Coding Puzzle

The following exercise should take around 2-4 hours to complete. You may consult references such as Wikipedia but should not talk to anybody about this problem. This is on the honor system.

How to Answer

You should provide your solution in the form of a private or public GitHub repository. If the repository is private, please add the following GitHub users as collaborators:

amikhail0
pcooksey
cram9030

You should provide your build instructions, run instructions, and prose answers in plain text, Markdown formatted files.

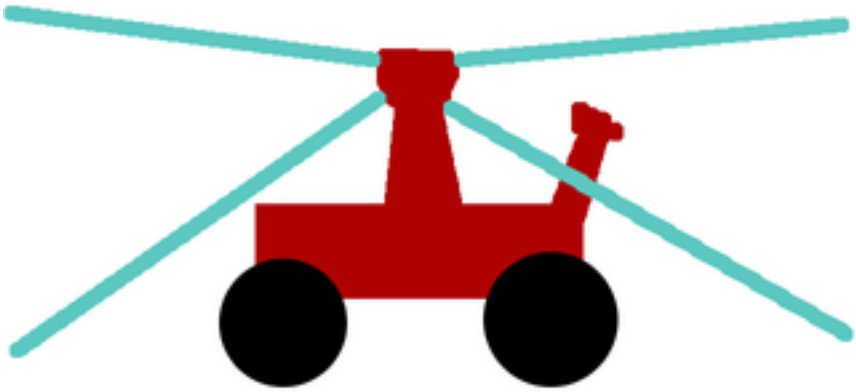
Your solution should include two parts: - code that solves the problem - prose discussion of the problem and your solution

If you have any questions about the exercise, just make an assumption, document it, and move on.

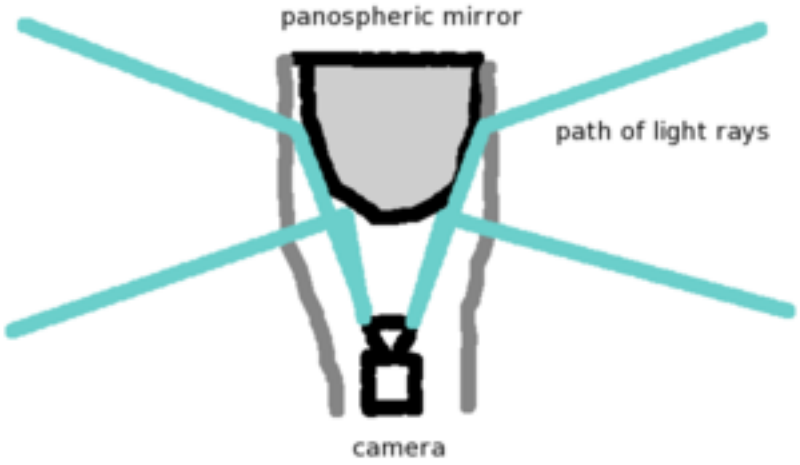
Problem statement

You will develop part of the processing pipeline for an object tracking system.

We have a mobile robot that needs to detect obstacles in its environment so it can avoid them. One of the robot's sensors is a panospheric camera mounted on top of a central mast that captures 360-degree panoramas of the robot's environment.



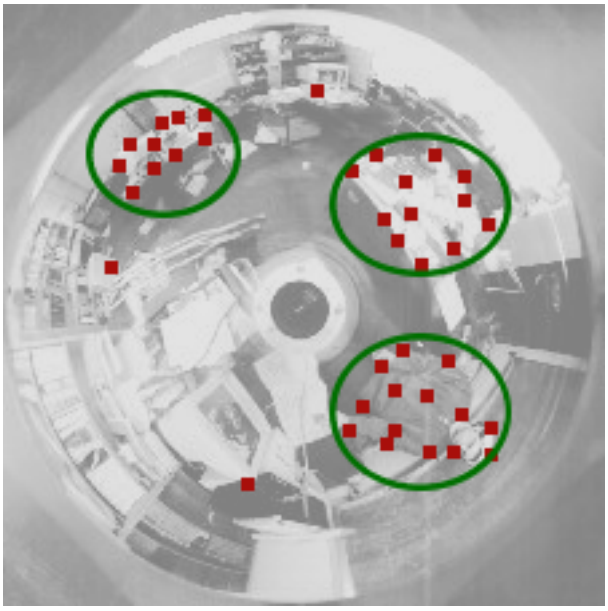
Panospheric camera mounted on robot



Panospheric camera side view

Example panospheric camera image

One of your co-workers has developed a feature tracker that finds the location of interesting visual features in the image, such as corners. Through experimentation, we have found that when you see a cluster of visual features together, that usually indicates the location of an obstacle, such as a big rock, or a dimwitted roboticist.

