Analysis of Three-Dimensional Shapes
F. R. Schmidt, M. Vestner, Z. Lähner
Summer Semester 2016

Computer Vision Group
Institut für Informatik
Technische Universität München

Weekly Exercises 7

Room: 02.09.023

Wed, 15.06.2016, 14:00-16:00

Submission deadline: Tue, 14.06.2016, 23:59 to laehner@in.tum.de

Mathematics: Euler Characteristic

Exercise 1 (2 points)**.** Show that on a triangle mesh with negative geodesic curvature everywhere, the shortest path between two points is unique.

Exercise 2 (2 points)**.**
1. Calculate the number of vertices and edges for a closed triangle mesh with f faces and Euler characteristic χ.
2. Extend the Euler formula for triangle meshes to 4-sided polygon meshes.

Programming: Geodesics

Download the supplementary material 07 from the homepage. It contains a MinHeap class file, `incidenceMatrix.m`, `sheet7check.m`, `dijkstra.m` and `fastmarching.m` with the basic constructions of the algorithms (and some other things to let you check your solutions). If you do not know how a Heap works, I suggest you make yourself familiar with it. Cheat sheet for the functionality:

1. `MinHeap = MinHeap(int)` creates a new instance of the Heap with maximum size n (and maximum key n)
2. `bool = MinHeap.decrease(key, value)` decreases the value of the given key, nothing happens if the key does not exist in the heap or the value is larger than the previous value, the return value indicates if the heap was changed during the operation
3. `bool = MinHeap.increase(key, value)` like decrease basically
4. `double = MinHeap.peakKey(key)` returns the value of the given key, -1 if the key is invalid
5. `[value, key] = MinHeap.pop()` returns key and minimum value in the heap. $(-1, -1)$ if the heap is empty
6. `MinHeap.push(value, key)` adds a new (key, value) tupel to the heap. Nothing happens if the key already exists in the heap
7. `bool = MinHeap.isEmpty()` returns true if the heap is empty

You can check if your calculations are correct by running the scripts in `sheet7check.m`.

Exercise 3 (3 points)**.** Implement the geodesic distance and path between two points a, b on a manifold with the dijkstra algorithm introduced in the lecture. The file `dijkstra.m` already contains the cornerstones such that you only have fill in the dijkstra step and extract the solution. Additionally to the distance, we also want to get the shortest path. For that we simply need to keep track of the predecessor of every vertex that resulted in an update of the minimal distance there. In the end, we need to backtrack through the predecessors from b until we reach a.

Exercise 4 (5 points)**.** Implement the distance map of one point of a manifold using the fast marching algorithm from the lecture. `march.m` is meant to include the update step which is the only real difference to dijkstra.An assumption in the lecture was that the triangle is aligned to the x-y-plane, the code in `march.m` already includes this alignment. If you are getting confused, use the B, R, G sets as in the lecture but this is in general not necessary and slows down the code. You do not have to implement the pre-processing step eliminating obtuse triangles.