Puzzle Programs

This document contains two sections: Generator and Noisy, which correspond to the folders in the solving algorithm folder. Although there are two additional folders called Egg and Synthetic, each contains nearly the same code with different smoothing parameters, interference thresholds, segmentation thresholds, etc. Each section is organized into main and called programs. The main program is the program that the user must use to generate or solve a puzzle. The called programs are called in the main program, so the user will never directly run these programs. If a program is italicized, then either it was found online or Rob's Macalester students wrote it.

I. Generator

A. Main Program

PuzzleGenerator.m	
Input	None. The user will be asked a series of questions, where the user must enter the number of pieces, edge style, and puzzle shape
Function	Generates the puzzle
Output	Puzzle with discretized edges in the form of a cell array. This output may be inputted into NoisySolver.m or Synthetic Solver.

B. Called Programs

voronoisphere.m	Generates a spherical Voronoi diagram.
CurvyEdge.m	Generates a curvy shaped puzzle edge given four points from the Voronoi diagram. The output is a discretized Bezier curve that forms the puzzle edge
JigsawEdge.m	Generates a curvy shaped puzzle edge given four points from the Voronoi diagram. The output is a discretized Bezier curve that forms the puzzle edge

II. Noisy

A. Main Program

NoisySolver.m	
Input	Synthetically generated puzzle from the program PuzzleGenerator.m. After generating a synthetic puzzle, you can solve the puzzle by entering: NoisySolver(puzzle)
Function	Scrambles the puzzle, detects matching edges, and assembles matching edges together
Output	Fully assembled puzzle, which is in the form of a cell array

B. Called Programs

scrambler3d.m	Scrambles the pieces by performing a rigid transformation to each puzzle piece.
KappaSegmentation.m	Partitions the signature wrt the kappa axis.
compareEgg3DEXP.m	Calculates the similarity score between two puzzle edges' signatures.
PutTogetherAdvanced2.m	Assembles two edges together and determines whether the edges are outdented or indented
AssemblyRevise.m	Assemblies the matching edges using the seed methodology. A piece is selected as the seed and all of its matching edges are assembled to it. Sequentially, the seeds of the original piece will become the new seed. The seed methodology is an effort to avoid sub puzzle assembly
UpdateSolution.m	Called directly after AssemblyRevise.m in order to detect inadvertent matches
PutTogetherAdvanced.m	Assembles matching edges together

C. Called by Called Programs

compsig.m	Computes signature curve of each puzzle piece
estimateRigidTransform.m	Computes rigid transformation matrix to assemble two matching edges together
HausdorffDist.m	Calculates the Hausdorff distance between two assembled edges in order to detect false positive matches
InterferenceDetection.m	Calculates the surface area of overlap and space between two assembled edges
Noise.m	Adds noise to each puzzle piece
RotatePiece.m	Assembles two matching edges together by using the rigid transformation matrix from estimateRigidTransform.m
Smoother.m	Smooths a noisy puzzle piece
SmootherSegments.m	Smooths each puzzle edge and signature, which insures that each edge has a uniform number of points. Note: the degree of smoothing is very minimal