Research Topic 10

Turing 1901

November 11, 2020

Table Of Contents

Description

Try to interpolate the profile of the following red bull. Can any of the interpolation methods that we have discussed be applied directly?

Extraction of coordinate points

- Extraction Contour
- Raw image $\rightarrow$ 1 Bit/Channel $\rightarrow$ Bitmap

- Now all pixels are represented by 0 or 1.

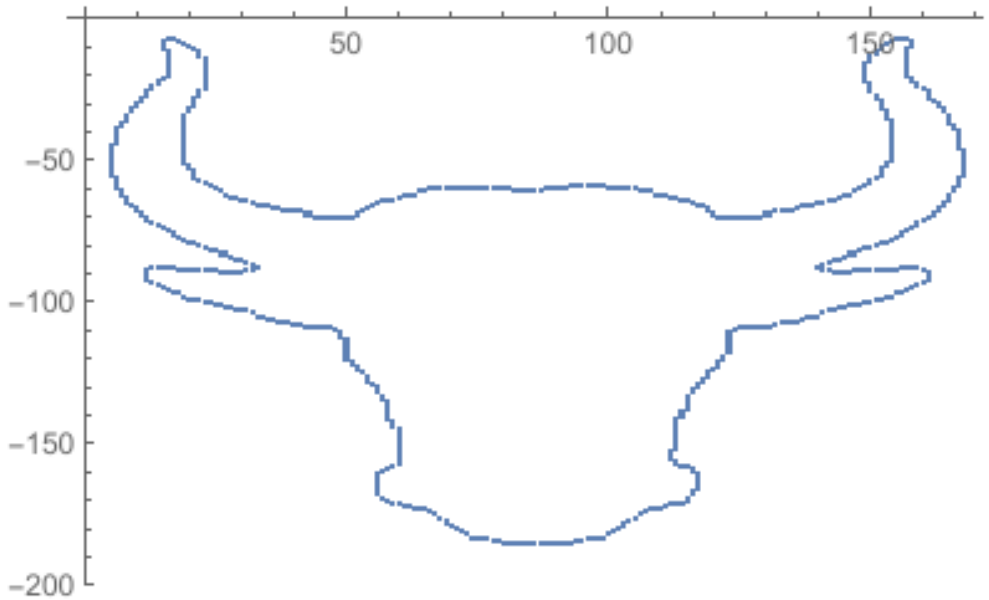
Extraction of coordinate points(Cont.)

1. Bfs at the 0 pixel in the lower left corner of the image and mark all the 1 around the 0 connectivity block.
2. Bfs from the center of the image, marking all the points encountered that were marked in the previous step.

The marked point in the second step is the contour of the graph.

Extraction of coordinate points(Cont.)

The Chebyshev distance between two adjacent points is less or equal to 2.



Lagrange Method

Lagrange Polynomial as below

$$P_n(x) = \sum_{i=0}^{n} L_{n,i}(x) y_i$$

where

$$L_{n,i}(x_j) = \delta_{ij} = \begin{cases} 0 & \text{for} \quad i \neq j \\ 1 & \text{for} \quad i = j \end{cases}$$

Thus we get

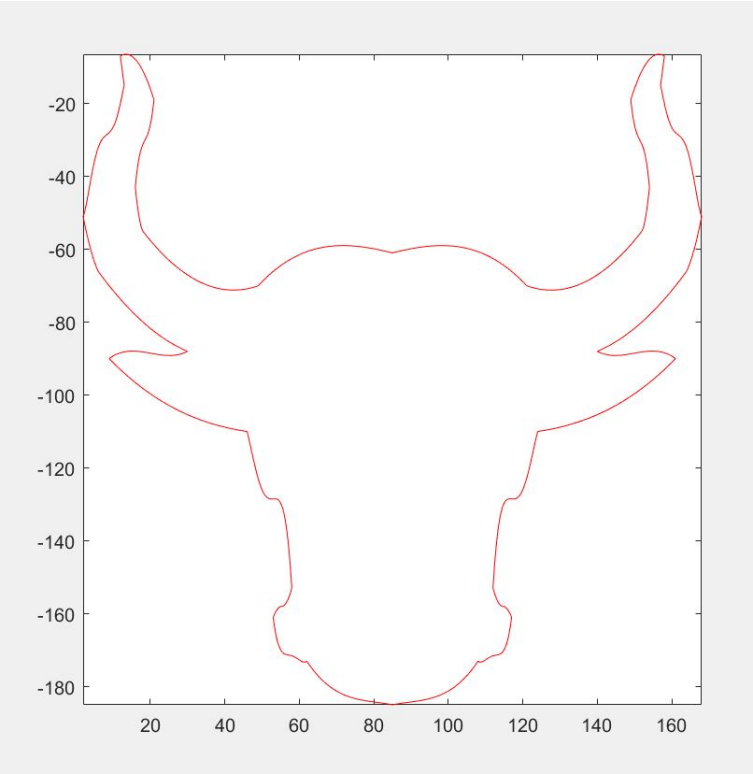
$$L_{n,i}(x) = \prod_{j \neq i \ \& \ j=0}^{n} \frac{(x - x_j)}{(x_i - x_j)} y_i$$

Matlab Code

```
function L=mylagrange(Points)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% L: Lagrange polynomial
%% Points: Matrix loaded with (xi,yi)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=size(Points,1);
x=sym('x');
P=0;
for i=1:n
    p=Points(i,2);
    for j=1:n
        if j~=i
            p=p*(((x)-Points(j,1))/(Points(i,1)-Points(j,1)));
        end
    end
    P=P+p;
end
L=P;
```

Result

Image obtained as below



Note: The yellow curve around the red bull considered as the profile.

