Programming Assignment 1

Cure Reconstruction using matchings

January 17, 2006

An instance of the curve reconstruction problem is a finite sample $V = \{p_0, p_1, \ldots, p_{N-1}\}$ of an unknown curve γ. The task is to connect the points in V in the order in which they lie on γ. In this programming assignment we will generate the connections in the samples using a modification of Gale-Shapely. Figure 2 shows an example output of a curve reconstruction algorithm. For the purposes of this homework, we will assume that the curve is closed and smooth. Hence the reconstruction graph that is given as output by the algorithm should have degree 2 at each vertex.

The Algorithm: The first step in our curve reconstruction algorithm is to connect every point $p \in V$ to its nearest neighbor $q = nn(p) \in V$. These edges are already computed using the function `compute_nn_edges()` in the `main.cpp` file and drawn in the output window using white color. The results of this function are stored in an array $G1$. After the execution of this function, Point p_i has its nearest neighbor $p_j, j \neq i$ such that $G1[i] = j$. As you can see by playing with the program on `linprog`, this does not complete the curve (and leaves a lot of connections, yet to be made). The second step of the algorithm is what you have to implement (`compute_matching(void)`). This function has to compute the array G2 and draw all the resulting edges $(i, G2[i])$ in red using the function `draw_segment()`. An example of drawing a red edge is already implemented in the function `compute_matching(void)`.

In this assignment you will be implementing the `compute_matching(void)` function. This function should work in the following way. First the function computes all the vertices which have only one edge incident on it (after the nearest nighbor matching of points with other points). We will call these vertices *single and looking*, in short an sl-vertex. Similarly, we can define the notion of an sl-edge which connects two sl-vertices. Let weight w_{ij} be the weight of an sl-edge connecting p_i, p_j. The function that you will implement, first computes these weights, then every sl-vertex is paired up with another sl-vertex using Gale-Shapely. Note that in this case, for some configuration of points, the weights might lead to a ranking that is same for two sl-vertices. We will ignore this for now and still run Gale-Shapely on the rankings.

The only thing left to specify is how to compute the weights w_{ij} for two sl-vertices. Let $G = (V, E_{nn})$ be a graph such that $E_{nn} = (i, G2[i])$ for all

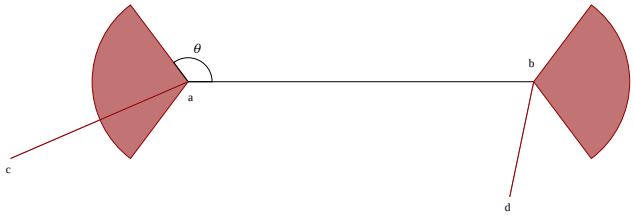

Figure 1: An example of a non-admissible edge. Let us assume that we want to compute w_{ab}. (a, b) is an admissible edge if the two edges incident on the sl-vertices a, b make a large angle on them (You should experiment with different values of θ in your program. For beginners you can set $\theta = \pi/2$). In this example, the edge (a, b) is not admissible, since the angle made by the edge (b, d) is small with (a, b). For a given θ, the cone of angles allowed is colored in brown.

$i = 0..N - 1$. We first will need the definition of an *admissible* edge (See figure 1). An admissible edge (a,b) is an edge which makes an angle $> \theta$ with the two edges incident on a and b. Now the weights are defined by the following formula:

$$w(i,j) = \begin{cases} p_i.\texttt{sqr\_dist}(p_j) & \text{if } (i,j) \text{ is admissible;} \\ \infty & \text{otherwise.} \end{cases} \tag{1}$$

Once the weights are computed for all sl-edges, you have to match all the sl-vertices and draw the bipartite matching in red.

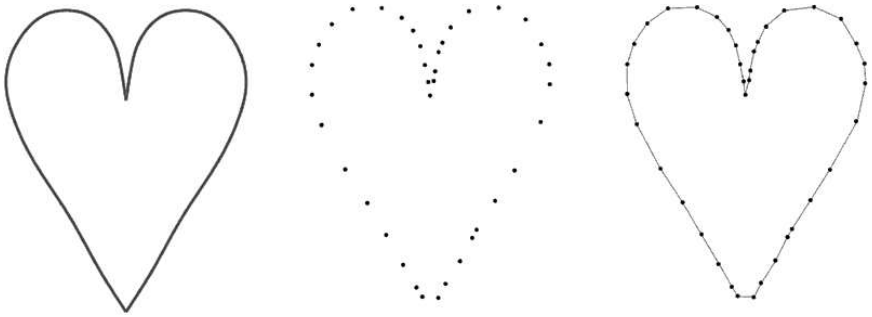

Figure 2: An example output of a curve reconstruction algorithm. (Left): A curve that is not given to the program as input. (Middle): A sampling of the curve that is input to the program. (Right): Output of one of the known curve reconstruction algorithms. Note that the sample points adjacent to each other on the original curve are the only ones connected in the reconstructed curve.