CS3105 AI Robotics Practical 1 Notes

Code considerations

Main code location:
PotentialFieldsRobot.java

This class Is the only one that that needs to be edited to incorporate the following effective functions:

1. A fractional progress function
   To measure fractional progress
2. Winding functions
   (i) A function to evaluate the winding if a move to sample point have been executed.
   (ii) A function to update the current winding.
3. An edited version of moveTowards
   a new line to be added to update the winding after each move.
4. A new version of evaluateSamplePointsArc with 2 main stages:
   1) go over the sample points and find three points:
      (i) the sample point that has the maximum fractional progress
      (ii) the sample that will wind the least
      (iii) the sample that will unwind the most
   2) select the appropriate point (from the 3 points):
      (i) if the robot is in Free Space
      (ii) if the robot is not in Free Space

   (The new version can be done within about 100 lines of relatively unsophisticated code.)

(Unwinding/Winding procedure:
If Unwinding is possible that does not increase the obstacle potential beyond a given threshold,
   then Unwinding is deemed to be viable and choose Unwinding that makes maximum fractional progress [or the most unwinding]
followed by
If only Winding is viable
   then choose Winding that makes maximum fractional progress [or the least winding]
followed by
If all the samples that unwind or wind are rejected [this should mostly be relatively rare]
   then the sample with the most fractional progress is to be selected.

You should take a minimalist approach to editing the code. There is no essential need to understand the other classes' code, except for knowing that there are some existing functions from the other classes that may be used. These functions can be found by considering those already used in “PotentialFieldsRobot.java" like the functions for computing the arc length and the curvature. (The latter are useful since
(signed) Turn = (signed) curvature * arc length.)

[Further arc tips:
Here is an example of how to use existing variables to get useful information such as arc length:
look for :
arcs.firstArc.arcLength
inside evalMoveArc () in the PotentialFieldRobot class.

If you wish to get the curvature, the class myArc already has a curvature attribute so you can get its value like this :
arcs.firstArc.curvature]

How to find where to insert code:
Take a look at the “PotentialFields.java" class to find that when the "Go Little Robot!” button is clicked, the function "ArcMove" or "move" will be executed repetitively until reaching the goal. Those two functions are in PotentialFieldsRobot.java. By looking at those functions, you can trace (see below) the code from this point to understand how it works. The "move" and "ArcMove" functions will call a number of functions that will get the sample points and evaluate them and then return the best move based on the potential.

The function "move" is for Euclidean distance
The function "Arcmove" is for arc distance

Trace:
1. Go to Potential fields.Java and look for goLittleRobot function
2. Note that ArcMove and Move are invoked.
3. Go to ArcMove and Move in PotentialFieldRobot.java and note that EvaluateSamplePointsArc is invoked.
4. Note useful functions for winding in PotentialFieldRobot.java such as curvature calculating function.
5. Use the above in writing a new version of EvaluateSamplePointsArc with winding/unwinding.

Also, a new button that would be useful is one for "Fractional Progress" to sit alongside “Arc” and “Euclidean” to allow your Fractional Progress planner to be demonstrated (for both Free Space and obstacle navigation). So take a look around for where the Arc and Euclidean buttons are made to use these as a basis for the new button.

Theory and Design

Always remember to build on success by varying one factor at a time.
Aim to *gradually* increase completeness.

What is fractional progress?
Fractional progress may be defined to be a decrease of the fraction *f/(p+f)* or equivalently an increase of the fraction *p/(p+f)* where *p* is the past up to the end of the candidate next transition and *f* is the estimated future.

There are 3 arcs that connect a path to the goal: a 1st arc going from the current robot position to the candidate sample point, and 2 remaining arcs from the sample point to the goal.

Suggestion for *p*: 1st arc length to candidate sample point.
Suggestion for *f*: total arc length of 2 remaining 2 arcs + obstacle potential.

Potential counter-example: *p* increasing without *f* decreasing as much, e.g. spiral. Test your definitions of *p* and *f* and your methodology conceptually or empirically against the counter-example to make sure they are likely to work.

Think about whether it is best to reset fractional progress to be measured from the current position at each stage or to keep it from the initial position.

How do Winding and Unwinding combine?
Suggestions:
1. Use a lowish obstacle potential threshold to determine when a sample point is *viable*, i.e. leads to sensible navigation, and when it should be rejected.

So when the threshold is breached by a sample point, reject the point as non-viable. This should stop the transitions taken from getting too close to the obstacle edge.

2. If there is at least one viable Unwinding sample point, then take the Unwinding point that makes maximum fractional progress. [Is your maximum optimal in some clear sense?]

3. If there is no viable Unwinding sample point, but there is a viable Winding point, then take the Winding point that makes maximum fractional progress. [Is your maximum optimal in some clear sense?]

4. If there are no viable sample points, take the one that makes the most fractional progress.

Be aware that how you Unwind can be optimised - your 1st version may need refining in some way for the trickier obstacles.

Path Deformation
There is no need to be worried about estimated future paths going through obstacle boundaries beyond any current boundary. The path can be deformed to bend round new boundaries as they are sensed and traversed.

Some (but not necessarily all) aspects to think about
1. A winding variable for how much winding has gone on in the past.
2. An update function for how much winding has gone on in the past.
3. When should the Unwinding be stopped?
4. What is a safe distance from the nearest obstacle to say the robot is in Free Space?
5. Should the radius of the forwards sample point semi-circle change as the goal is neared?

Where you should be:
If you are aiming to just do Part 1 (i.e. Free Space travel only) at this stage, all that would be required is to write and insert the fractional progress function. The main point of the latter overall is to make it ready for Part 2 where it supports the new potential function for the 2nd planner using winding and unwinding.

Also now see the CS3105 P1 End game notes in the CS3105/Practicals folder on StudRes.