(As you work through this exercise, make note of parts you get stuck on and parts you find interesting. In addition to giving some exposure to Gen and probabilistic programming, this exercise is meant to help you begin thinking about how probabilistic programs play a role in common-sense AI)

(See bolded text for the actual TODOs)

Part 1 - Enumeration

Write code to enumerate over poses in the room, compute the log score under the generative model, normalize, and visualize (like the examples below showing the posterior as discovered by enumeration)

a. Pose is composed of a position (translation) and a rotation (orientation). Since we are considering a 2D SLAM setting, for the position we constrain the y value to always be 0 and the x and z to be within the room_width_bounds and room_height_bounds respectively. So the translational component will be [x, 0.0, z]. For the orientation we only have rotation about 1 axis (the axis pointing upwards of the agents head). This rotation can be constructed with R.RotY(angle) with angle the being the angle [0,2*pi] of rotation. This is why you see all the poses are Pose([x, 0.0, z], R.RotY(angle))

Show some of the generated visualizations which allow you to get a sense about the true posteriors.

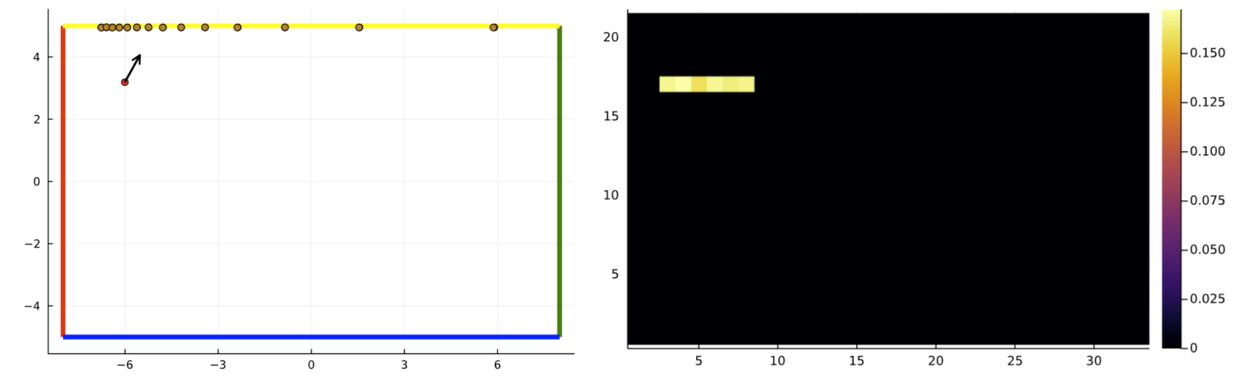
Vary the wall colors. Create some examples with:
wall_colors = [I.colorant"red",I.colorant"green",I.colorant"blue",I.colorant"yellow"]

which has different wall colors. And generate other examples with:
wall_colors = [I.colorant"red",I.colorant"red",I.colorant"red",I.colorant"red"]

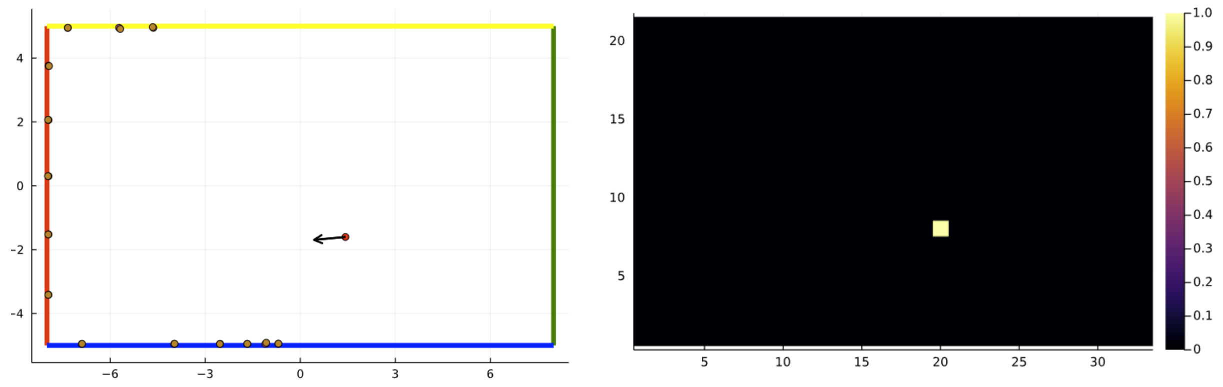
which has all the walls being the same color, which can result in more interesting posteriors due to the inherent ambiguity

See below for examples of the sort of visualization we are looking for. These types of visualizations are important in the development process of generative models and inference programs in Gen and in general, so it's good to get comfortable writing these tools.

