Particle Filters and Tracking

Computational Perception and Artificial Intelligence

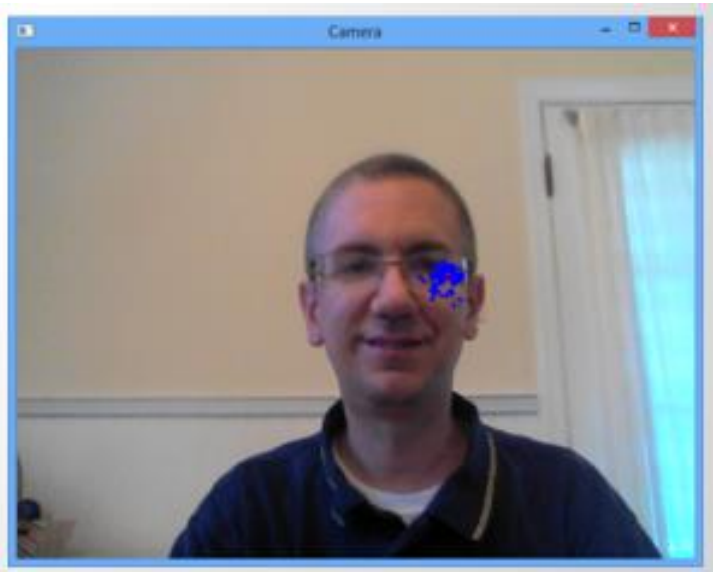
Mr. Michaud

Particle Filters and Genetic Algorithms

- Modeled after Evolutionary Biology
- “Wisdom of the Crowd”
- Extension of a Bayesian Filters
- Used in conjunction with sensor systems, robotics, and Machine Learning Classification problems
- Key Idea: Position of Robot or sensed object is not discrete (a real number). Rather, the predicted position given sensor readings is a probability distribution over space.

Summary of Particle Filter

- Create many (hundreds) of instances (particles) of a model or object.
- Run the model
- Get a measurement
- Calculate a weight for each particle based on the measurement
- Harvest at random according to the weight
- Remove the rest
- Breed the harvested particles
- Repeat!



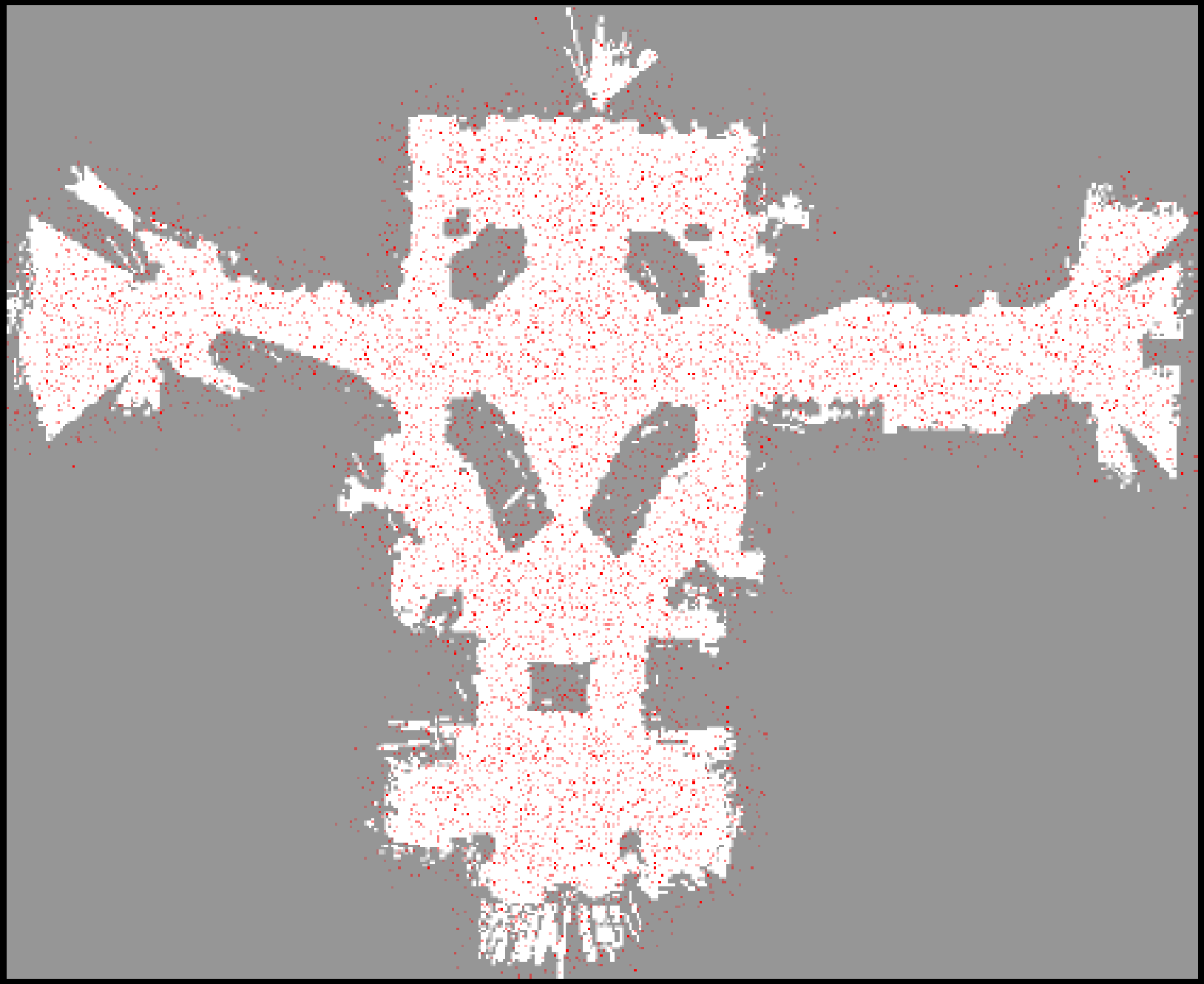
Localization: A robot sensing problem

- Assume a robot **knows** a 3D map of its world.
- It has noisy depth sensors but whose sensing uncertainty is ***known***.
- It moves from frame to frame.
- How well can it know where it is in (x, y, θ) ?

