**MECH/MESF5520 Programming Projects**

**Programming Lab 1 – curves**

Using any Computer Graphics programming language such as MATLAB, write a simple program that has the following functions:

(1) The user can specify the coordinates of a list of 3D points, using BOTH the following ways:

(a) Keyboard input; AND

(b) Read from a text file (see the Note below).

(2) The program constructs the Bezier curve using the points got from (1) as the control polygon and displays it on the screen.

(3) The program should be able to save the control points of the curve into a text file which can then be input again into the program (i.e., the (b) of (1)).

Note: eventually for the other projects you will be asked to define a Bezier curve input from a text file (i.e., (b) of (1)), so, it is suggested you do it now; the format of the text file must be this:

Line 1 m /\* the number of Bezier curves \*/

Line 2 n1 /\* the number of control points of the first Bezier curve \*/

Line 3 Either 0 or 1 /\*0 means the curve is closed, 1 means it is open \*/

Line 4 the XYZ coordinates of the 1st control point /\*separated by space or “,” \*/

Line 5 the XYZ coordinates of the 2nd control point

.

.

.

Line n1+3 the XYZ coordinates of the n1th control point

Repeat for the next m-1 Bezier curves.

Example:

2

4

1

1.5 0.4 -3

1.7 -2.56 4.78

0 245.6 10

0 0 0

3

0

3 5 -6

-4 3.5 7

1.5 -5 -10.5

**Programming Lab 2 – surfaces**

Based on your programming lab of curves, you are required in this lab to define and display Coons patches with different (*u*) and (*v*) blending functions. You program should have the following functionalities.

(1) The user can define the four boundary curves of the Coons patch. The format of the curves is Bezier. For your convenience, you can assume that the number of control points of any Bezier curve is 6.

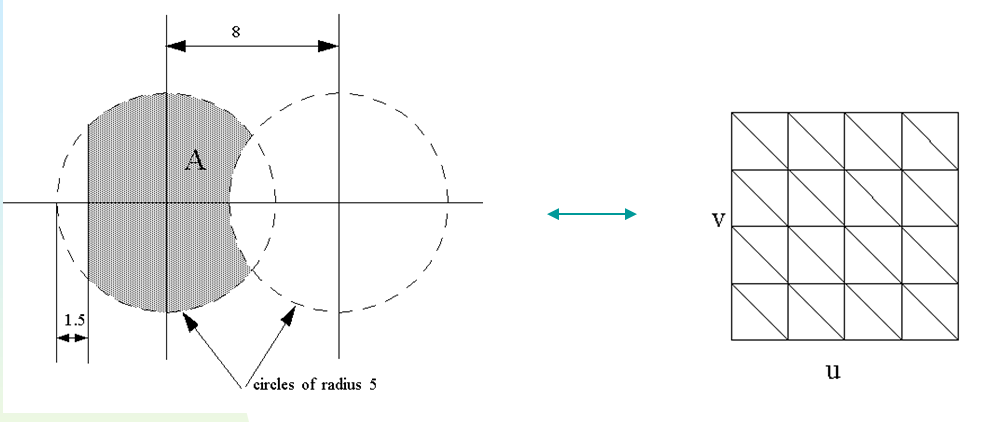
(2) You should allow the user to define different (*u*) and (*v*). One suggestion is that you can define the (*u*) (and (*v*)) as a Bezier curve with its two end control points on (0, 1) and (1, 0) respectively. For your convenience, you can fix the degree to 3 (i.e., 4 control points).

(3) Your program should be able to display the Coons patch by sampling the iso-curves and drawing their linear segments.

Note: when your program is examined, it will be viewed in the XY view; try to define 4 boundary curves that do not cross each other in the XY view.

**Programming Lab 3 – Coons patch mapping and smoothing**

Refer to the figure below. The boundary of the shaded region A is made of four curves: two arcs on the circle in the left, one vertical line segment, and one arc on the circle in the right.



1. Derive the Coons patch formulation based on these four curves to establish a mapping between A and the unit uv square [0,1]x[0,1].
2. Write a computer program, to calculate and display the points of the Coons patch corresponding to the equal-spaced sampling points on the uv square (please display between 100 – 200 points) and connect them to form the corresponding triangulation of A.
3. Calculate the surface area of the Coons patch by adding up the areas of the triangles in the mesh. (Note that, in this special 2D case, different (*u*)’s (and (*v*)’s) will actually have the same surface area (as the 2D boundary remains the same); however, if it is a 3D surface, then different (*u*)’s (and (*v*)’s) will have different surface areas even though the 3D boundary curves remain the same.)
4. Write a program that uses the Laplacian method to smooth the triangulation mesh of region A. You can assume the nodes on the boundary are fixed and only the interior nodes can move. Set an upper limit on the number of the iterations of Laplacian to 20, 40, and 100, and compare the results.