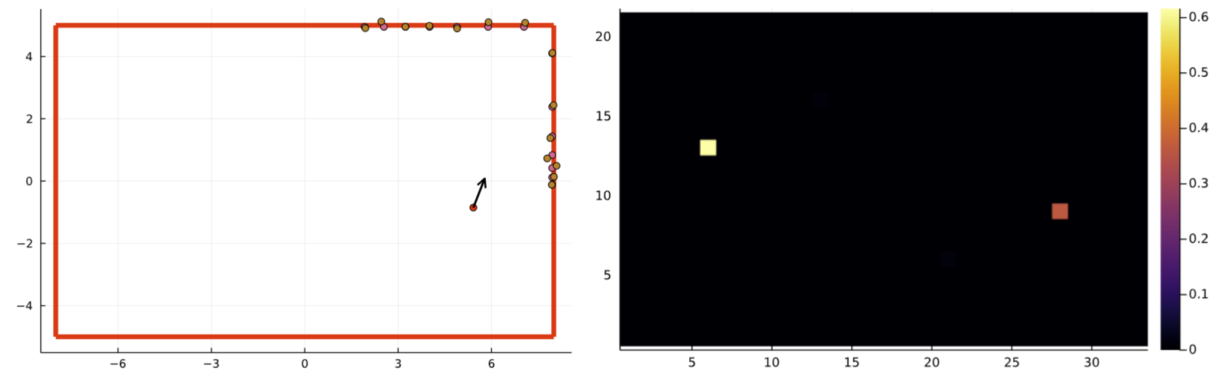
Write a couple sentences distinguishing SLAM done using a probabilistic model and Bayesian inference from a SLAM approach that simply uses optimization.



−6
−3
0
3
6
4
2
0
−2
−4
5
10
15
20
25
30
0.150
0.125
0.100
0.075
0.050
0.025



−6
−3
0
3
6
4
2
0
−2
−4
5
10
15
20
25
30
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1

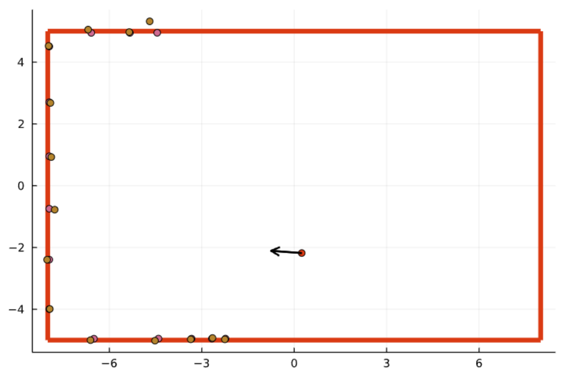


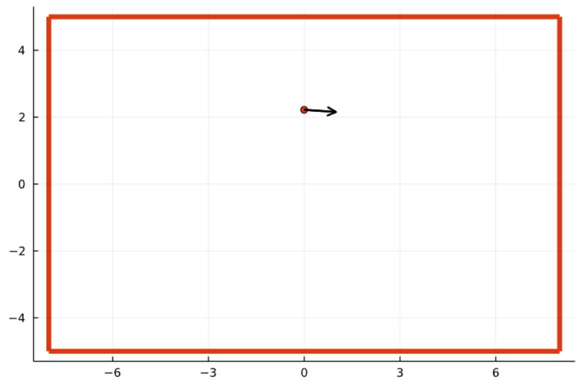
−6
−3
0
3
6
4
2
0
−2
−4
5
10
15
20
25
30
0.6
0.5
0.4
0.3
0.2
0.1

Part 2 - Naive Inference

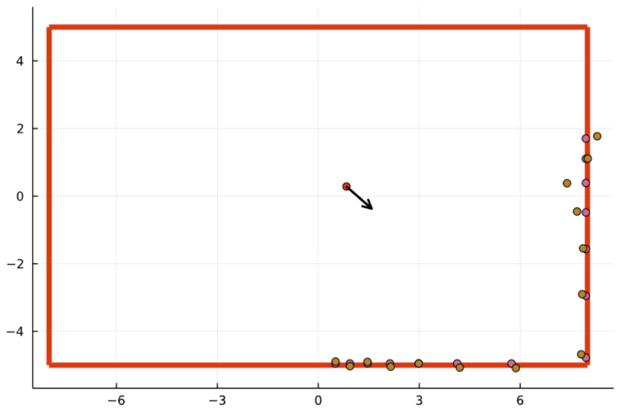
Write a basic inference program that uses the prior distribution on poses as a proposal distribution. Use `Gen.regenerate` to do this. Visualize the results (using provided visualization code snippets).

