5CCS2ITR (Introduction to Robotics)

Coursework 2: Due 27th November @ 4pm

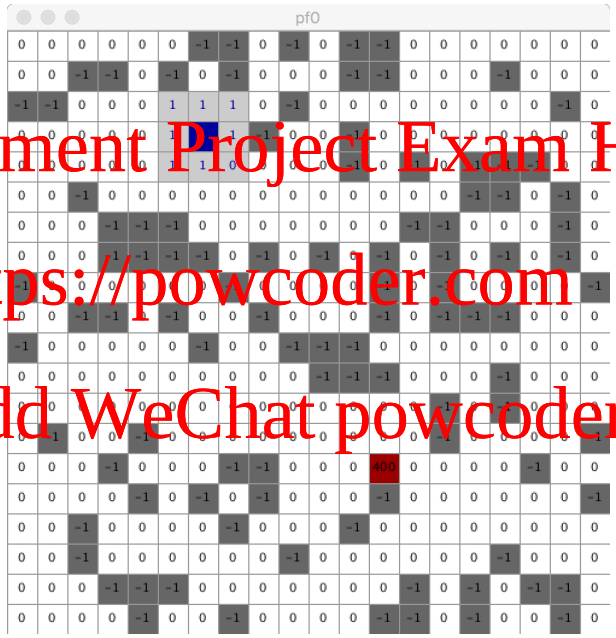
(version 1.0)

This coursework has two problems, both to be done in Processing. ⇒ Submission instructions and a marking rubric are on the last page.

1 Navigation using Potential Fields

FIRST: Download my starter code, **pf0.pde**, from the KEATS page.

- Start up **Processing**.
- Run it to make sure it works for you. You should get something like this:



Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

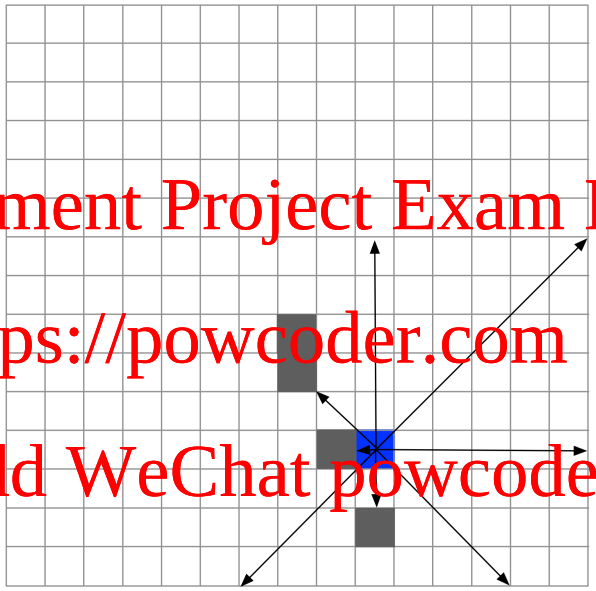
- The blue square ("cell") represents a mobile robot in a 20×20 grid world.
- The red cell represents the target location for the robot.
- The light grey cells around the robot represent its sensors. These are *range* sensors and should compute the distance from the robot to the nearest obstacle in each of the eight directions. The sensor values (integers) are indicated in blue.
- Non-sensed cells contain values (integers), which you will later fill in when you complete the assignment (as described below).
- The world is controlled using the keyboard. If you hit the space bar, then the world will reset.
- You can use the arrow keys to move the robot around: up arrow moves it up, left arrow moves it to the left, etc.

- When you reach the target, the "game" is over and the target cell turns magenta.
- You can quit the program by hitting **q** or **Q**.
- NOTE that my code isn't terribly robust—but it should be functional, as described above. Let me know if you find bugs!

YOUR JOB is as follows:

(A) Modify the **sense()** function so that it emulates range sensors with a range of 5 cells. Currently, the function only looks at the surrounding adjacent cells (i.e., cells that are ± 1 from the robot's cell). The sensors should return the distance to the nearest obstacle, or 6 if there is no obstacle within range. The obstacle can be an object in the world (indicated by dark grey cells) or the edge of the world.

Given the example grid world below:



Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

The corresponding range sensor values are:

2	6	6
1	–	6
4	2	4

In my Processing code, these are stored in an integer array, like this:

```
int sensors[8] = { 2, 6, 6, 1, 6, 4, 2, 4 };
```

Note that we don't store a value for the robot's location.

Currently, in my version of the code, the above scenario would render the following values:

```
int sensors[8] = { 0, 0, 0, 1, 0, 0, 0, 0 };
```

So your job is to modify the **sense()** function so that the sensor values are set using a range of 5 and producing values like the first example, above.

(B) Modify the code so that the robot's "memory" includes a **potential field** map of its world. Currently, in my version of the code, the robot's "memory" stores a map of the world in the variable `int world[][]`). This map currently contains fairly meaningless data:

– 0 if the corresponding cell is empty;

- -1 if the corresponding cell contains an obstacle; or
- a large number if the corresponding cell contains the target.

Your job is to modify the **initWorld()** function so that the `world[][]` data structure contains *potential field* values. (Remember Lecture 5 when we talked about potential fields.) Your robot will use its potential field to navigate in its environment. The idea is for the robot to *move up a gradient* towards the target location. Since this location is given a high value, the cells around it should also be given high values, though not as high as the target. The cells around those should be given a slightly lower value, and so on. Fill your robot's potential field map with values so that:

- Higher numbers are *attractors* and will pull the robot towards them.
- Lower numbers are *repulsors* and will push the robot away.

