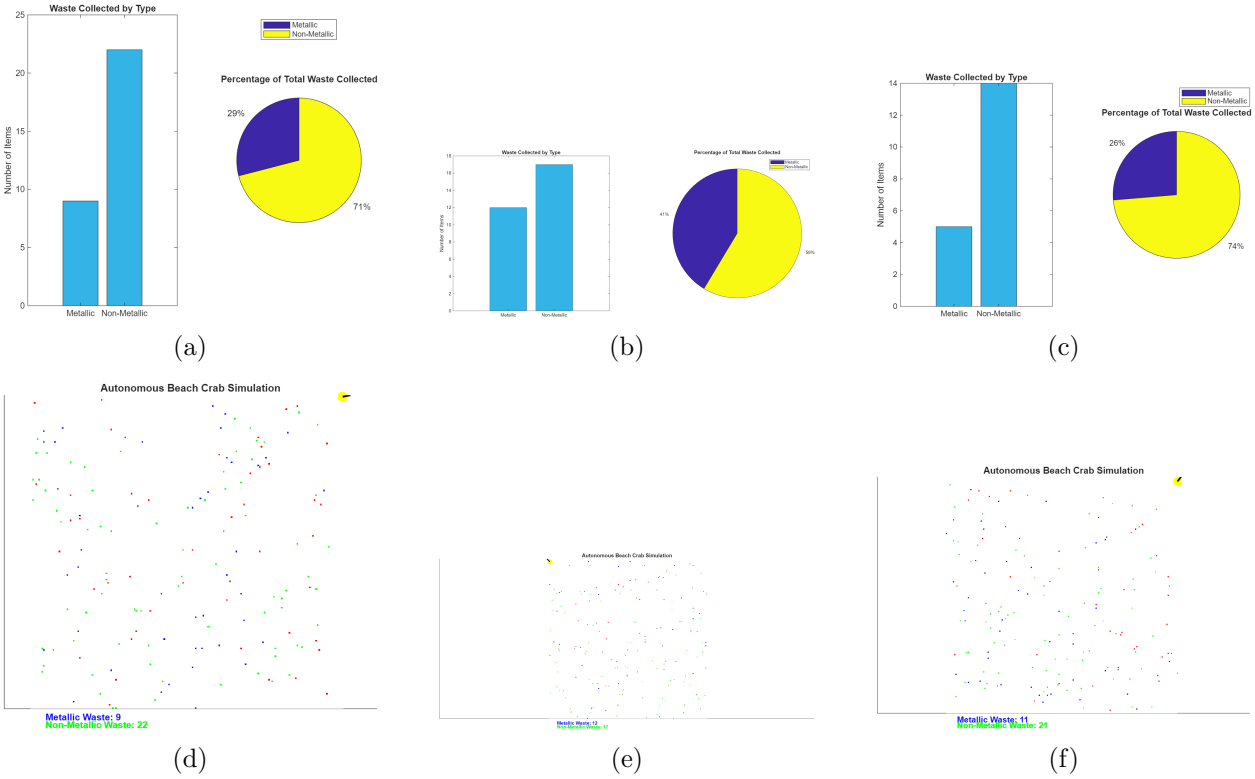

Figure 1: Representative simulator outputs (top row) and simulation frames (bottom row).

5 Discussion

The reactive controller achieves partial coverage without any global planning. Observed collection rates are consistent with local exploration under limited horizons and stochastic placement of debris and obstacles. The *leap* mode plausibly accelerates traversal through open corridors but does not guarantee uniform coverage, especially when debris is clustered behind obstacle fields.

6 Limitations and Future Work

Key limitations include: small number of summarized runs; unspecified or varying random seeds; sensitivity to sensor window sizes (R, R_{leap}), turn rate ω_{turn}, and time horizon $T_{\max}$. Future work: ablation on leap versus no-leap; parameter sweeps; informed exploration (frontier-based or information gain); comparison to global planners (e.g., PRM, RRT); and multi-agent cooperation.

7 Reproducibility Notes

Core MATLAB scripts: world generation, main simulation loop, and plotting utilities. Results CSV aggregates per-run totals and percentages. Ensure consistent seeds and fixed environment parameters for reproducible sweeps.

References

[1] H. Choset, "Coverage for robotics – a survey of recent results," *Annals of Mathematics and Artificial Intelligence*, 2001.

[2] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *IJRR*, 2001.