Planning Algorithms in AI
PS1: Discrete Planning

Gonzalo Ferrer
Skoltech

1 November 2024

This problem set has three tasks and is individual work. You are encouraged to talk at the conceptual level with other students, discuss on the equations and even on results, but you may not show/share/copy any non-trivial code.

Submission Instructions

Your assignment must be submitted by 11:59pm on **November 14, Thursday**. You are to upload your assignment directly to Canvas as **two** attachments:

1. A `.tar.gz` or `.zip` file *containing a directory* named after your uniquename with the structure shown below.

   ```
   yourname_ps1.zip:
   yourname_ps1/
   yourname_ps1/any_code_file.py
   yourname_ps1/utils.py
   yourname_ps1/rod_solve.mp4
   ```

2. A PDF with the written portion of the problem set, reporting results, figures and answers to questions. Scanned versions of hand-written documents, note-books converted to PDFs, are perfectly acceptable (reduced size). No other formats (e.g., .doc) are acceptable. Your PDF file should adhere to the following naming convention: `yourname_ps1.pdf`.

Homework received after 11:59pm is considered late and will be penalized as per the course policy. The ultimate timestamp authority is the one assigned to your upload by Canvas. No exceptions to this policy will be made.

Task 1: Configuration Space (40 points)

For this task, you will write a Python script or note-book that generates the configuration space, including the obstacle regions and free space. The Workspace is a rectangular grid area (image) of 100×100, such that $\mathcal{W} = \{0, 1, \dots, 99\} \times \{0, 1, \dots, 99\} \subset \mathbb{Z}^{0+} \times \mathbb{Z}^{0+}$.

The robot is a "rod" object, whose configuration is $q = [x, y, \theta]$, taking into account that x, y are the image coordinates, therefore they are integer values in the grid and the orientation θ is also discretized into several values.

All the required data can be obtained from `data_ps1.npz`. This numpy data file consists of the environment map `['environment'](np.ndarray([100,100]))` an image with values 0 to indicate free

space and 1 to indicate an obstacle. The second component is the robot/agent, whose shape is included in `['rod'](np.ndarray([11,11,4]))`. There are `4` dimensions to represent the different discretization orientations - the robot rotated by 0, 45, 90, 135 degrees positions.

A (10 pts) Visualize from the given data the workspace and the different rod configurations for each discretized orientation. Comment on the given discretized values for orientation.

B (10 pts) Visualize the environment together with the object. For this, you may want to use the function `plot_joint_environment` from `utils.py` and select any valid configuration value for the rod.

C (10 pts) Create the C-space for the 2D environment map. For this, plot all the images corresponding to each of the orientations by using collision checking.

hint: you might want to look at the library `scipy.signal` and use function `signal.convolve2d(env_map, kernel, boundary='symm', mode='same')` to check for collisions.

hint: you may want to use `normalize_image(img)` from `utils.py` to normalize created space to $\{0, 1\}$, since after convolutions, values are not exactly 0 and 1 (this will be useful for task 2).

D (10 pts) Comment on the obtained C-space with the previous method. What is the size of the C-space?

Task 2: A star Algorithm (60 points)

For this task, you will implement a graph search algorithm. The actions allowed in this problem are moving up, down, left, right, rotate right, rotate left. In total 6 actions, each of them has an assigned a cost of 1.

A (40 pts) You need to implement the A star algorithm and plan in the generated discrete C-space from the previous task. The starting configuration of the agent is (6,6,2) and the goal configuration is (72,64,0). On this first iteration, use an heuristic function $h(q, q_G) = 0$, which is equivalent to the Dijkstra algorithm.

Save the result of calculated plan in `rod_solve.mp4` using `plotting_result(environment,rod,plan)` from `utils.py`, where `plan` is list of rod states from start to goal.

hint: Track the number of visited states to avoid/debug potential issues with internal loops.

B (10 pts) Change the heuristic function now to be $h(q, q_G) = L_1$ norm of the x, y components. Comment on the changes, how many states have been visited compared to Dijkstra? What is the final cost?

Comment on the results.

C (10 pts) Propose an heuristic function $h(q, q_G)$ that includes orientation. Compare metrics with the previous results. Comment on the results