Home Assignment #3
Fast marching method/Level Sets
Submission date: 8/1/2019

• For implementation exercises, hand in all relevant .m, .mex, .c, .cpp, .h files, and relevant images as .png/.gif files. If requested, submit also the results of distance calculation, as .mat files.

• Send the solutions by email to gautamppai89@gmail.com

• If you do not have a webcourse account, you can send the solution by e-mail.

• Solutions should be electronically submitted (doc / pdf / ps for the dry part).

• Solitary hand in - no couples.
Problem 1: MATLAB / C / C++ assignment

Note that a running MATLAB code will be checked. Your implementation is required to solve the maze in at most 1 hour (with ‘display’ argument set to ‘silent’). We highly recommend implementing the Fast Marching in C / C++, and linking it with MATLAB as a matlab executable (mex).

Implement the Fast Marching algorithm for weighted Euclidean domains.
Write a MATLAB (mex) function with the following matlab interface

\[
\text{function } [T] = fmm (F, \text{src, t0, display})
\]

accepting as the input an array of weights \( F \) (propagation “slowness”, inverse proportionally to propagation velocity), a vector of source point indices \( \text{src} \) and the corresponding vector of initial values \( \text{t0} \). The function solves the Eikonal equation

\[
\| \nabla T \| = F , \quad T(x_i) = T_i \quad \forall x_i \in X_0
\]

and returns the distance map \( T \).

Use an additional display argument to allow two display modes:

'silent' produces no output.

'iter' draws the current distance map \( T \) and the point attributes (Black, Green, Red) at each iteration.

Use \texttt{drawnow} to redraw the figure in a running process. Use \texttt{mexCallMATLAB} inside the C++ code to run matlab functions as part of your \texttt{mexFunction}. You can use STL data algorithms or downloaded modules to keep the list (priority queue) of alive (Red) points.
1. Calculate the distances inside the maze given as ’maze.mat’, using FMM and a source point at \((x,y) = (815,384)\). Submit the result in ’maze_dist.mat’.

2. Find the shortest path from inside the maze, at \((815, 384)\), to the point \((9, 234)\), outside of it. Plot the shortest path on the top of the maze image.

3. Now, rotate the maze by 45°, and re-calculate the distances from the same (transformed) point inside the rotated maze. Compare the distances calculated for the two versions of the maze. Submit the result in ’maze45_dist.mat’.

4. The optical path length (OPL) is defined as the product of the Euclidean length of the path of light, and the index of refraction of the medium through which it propagates (denoted by \(n\))

\[
OPL = \int_C n(s)ds
\]

Use FMM to calculate the OPL between two points, one inside a pool with water, with coordinate \((x, y) = (400, 500)\) and one outside,
Figure 2: Indices of refraction given in the ‘pool.mat’.

with coordinate \((x, y) = (1, 1)\), using the indices of refraction given in ‘pool.mat’. Plot the shortest path on the top of the image of the indices of refraction.

Solve the maze given as ‘maze.mat’, using FMM, finding the shortest path from inside the maze, at \((815, 384)\), to the point \((9, 234)\), outside of it.
Problem 2

Create a closed curve as shown below. Implement the flow $C_t = \vec{N}$ using level sets, such that the curve shrinks. Show the evolution at different time steps. At each time step, calculate the curvature of the curve and plot it alongside. Your report should contain a sequence of pair of plots. Each row corresponds to a time. The first column should show the curve evolved to that time and the second column should be the plot $\kappa$ vs arc-length $s$ for that curve. Comment on what you see.

Figure 3: