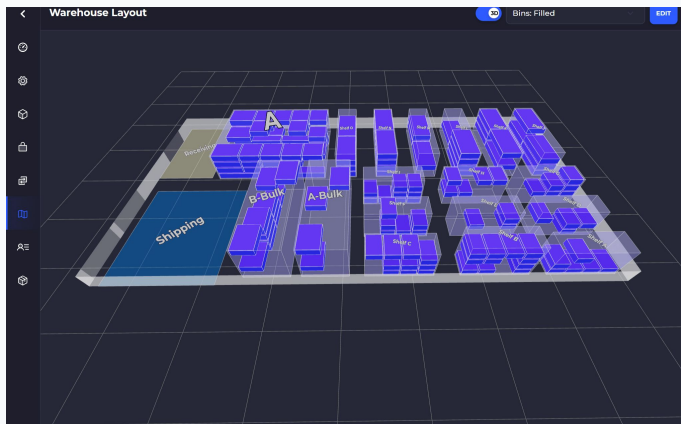


B0: SWARM (Zhuoyi Zou, Rachel Lewis, Kevin Huang)

S**Scalable****W****Warehouse****A****Autonomous****R****Robotic****M****Mapping**

Use Case

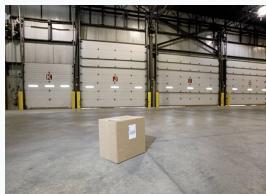
Static Baseline



Warehouse mapping exists

Static 2D layouts represent an idealized representation of the physical warehouse, but do not capture real-time operational shifts

Reality



Boxes can be misplaced

Inventory is frequently left in incorrect corridors or unmapped transit zones



Humans make mistakes

Manual logistics and pilot errors result in deviations from the official layout



"Temporary" piles accumulate

Workflows can cause build-ups of unmapped inventory in staging areas

Use Case



Current AMRs

- Orchestrated by central compute
- Share a pre-built facility map across the fleet
- Facility map updates typically rely on single-robot SLAM



Source: <https://keyslogistics.com/amr-warehouse/>



SWARM

- Multiple robots build global map in parallel



Coverage Areas

- Circuits: Robot Hardware
- Software Systems: Robot Control and Mapping Protocols
- Signals and Systems: Robot Communication and Control

Use Case Requirements



High Accuracy

Mapping achieves high accuracy, ensuring consistent and reliable results



Efficiency

Multiple robots collaborate to complete mapping significantly faster than any single unit.



Decentralized Resilience

Robot swarm functionality is not dependent on any single piece of equipment

Use Case Requirements

High Accuracy

At least 90% IoU of occupancy grid validates high accuracy of mapped obstructions

Under 3cm of median wall-position error validates high accuracy of identified walls

Efficiency

Mapping with 2 robots should lead to a time reduction of at least 40% compared to one robot

Running more robots should not decrease accuracy

Decentralized Resilience

Upon immobilizing or turning off a robot mid-run, we continue mapping to completion

Robots do not rely on the existence of a central compute or "leader"

Technical Challenges

Accurate Distributed Map Merging

Linked Requirement: High Accuracy
 $\geq 90\%$ IoU of occupancy grid
 < 3 cm median wall-position error

Partial distributed maps from different robots need to be aligned and merged without drift or duplicate obstacles.

Mitigation: Use tested distributed SLAM protocols (DOOR-SLAM) designed for accuracy and error tolerance

Efficient Robot Coordination

Linked Requirement: Efficiency
 $\geq 40\%$ time reduction with 2 robots
 $\leq 1\%$ accuracy drop with more robots

Robots must explore different regions to avoid redundant mapping. We need a distributed method to assign (and reassign) exploration regions as the environment is discovered.

Mitigation: Use frontier-based exploration and distributed task allocation. Dynamically re-assign regions when a robot fails.