Set the edge of the robot's world and any cells containing obstacles to lower values.

(C) Create a function for the robot to navigate using its potential field, from its starting location to the target.

Currently, in my version of the code, the robot only moves according to the user's keyboard. Add another keyboard command, **N** or **n**, that you can press to get the robot to navigate autonomously (i.e., by itself) to the target location. Your robot should use its sensors and the potential field map to decide where to go next. Each time the user presses **N** or **n**, the robot should move ONE CELL towards its goal, using its potential field map. (This means that you will probably have to press **N** or **n** multiple times if the robot is more than one cell away from its target, but it will make it easier for you to debug your code and easier for us to mark you code, because we'll be able to see if the robot is behaving correctly.)

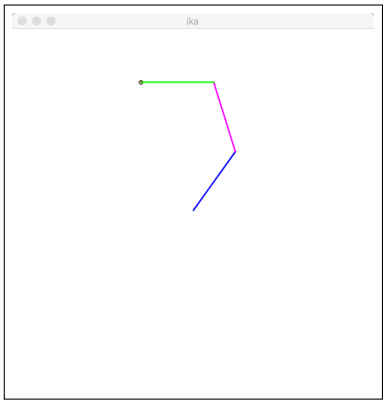
Assignment Project Exam Help
https://powcoder.com

2 Differential Kinematics

Add WeChat powcoder

FIRST: Download my starter code, **ika.pde**, from the KEATS page.

- Start up **Processing**.
- Run it to make sure it works for you. You should get something like this:



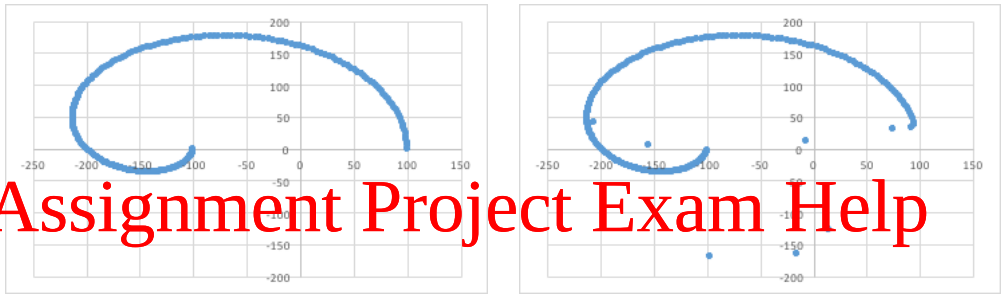
- The program shows a 3-link arm which is animated over 200 positions. Let it run until it stops rotating; then you can close the window.

- The code uses the **Analytical** approach to solving the inverse kinematics in order to determine the position of each link as the arm rotates.
- The data is generated in the **setup()** function, and notice that the data is also written to a file (called **ika-data.txt**, which will be saved in the same folder as your Processing sketch).

YOUR JOB is as follows:

(A) Create a new version of this program which uses the **Jacobian** approach to solving the inverse kinematics. Make sure that your version also saves the data generated, but make sure to save it in a different file, e.g., **ikj-data.txt**.

(B) Using your two data files (ika-data.txt and ikj-data.txt), generate scatter plots of the trajectories of the end effectors. Feel free to use Excel or any plotting program you prefer. Your results should look something like this:



Analytical Jacobean

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

(C) Compare the results generated by the two different solutions. Write an explanation for why you believe the results are different. Be thorough in your examination of the results. For example, you could:

- Compare the positions of the arm links (P1 and P2), in addition to the position of the end effector (P3).
- Compare the position of the benchmark end effector (see **generateBenchmarkData()** in my Processing code) with the positions computed by the Analytical and Jacobian methods.

I am expecting a response written in clear English. Your response should include some graphs (like those above). Your response should be no longer than ONE PAGE. Your response should be submitted in a PDF file. **ONLY PDF FORMAT WILL BE ACCEPTED.** We will not accept or open Word or other documents. ONLY PDF.

3 Submission Instructions and Rubric

This coursework assignment is worth **10%** of your mark for the module. It is due on **27th November no later than 4pm**. There will be a submission link on the KEATS page.

Put all your files in a **ZIP** archive and upload that to the KEATS page. Put the following files in your ZIP archive:

- a Processing program that contains the solution to Part 1 (Navigation using Potential Fields);
- a Processing program that contains the Jacobian solution to Part 2A (Differential Kinematics);
- a PDF file that contains your graphs (Part 2B) and explanation (Part 2C); and
- a README (PLAIN TEXT) file that contains an explanation of which file is answering which question.

You can also use the header comments in your code to indicate which program answers which question.

We reserve the right to deduct up to 25% of your mark on this assignment if you fail to follow the submission instructions above.

Assignment Project Exam Help

RUBRIC.

Your coursework will be marked out of 100 points, distributed as below.

https://powcoder.com

Add WeChat powcoder

1A. Modfify **sense()** function to use range=5.	10 points
1B. Generate potential field map and populate it.	20 points
1C. Add Navigate command to move robot using its potential field.	20 points
2A. Jacobian solution to Inverse Kinematics	10 points
2B. Graphs comparing your Analytical and Jacobian solutions	10 points
2C. Explanation about the differences between Analytical and Jacobian approaches	30 points
TOTAL	100 points