

1 Introduction

One of the major problems with approaches like the *potential field* (exercise 6) is the handling of local minima. This can be overcome for instance by finding a way to construct a potential that is guaranteed to have only one (global) minimum. In this exercise we examine the *NF1* navigation function, a grid-based potential calculation that has this desirable property.

2 Wave Propagation

The basic idea of the NF1 is rather simple: Divide the environment of the robot into equally sized grid cells, mark all cells that are occupied by obstacles, and iteratively construct a monotonically increasing potential starting at the cell that contains the goal. In *M-File* pseudo-code, one possible way to implement the NF1 is:

```

grid = -2 * ones(dimx, dimy); % unvisited cells contain -2

for i = 1:n_obstacles
    grid(obstacle(i, 1), obstacle(i, 2)) = -1; % occupied cells contain -1
end

grid(goal(1), goal(2)) = 0; % goal cell is global minimum

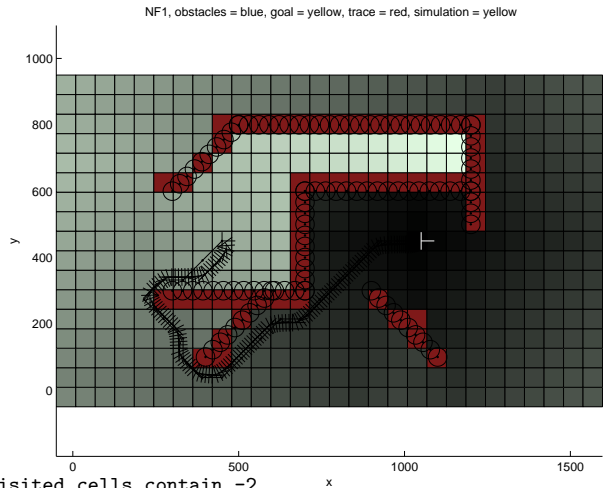
while we're not finished
    for all grid(i, j) that are on the wavefront
        propagate wavefront to the 4 direct neighbors
        by setting their value to grid(i, j) + 1

        taking care not to:
        * exceed the grid dimensions
        * propagate into obstacles
        * propagate into already visited cells

        if any such neighbors were found,
            then we're not finished
        end

        increment the wavefront
    end
end

```



Exercise 7.1: Implement Wave Propagation

Complete the file `calculate_nf1.m` such that it performs wavefront propagation. The code for inserting the obstacles is provided. Test your code using the `show_nf1.m` script.

3 Gradient Descent

Once the NF1 has been constructed, the potential can be descended from robot to goal. The following *M-File* pseudocode generates the list of grid indices that lead from the robot cell to the goal cell:

```

% define the neighborhood set
neighbor = [+1 +1 0 -1 -1 -1 0 +1;...
            0 +1 +1 +1 0 -1 -1 -1 ]';

```

```

% start the path at the robot position
path(1, :) = robot_cell;

% follow gradient until we're at the goal
n = 1;
while path(n, :) is not the goal

    for all candidate neighbors
        % HINT: candidate = path(n, :) + neighbor(i, :);

        check if the candidate is valid,
        and if it has the smallest value so far,
        in which case it should become the best candidate
    end

    n = n + 1;
    path(n, :) = best_candidate;
end

```

Exercise 7.2: Implement Gradient Descent

Complete the file `trace_nf1.m` such that it performs the gradient descent. The neighborhood is already defined in the parameter `nf1` and can be accessed using the form `nf1.neighbor`. Converting the resulting path to global coordinates is done as part of the already existing code. Test your solution using the `show_trace.m` script.

4 Robot Control, Comparison

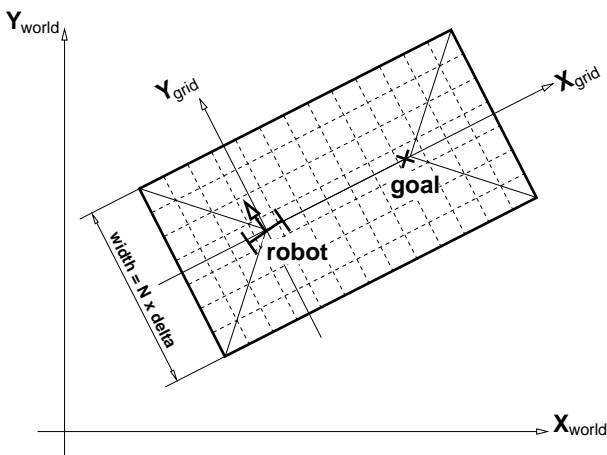
Exercise 7.3: Simulation

Complete the function `calculate_speed.m` and simulate the robot trajectory using `simulate.m`. Use the provided function `direction_nf1.m`, which calculates the gradient direction in the global frame of reference for any given point.

Exercise 7.4: Comparison with Potential Field

When compared with the previously studied potential field method, what advantages and drawbacks can you identify for the NF1? Are their respective strength and weaknesses complementary? Can you imagine ways of combining them into a more powerful approach?

5 Coordinate Transformations and Grid Indices



In order to be of use for controlling a real robot, the discrete grid representation of the NF1 has to be matched to meaningful coordinates and vice-versa. The *M-Files* used in this exercise perform this task for you. The file `create_transformer.m` initializes a data structure that can subsequently be used in the functions `from_transformer.m` and `to_transformer.m` to transform a given point either *from* the global coordinate system to a local one, or vice-versa. Once a point has been transformed from the global frame to the grid's frame of reference, a simple linear relation of the form $i_x = \text{round}((x + d_{off})/\Delta)$ can be used to find its corresponding indices. Finding a grid cell's coordinates in the global frame implies inverting that linear relation and then transforming it to the global frame. The files `flength_nf1.m` and `findex_nf1.m` implement the above mentioned linear equations. Examples of use can be seen in the files `create_nf1.m`, `calculate_nf1.m`, `trace_nf1.m` and `direction_nf1.m`.