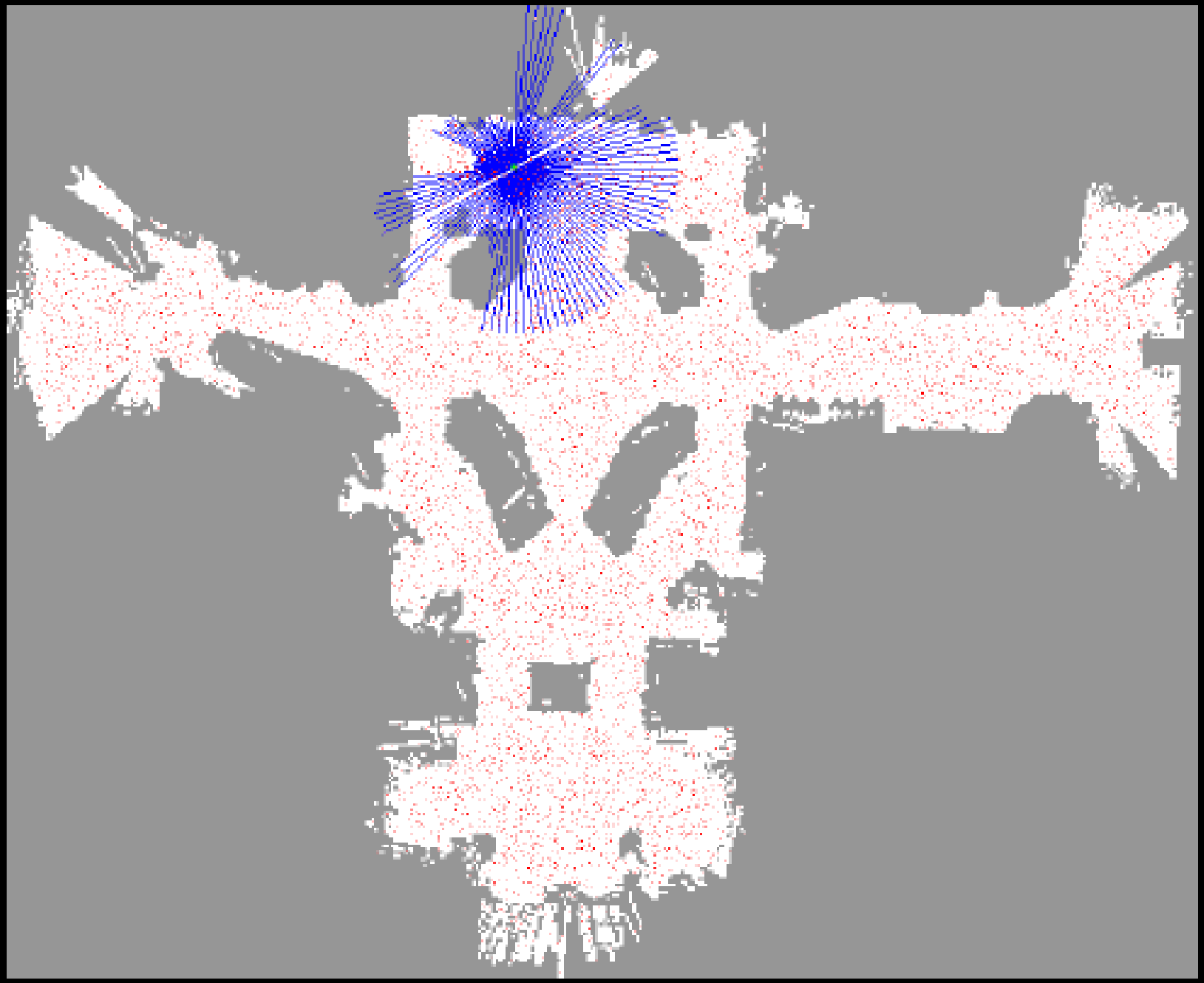
Sonar-based Localization Example

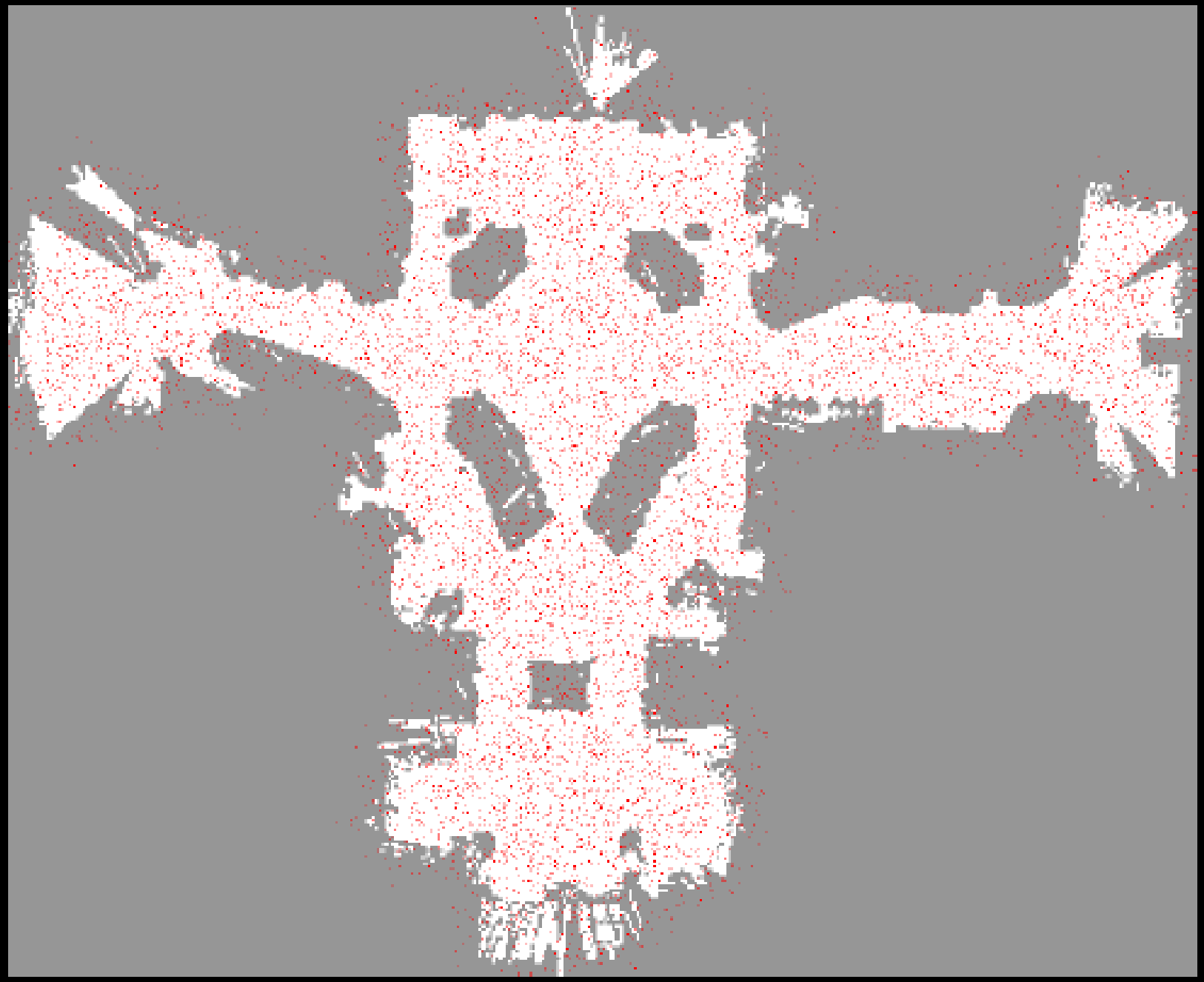
At the *Smithsonian Museum of American History*...

Initial prediction

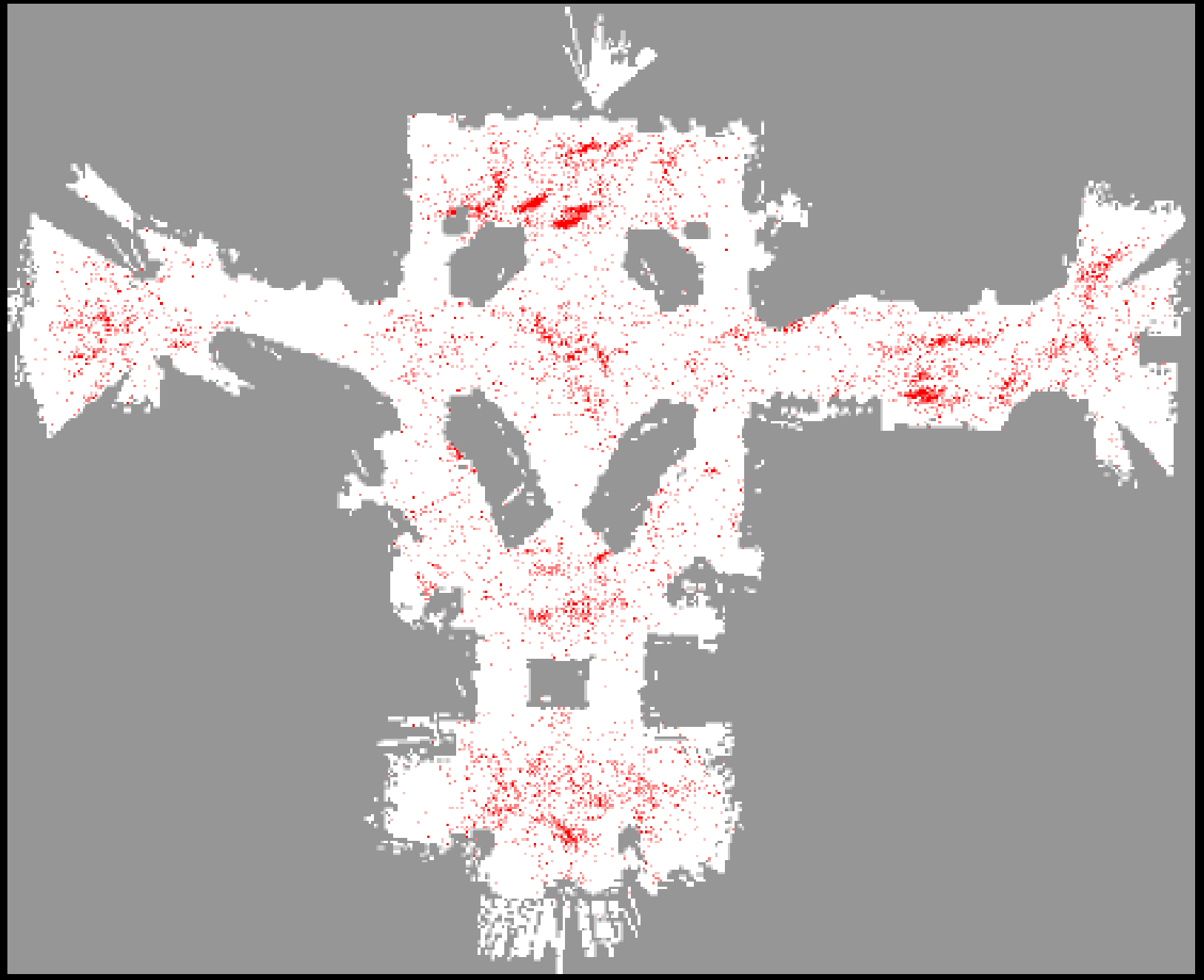


First observation

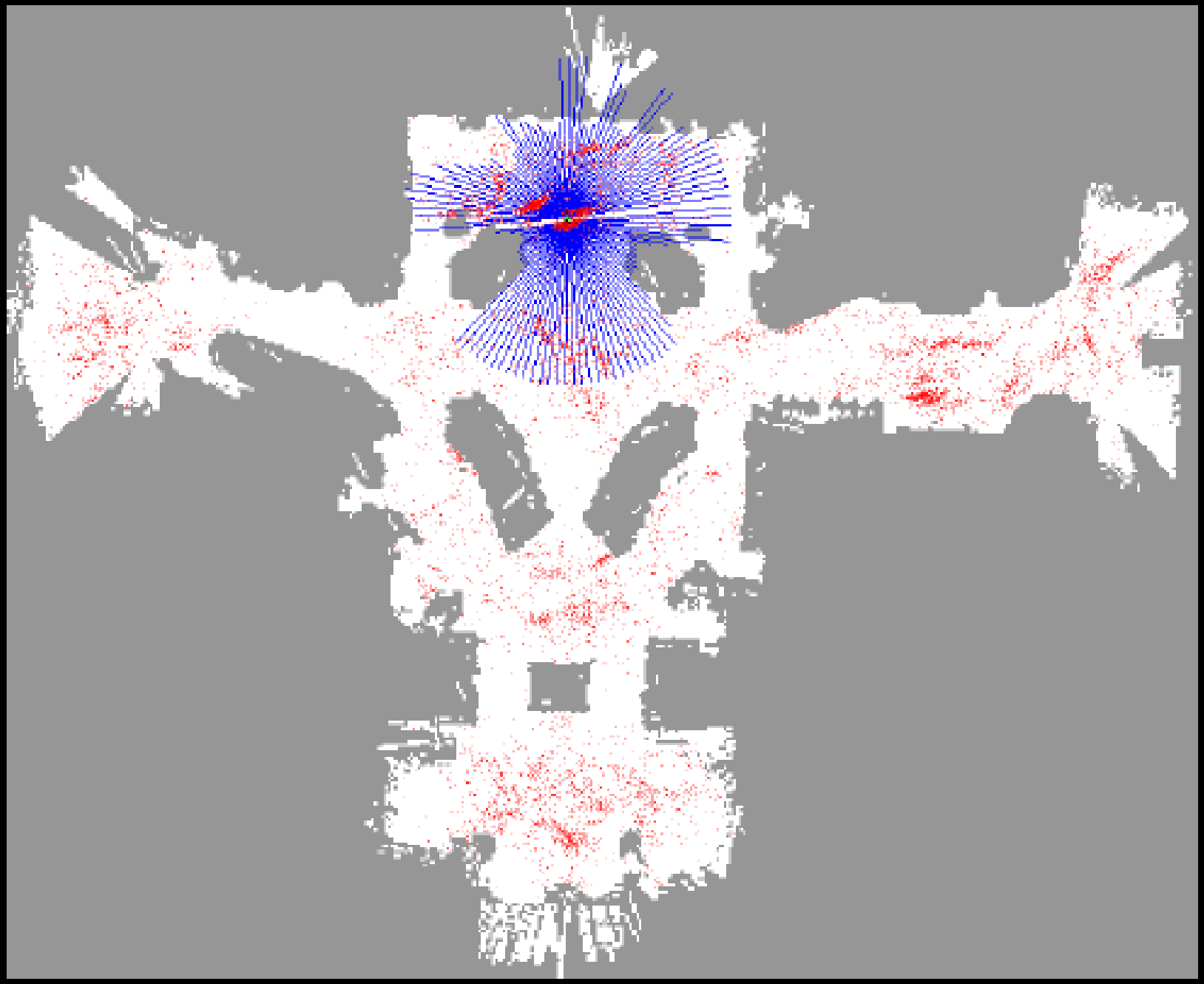


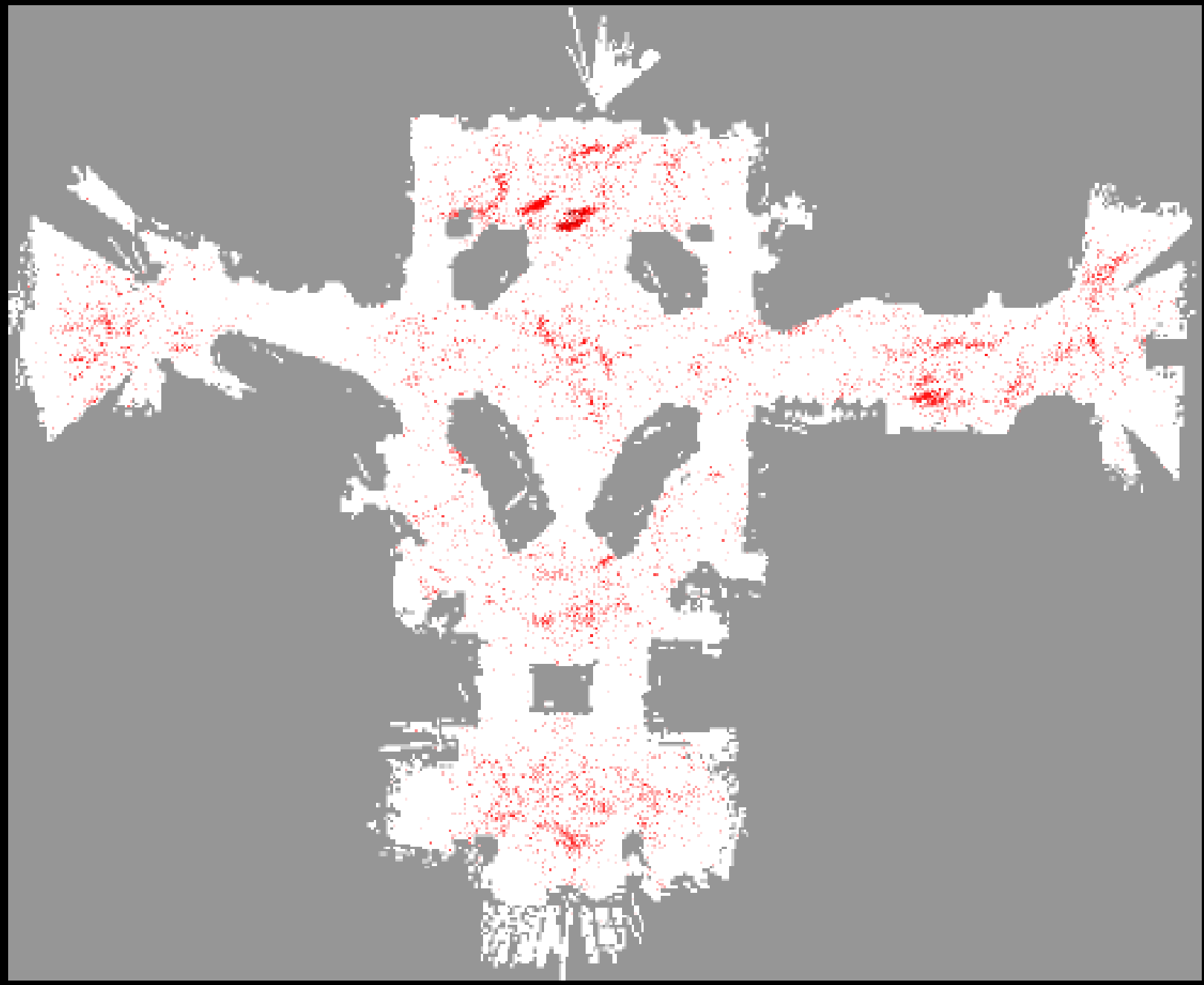


Corrected prediction

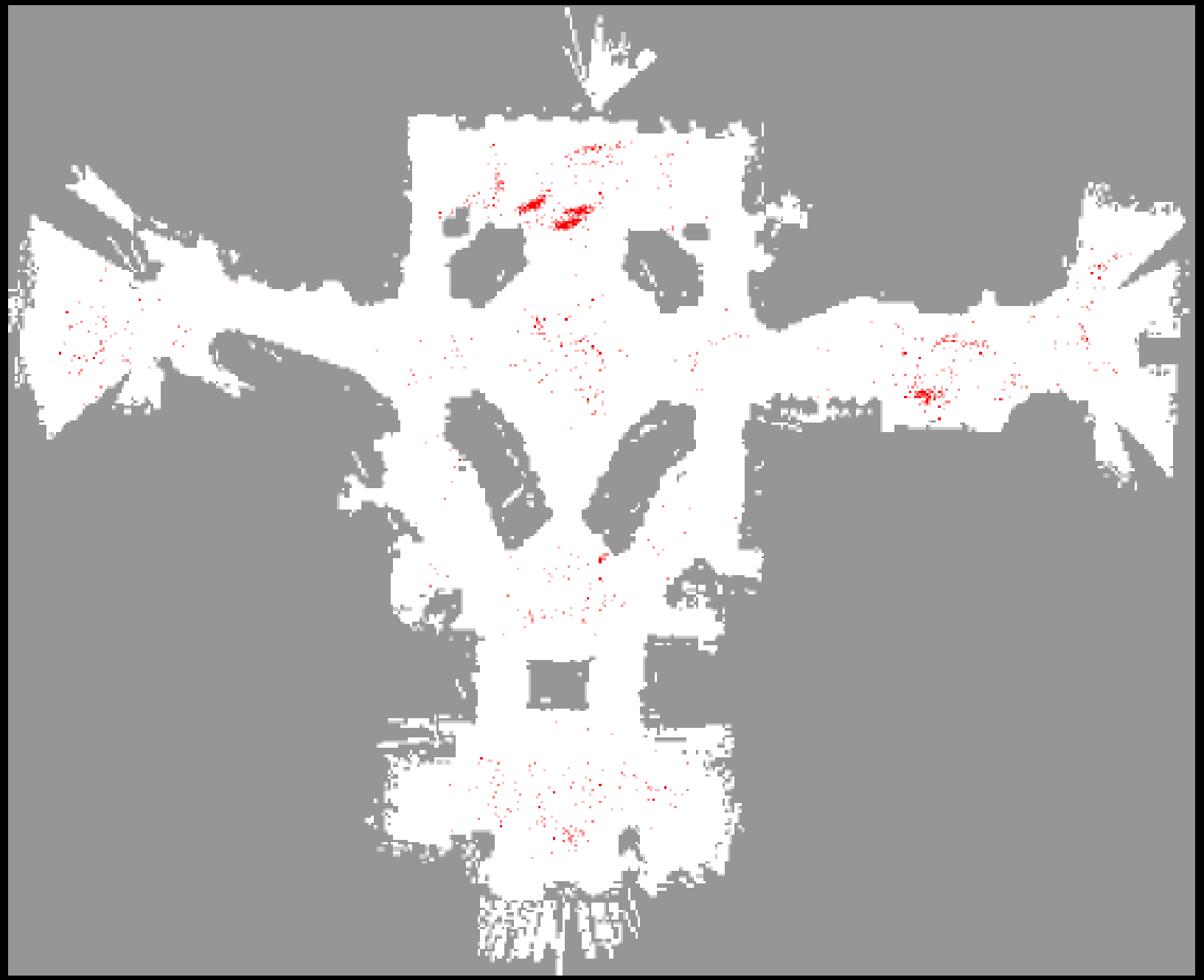


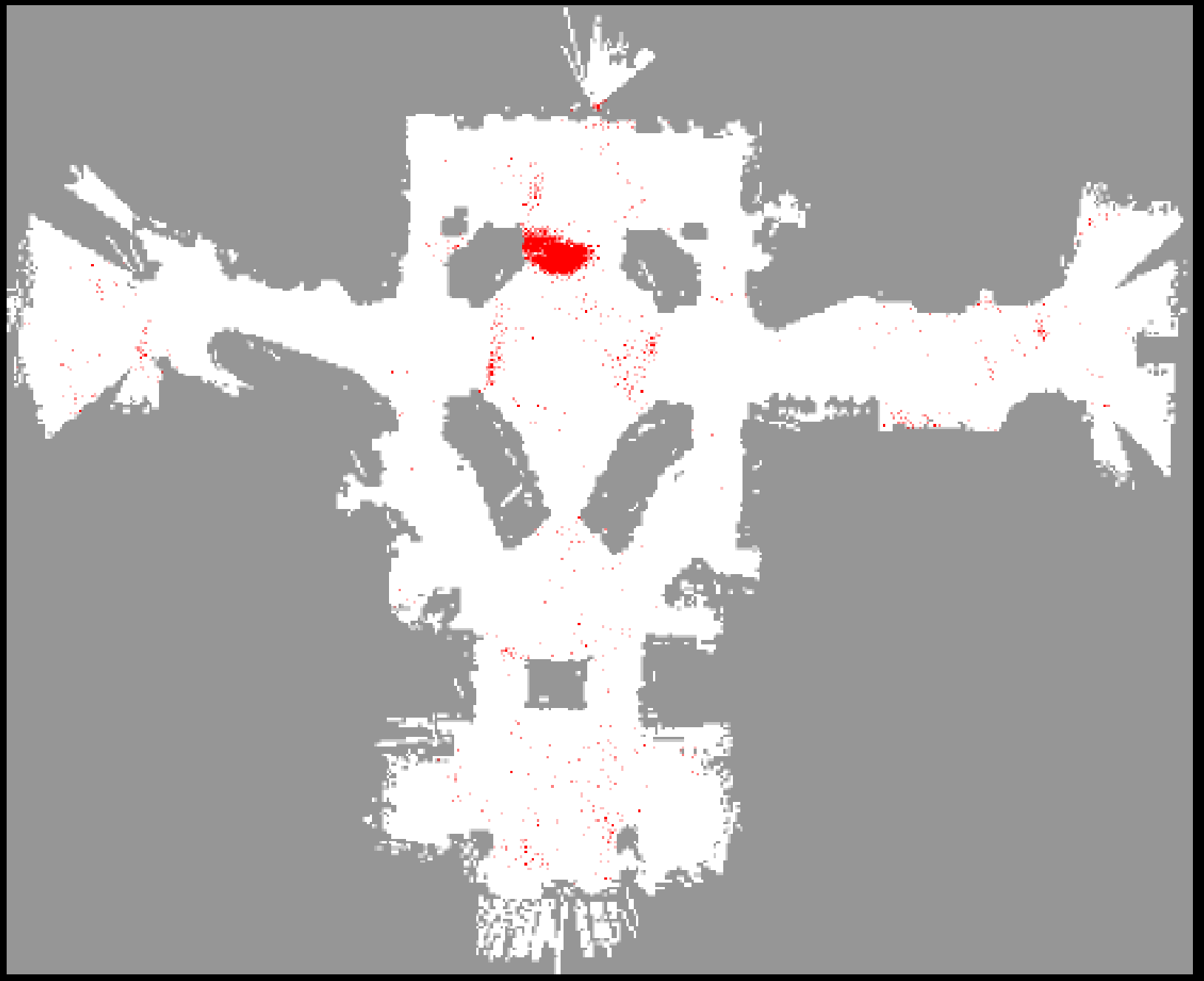
Next observation



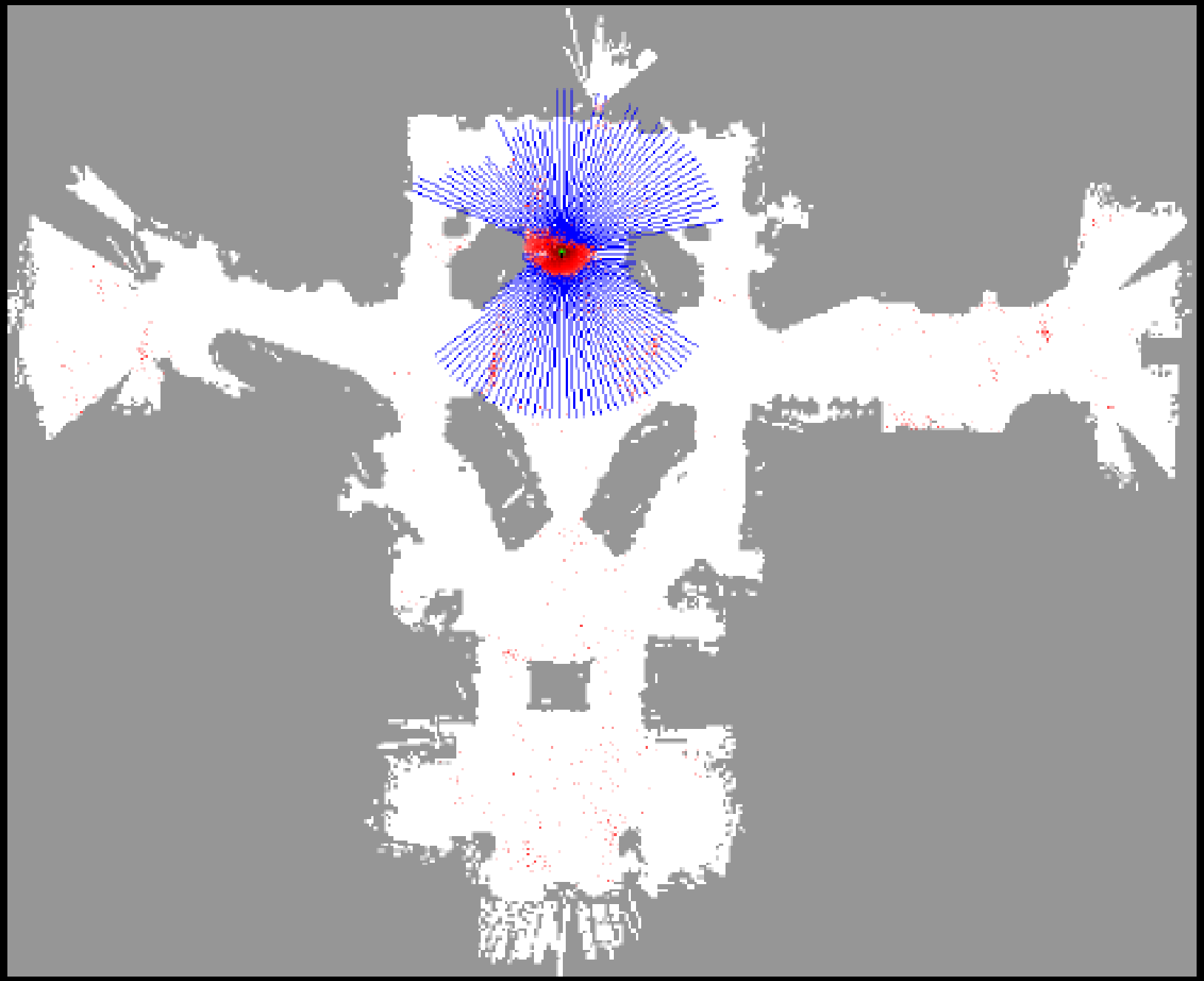


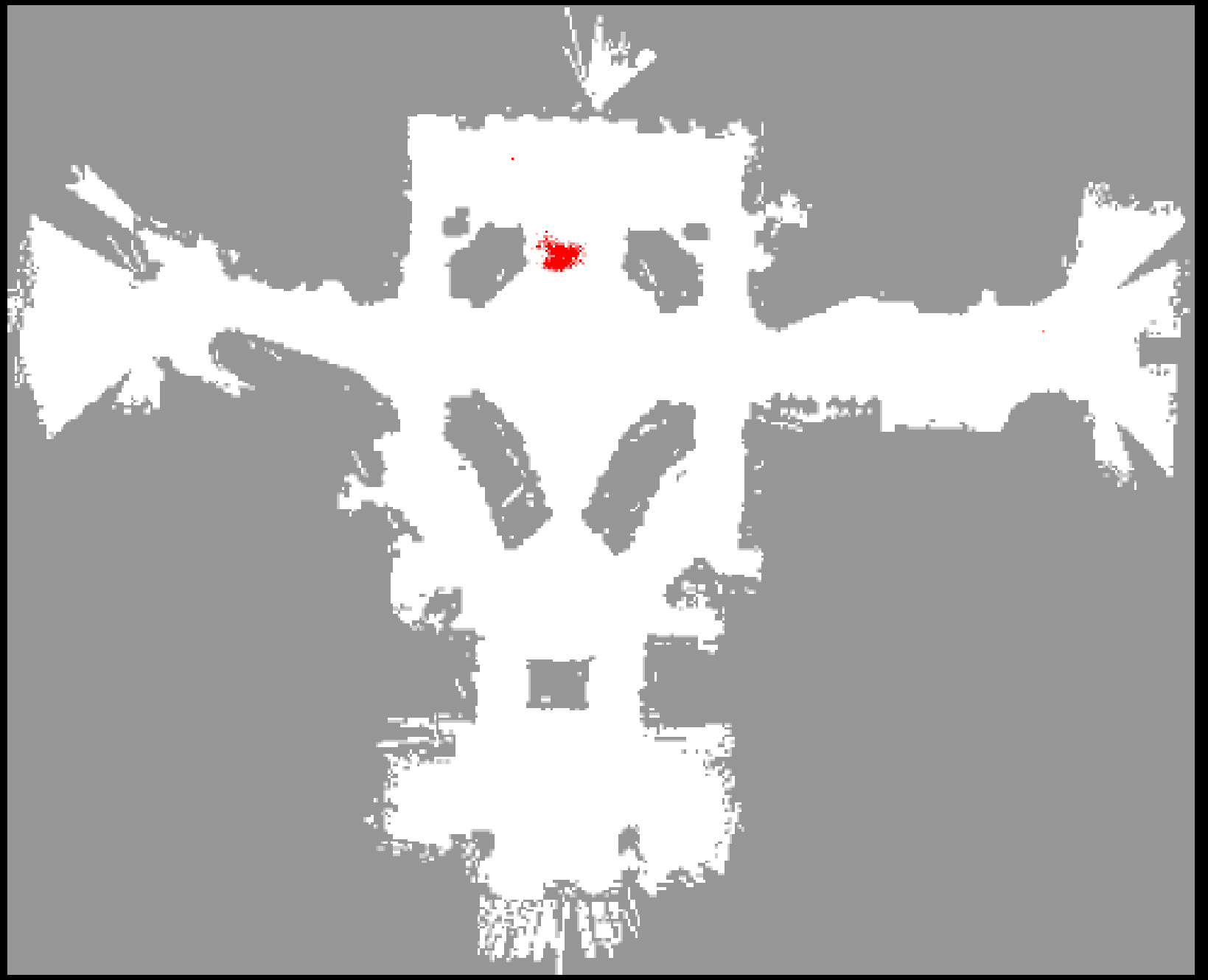
Next correction

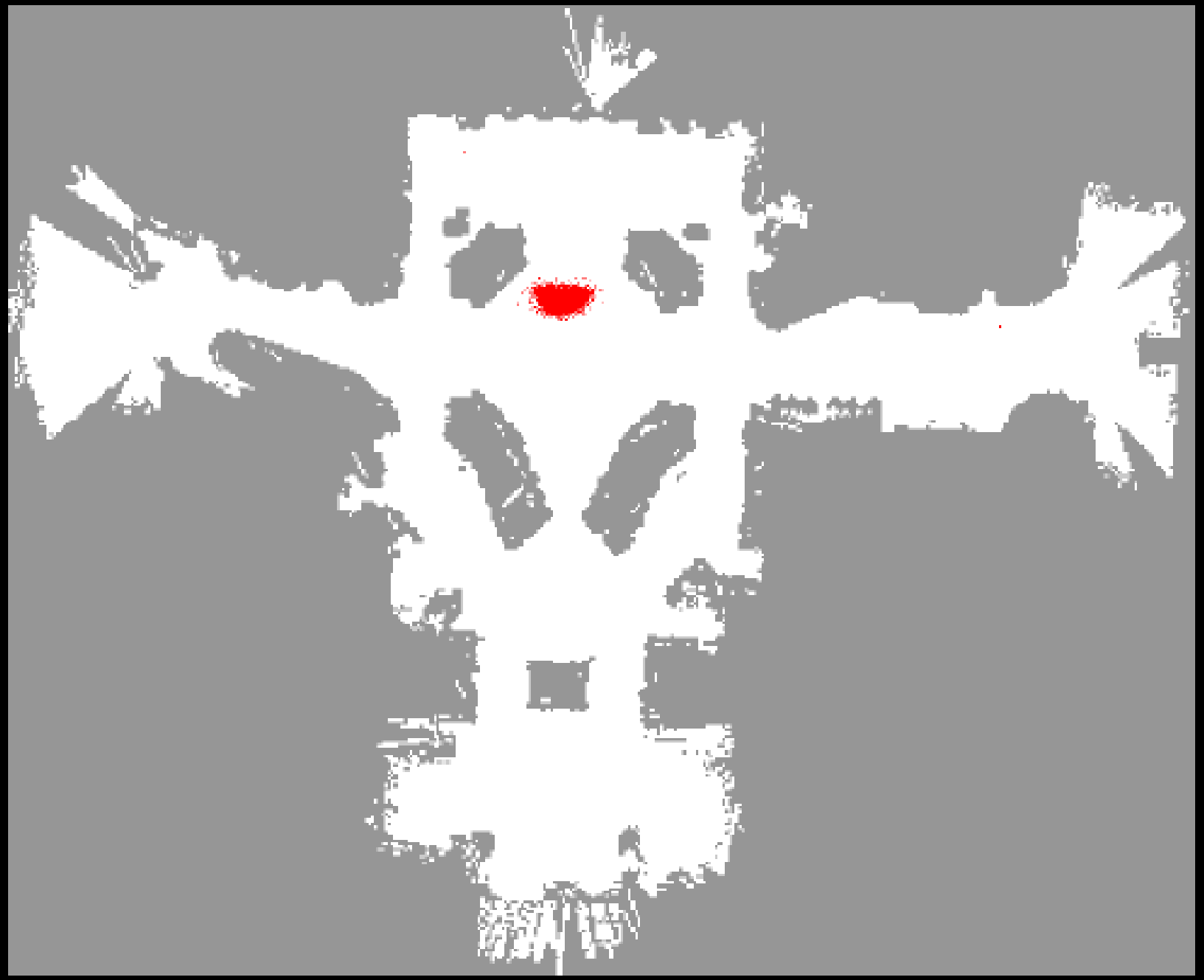