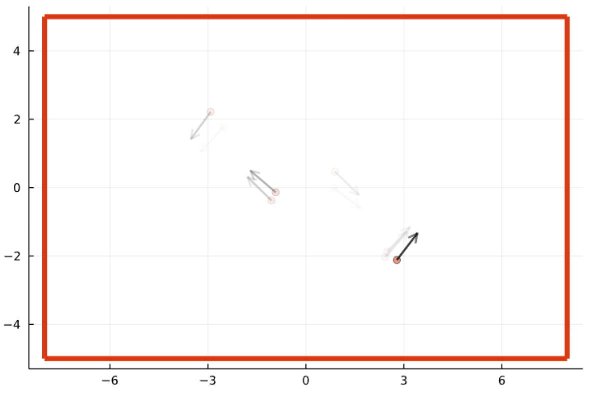
Look for cases where the inferred posteriors fail to capture the modes we know exist (from our enumeration investigation in the previous section). For example here we are showing the ground truth pose on the right.





And the inferred posterior on the left, which finds the second mode but fails to find the actual "true" mode. Or in this next example, we do find the different modes, but there weights are incorrect (because they are unequal).





Include examples of your own visualizations of these inferred posteriors using the naive inference algorithm.

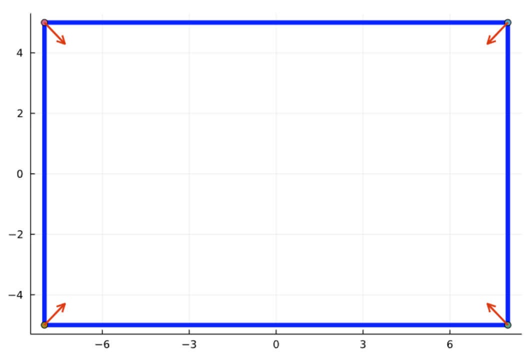
The inference algorithm we have developed in this section is slow and "wasteful" as it is simply using naive sampling from the prior, ignoring the data in the proposal. In the next section, we will look at more data-driven proposals.

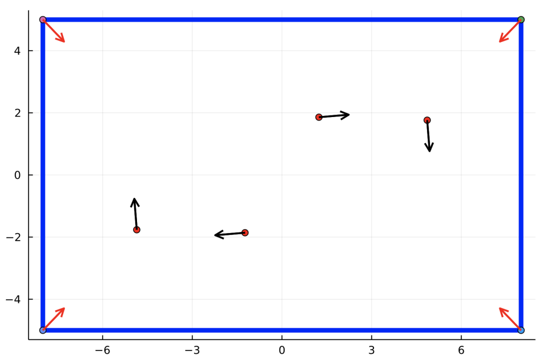
Part 3 - Corner detection

One way we can use the data to do faster and more accurate inference is by looking at features in our observations to help localize ourselves. One possible feature we could use to localize are corners

Here, we will write a simple corner detector. That will look at the observed depth data and find corners.

Then using these corners detected in the observed depth image and the 4 corners we know in the map, we can back calculate 4 possible Poses the agent could be at in the world.





Part 4 - PoseGaussian

In order to use our feature detector to do inference in the model, we need to build a custom proposal distribution that proposes near these 4 locations.

We will now write a custom distribution PoseGaussian that is a Gaussian distribution over the x,z coordinates of the agent's Pose and a Gaussian over the agent's head direction. You will need to implement the Gen.logpdf and Gen.random functions.

Visualize 1000 poses sampled from the PoseGaussian with different translation covariance and head direction variance parameters.

Part 5 - Custom Proposal

Combine part 3 and 4 + the mixture distribution given to you + Gen.metropolis_hastings to do MCMC with a custom proposal distribution. Show the results of the inference.

Part 6 - Parameter Experimentation

This part is a bit more open ended. One of the parameters of the model is the variance `var` in the depth likelihood. Varying is gonna change properties of the posteriors and of the inference. Experiment with versions of the slam model that have different values of this variance parameter. See how that affects the results of Part 1 and Part 2.

Part 7 - Next Steps

If you have extra time, think about other domains in which to apply these probabilistic programs e.g. vision, intuitive physics, etc. **In what ways do you imagine probabilistic programming approaches improving the robustness, data-efficiency, and generalizability of current methods (e.g. deep learning) in the fields of robotics and computer vision?**