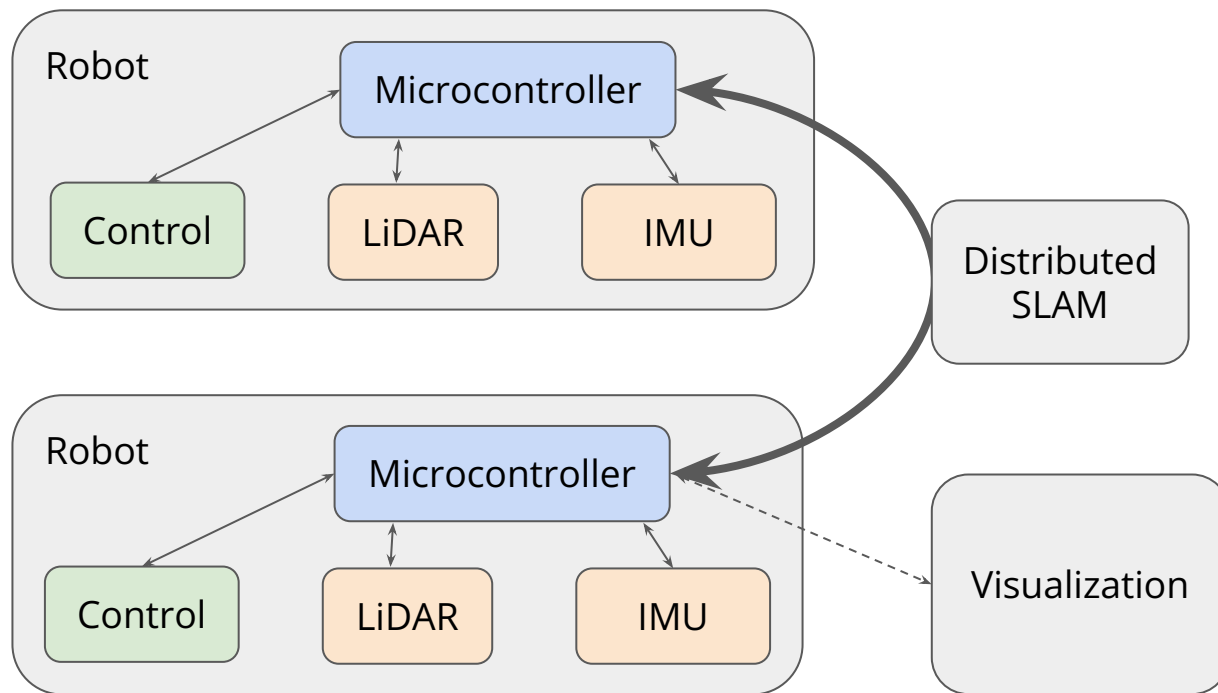
Decentralized Failure Recovery

Requirement: Decentralized Resilience
 Continue mapping after robots fail
 No central leader

Failures must be reliably detected, and the unfinished regions must be redistributed to other robots without relying on central compute.

Mitigation: Use decentralized heartbeats, timeouts, and motion pings for failure detection

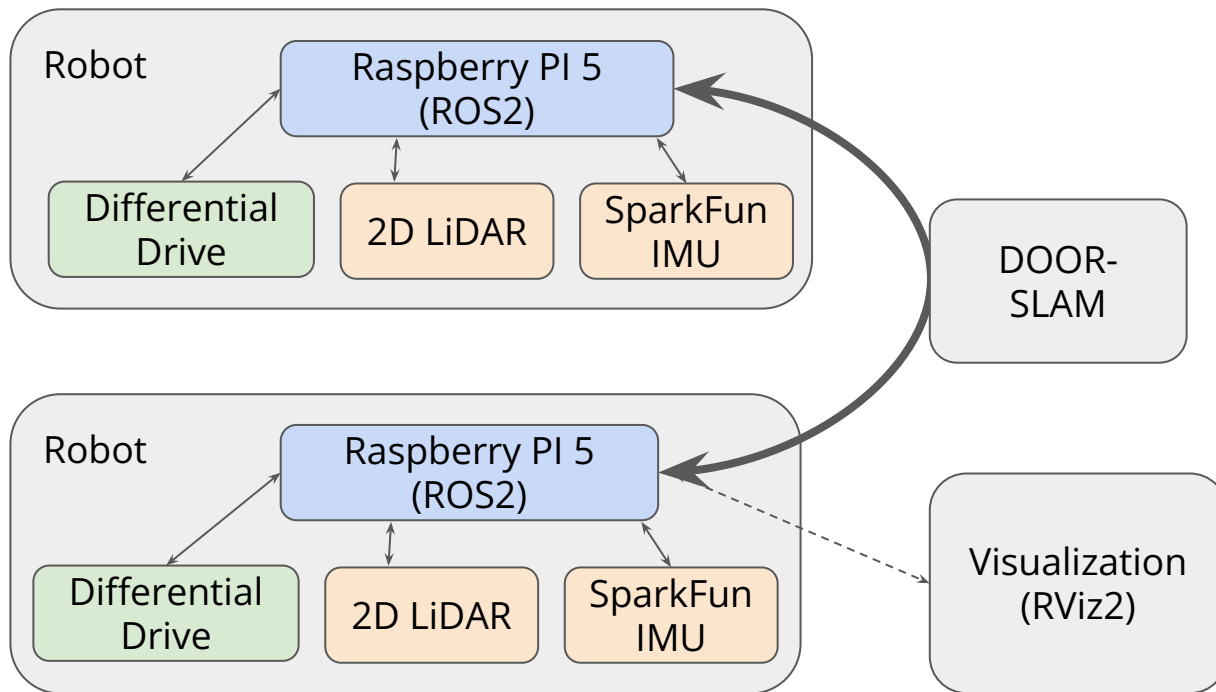
Solution Approach



Source:

<https://techvico.com/how-amrs-use-slam-to-navigate-smartly-in-factories-and-warehouses/>

Solution Approach



Source:

<https://techvico.com/how-amrs-use-slam-to-navigate-smartly-in-factories-and-warehouses/>

Testing, Verification, Metrics

High Accuracy

At least 90% IoU of occupancy grid validates high accuracy of mapped obstructions

Under 3cm of median wall-position error validates high accuracy of identified walls

Layout obstacles in a room with measured sizes and positions ->
Do robot(s) accurately map out positions of all obstacles?

Efficiency

Mapping with 2 robots should lead to a time reduction of at least 40% compared to one robot

Running more robots should not decrease accuracy by more than 1%

Run obstacle test with 1 robot and 2 robots ->
Do 2 robots complete the task faster without decreasing accuracy?

Decentralized Resilience

Upon immobilizing or turning off a robot mid-run, we continue mapping to completion

Robots do not rely on the existence of a central compute or "leader"

Run obstacle test with 2 robots and shut down / immobilize one ->
Does the remaining robot complete the task with the same accuracy?