Note: Laplacian method is the averaging method. Let p1, p2, …, pk be the nodes in the triangulation that are connected to node q. Then, after one iteration for node q, the new location q becomes:

q = (((p1+ p2 + …+ pk )/k) + q)/2

There are other variances of the above. For example, let c1, c2, …, ck be the centroids of the triangles adjacent to q, then the new q can be

q = (c1+ c2 + …+ ck)/k.

You are welcome to try both or more.

**Programming Lab 4 – Optimization (minimal-area surface)**

In this project you will use Genetic Algorithm to find the blending functions (*u*) and (*v*) of Coons patch to construct a minimal-area Coons patch of four curves. The specifics are as follows.

1. The four input curves are Bezier curves of degree 5 (6 control points).
2. The (*u*) (also (*v*)) can be modeled as a Bezier curve of degree 3 with its first and fourth control point on (0, 1) and (1, 0) respectively. Also, let the *u* of the 2nd and 3rd control point be 0.333333 and 0.666667 respectively. Then, the four control points of (*u*) are (0, 1), (0.333333, 1), (0.666667, 2), and (1, 0); 1 and 2 are therefore the optimization variables (so are 1 and 2 for (*v*) ).
3. You need to ***numerically*** calculate the surface area of any Coons patch. Refer to Lab 3.(3).
4. You now have four variables1, 2, 1, 2} to search and the optimization objective is the surface area of the corresponding Coons patch. For simplification, you can assume that the range for 1, 2, 1, 2} is [0, 2].
5. In addition to the surface area, you can also try other types of optimization objectives, e.g., to minimize the maximum magnitude of the principle curvatures |κ1| and |κ2| of the surface (you need to numerically calculate κ1 and κ2).

[**https://blog.csdn.net/hjq376247328/article/details/49465137?utm\_source=blogxgwz7**](https://blog.csdn.net/hjq376247328/article/details/49465137?utm_source=blogxgwz7) **三角形求面积公式**

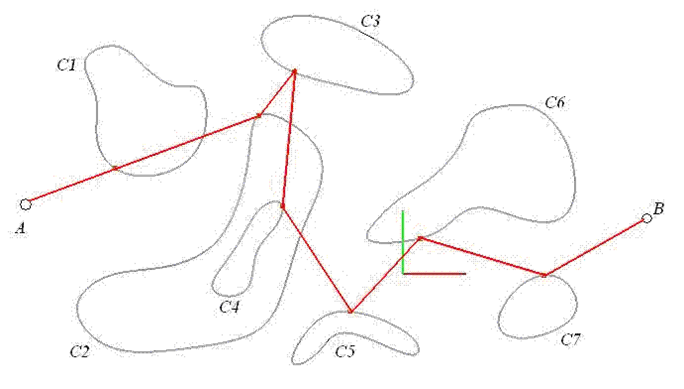
[**https://blog.csdn.net/Emiyasstar\_\_/article/details/6938608**](https://blog.csdn.net/Emiyasstar__/article/details/6938608) **遗传算法**

[**https://www.cnblogs.com/gaosheng12138/p/7534956.html**](https://www.cnblogs.com/gaosheng12138/p/7534956.html) **轮盘**

[**https://www.cnblogs.com/ddpeng/p/11302368.html**](https://www.cnblogs.com/ddpeng/p/11302368.html) **十进制二进制相互转化**

**Programming Lab 5 – Optimization of a trace**

Refer to the figure for an example: there are m = 7 curves and points A and B, the red polygon is a trace of the curves which goes through in the order C1, C2, .., C7; this curve is said to be minimal if its length is the shortest.



You will read in C*i* from a text file (in the (b) format of Lab 1) and key in the XYZ coordinates of points A and B.

Please do only one of (1) or (2) or (3). However, if you do more than one of them, you will get extra points.

(1) Write a computer program that will use GA to compute the minimum trace of the m input closed Bezier curves between the starting point A and the ending point B. Your program should allow the user to input the size of the population and the maximum number of iterations. Also, it should display the intermediate result ***after each iteration***.

(2) Write a computer program that will use the Local Optimum method to compute the minimum trace. Your program should allow the user to specify the initial solution (e.g., by giving the u-parameters of the m Bezier curves). ***Also, it should display the intermediate result after each iteration*** (e.g., the user hits a key, your program will run one round of local minimization on the 7 curves starting with C1, and then display the result, wait for the user to hit the key again). Local Minimum means, given two points p, q, and a Bezier curve C(u), the minimum trace of p, q, and C(u) is a function of u only, which can be found quickly as it is a one-dimensional optimization problem. A global minimum trace must also be a local minimum trace, but not the reverse.

(3) Write a computer program that will use the Steepest Ascend method to compute the minimum trace. Note that the length of the trace is a function D(u1, u2, …, u7) of 7 variables, where u*i* is the parameter of curve C*i* with range [0, 1]. Again, you should display the intermediate result ***after each iteration***.