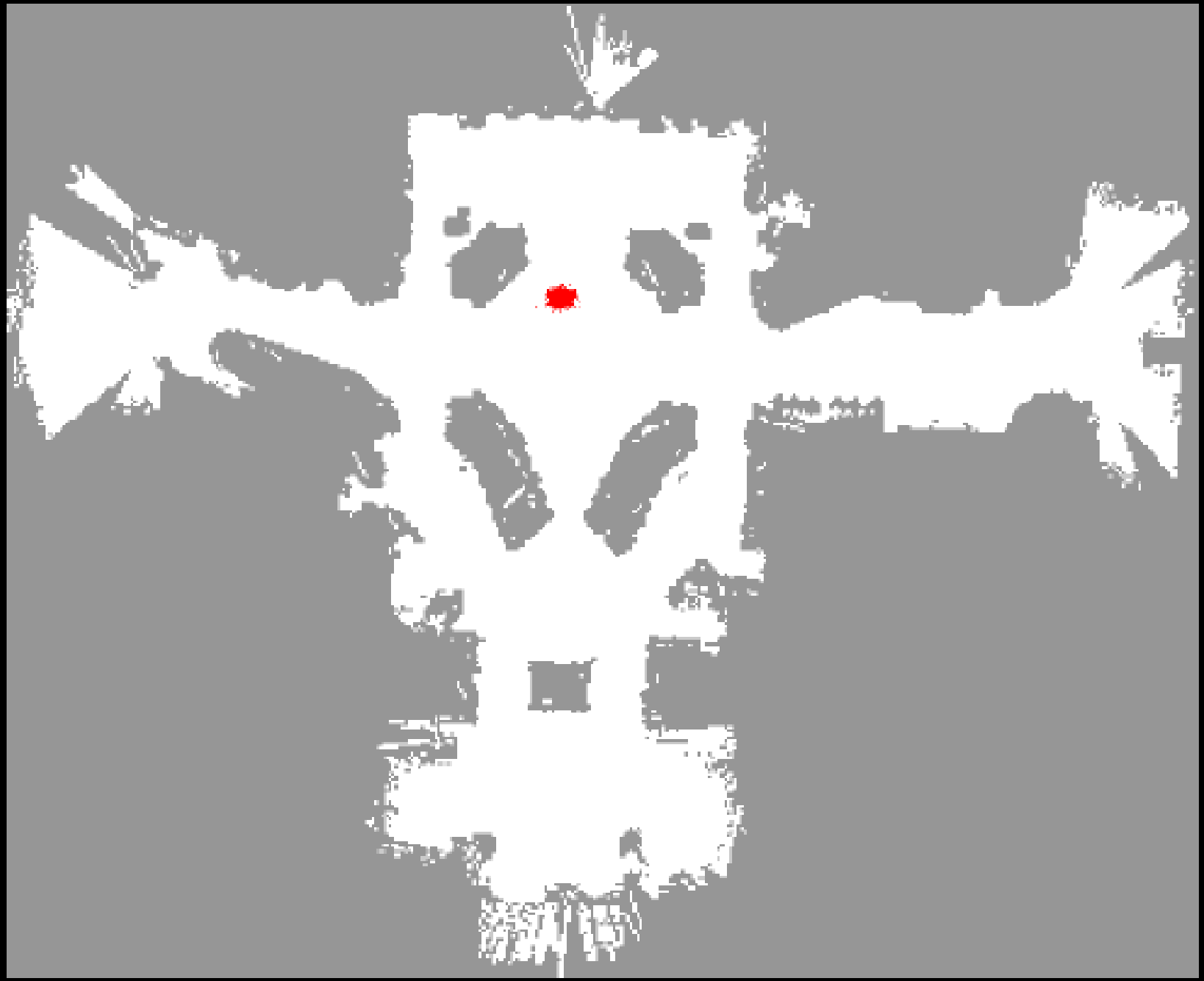


And so on...

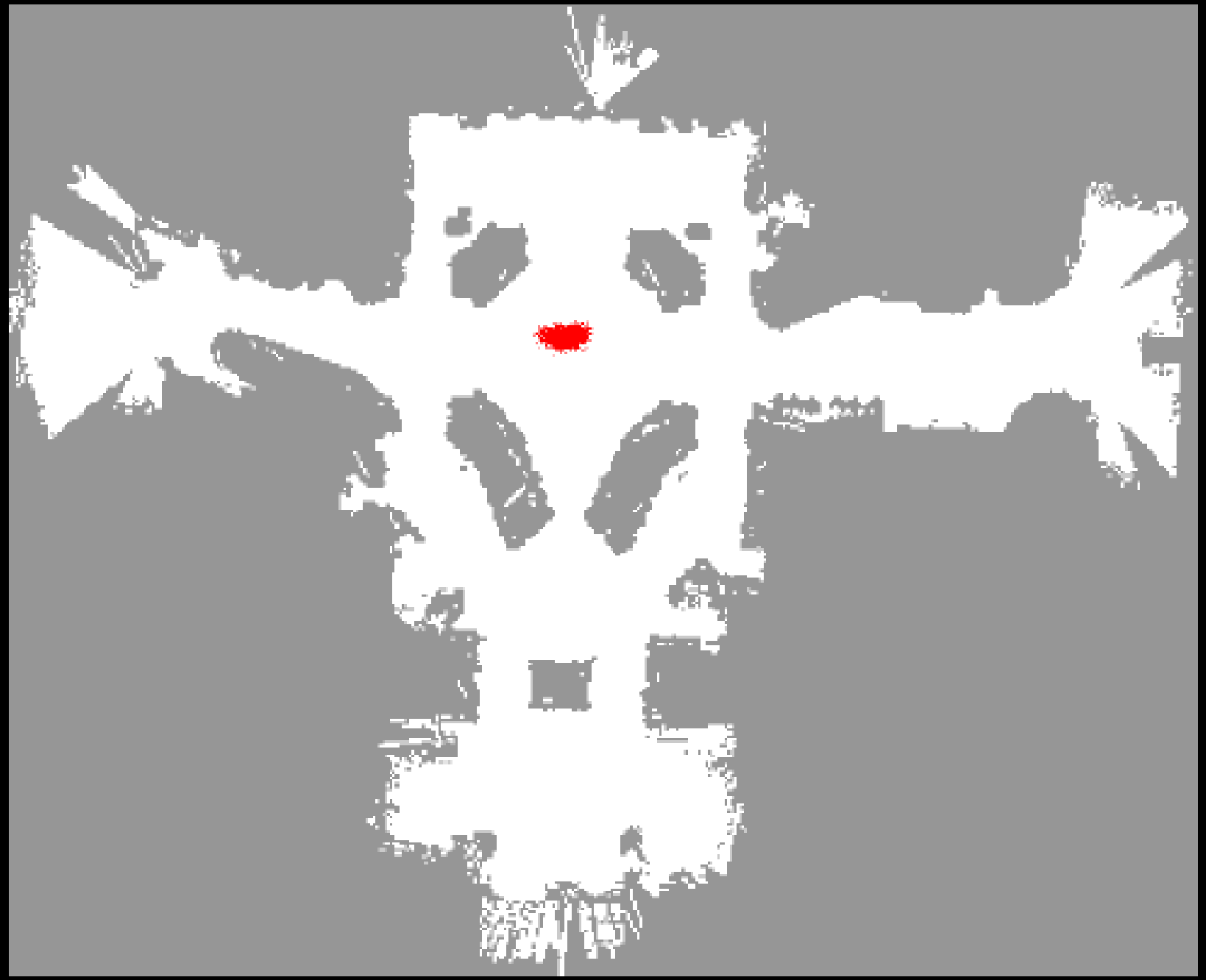








Converging
to one cloud



Particle Filter Algorithm(Part 1)

Setup Particles

Initialize Particles P with N particles

Set each particle in P with random values for location, velocity, and heading

Move Particles

For each particle p in P:

    Select a Random heading

    Select a velocity based on a Gaussian value from initial velocity