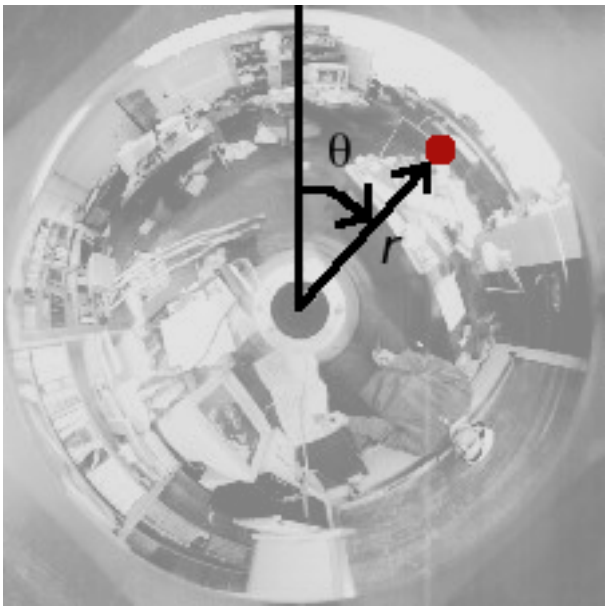


Red points are features, green circles are clusters

As shown in the sample image, there are also a few *outlier* features that are not part of a cluster. We want to ignore these outliers and only pay attention to clusters that contain several features.

Your goal is to write a module that inputs feature locations, finds clusters, and outputs the approximate center location of each cluster.

The coordinate system for image locations is polar coordinates (r, θ), where r is the distance of the feature from the center of the image, and θ is the relative bearing angle in degrees, which is 0 if the feature is straight up from the image center (straight ahead of the robot), 90 if it is to the right, and so on up to 360.



Polar coordinates

To keep this problem simple, we are *completely ignoring* the r value. You will only get the θ value of each feature, and you will only output the θ value of each cluster center. Your cluster center θ values only need to be accurate to within about 20 degrees.

Note: We have purposefully left out some details in this problem statement! We are looking for a quick solution that will allow some testing on the robot but might not be 100% robust to all possible inputs.

If you look closely at the test cases below, you'll see that you may be able to make some assumptions about the structure of the input data that allow you to take shortcuts in your solution. You are actively encouraged to make simplifying assumptions. You can get extra credit by explaining what they are and how they helped you.

Data Format

You will get feature θ values as a comma-separated list of floating-point values. Your solution should output the θ values of cluster centers in the same format.

Test Cases

We have provided 5 sets of sample input data. The first 3 have sample output as well. You can use the samples to test your code and demonstrate to us that it works.

Basic Questions

Answer all of these questions first.

1. Write your code in C++ to solve the problem stated above. Focus on the key problem of how to find clusters. Your code will be assessed according to these criteria, from most to least important:
    - Correctness on test cases
    - Simplicity
    - Maintainability: Modular, readable, "don't repeat yourself", testable, etc.
    - Performance: Don't improve performance at the cost of readability!
2. Write a paragraph discussing your solution. Talk about whatever seems interesting to you. For example: how it works, why it's a good approach, possible bugs you're concerned about.

Extra Credit Questions

If you have extra time, answer some of these questions and impress us. Pick any questions you like.

- Critique the problem statement and suggest ways it could be clarified in order to avoid surprises with the implementation. What assumptions did you make that helped you write your solution?
- Critique this obstacle avoidance approach from a robotics perspective. What problems would you expect to find with this approach and how would you mitigate them? (Obviously this is a bit of a toy problem and a cluster of visual features doesn't necessarily mean an obstacle is there in the real world, but you may have other interesting comments here.)
- What is the asymptotic time complexity of your algorithm? Could another algorithm have lower time complexity?