    Move each particle

Particle Filter Algorithm: (Part 2)

Measure and Assign weight to Particles
Take a measurement m
For each particle p in P:
    Assign error weight to p based on distance to m

Resample and Breed
Randomly select N particles from P based on weight (pick the closest particles)
Assign these particles to Particles P

Return Particles and Moment of Particles (Mean location of Particles)

Activity: Apply Particle Filter to Track an object in a live video stream

- What you Get: Python Modules (Available at: Problem Set 09)
  - `robot.py`: A Class modeling an individual robot object. This is the particle
  - `ParticleClassDemo.py`: An implementation of particles that only moves without resampling.
  - `particleDemo.py`: A module demonstrating implementation of camera and particles without resampling.
  - `Particles.py`: This is the class you will complete to implement the particle filter.
  - `ps09_ssdfollow.py`: You will complete this to test your SSD template matching.
  - `ps09.py`: You will complete this module to track the object using the particle filter.

Activity: What you will Do

- Run the `particleDemo.py` with your WebCam and make sure the camera and particles show on your Screen.
- Setup the “Playing Field” - a place for the ‘Toy’ to run around in and for you to track.
- Create a patch.png of the ‘Toy’ and place in the input folder of ps09.
- Complete `ps09_ssdfollow.py` and track the ‘Toy’ with an SSD template matching algorithm.
- Complete the code in `Particle.py` to implement the particle filter.
- Complete code in `ps09.py` to implement tracking using the Particle class.
- Use Snaggit or another Screen casting tool and record your Filter tracking the ‘Toy’. Narrate and explain your recording and publish to YouTube.