Comments

- Hard to make the curve fit to the original image with too many points causing oscillating while too few points not enough to limit the curve.
- The derivative changes fast thus if we choose a long interval making the exponential function fits much better than polynomial so we choose not a little bit of intervals.
- The Lagrange Interpolation Method not giving limit at endpoints of each interval causes the image performing badly at those or looking sharp to be exact.

Newton's Interpolatory Divided Difference Formula

Newton's interpolatory divided difference formula is as below:

$$f(x) = N_n(x) + R_n(x)$$

where

$$N_n = f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) + \cdots + f[x_0, x_1, \cdots, x_n](x - x_0)(x - x_1)\cdots(x - x_n)$$

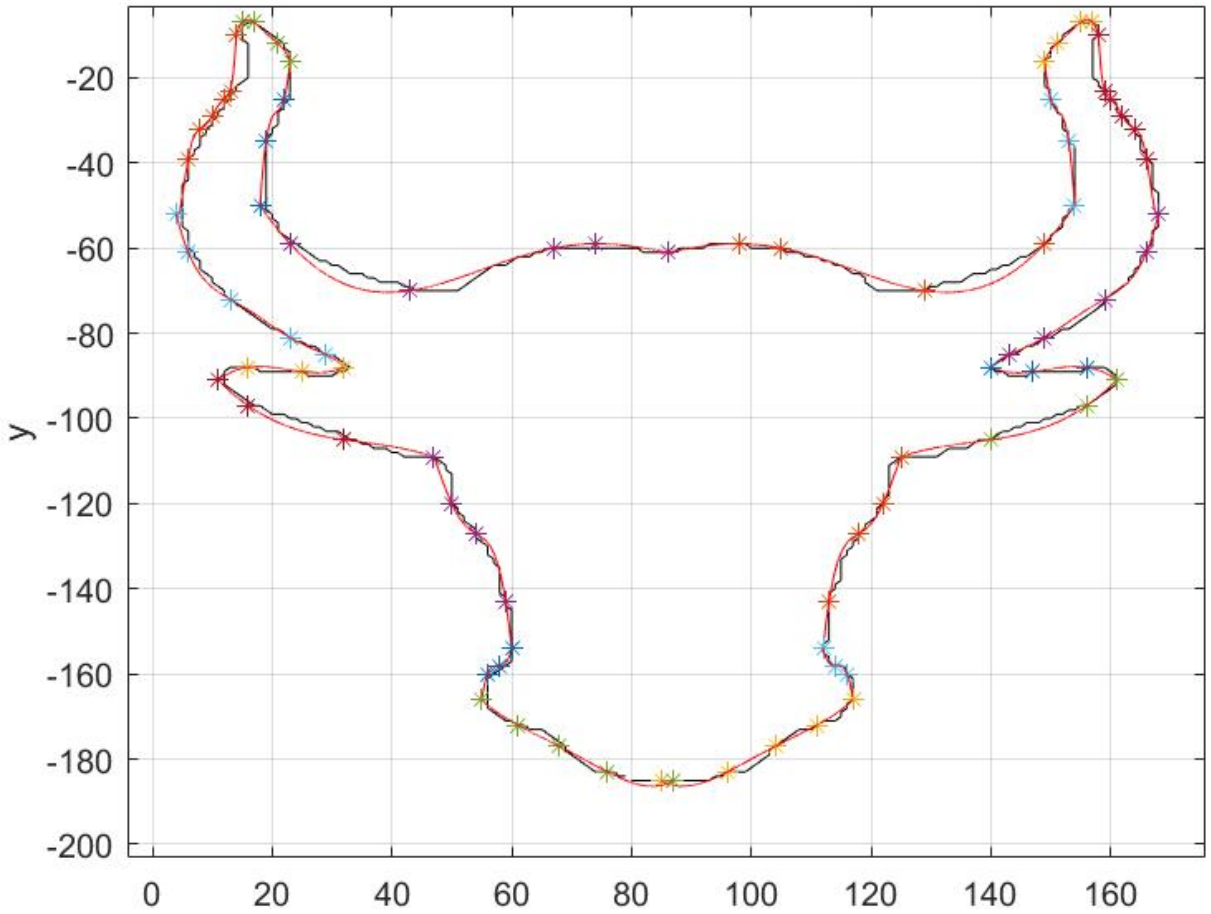
and

$$f[x_0, x_1, \cdots, x_k] = \frac{f[x_0, x_1, \cdots, x_k] - f[x_1, \cdots, x_k, x_{k+1}]}{x_0 - x_{k+1}}$$

$$= \frac{f[x_0, \cdots, x_{k-1}, x_k] - f[x_0, \cdots, x_{k-1}, x_{k+1}]}{x_k - x_{k+1}} = \sum_{i=0}^{k} \frac{f(x_i)}{\prod_{j=0, j\neq i}^{k}(x_i - x_j)}$$

Result

Image obtained as below:



Comparison

- In fact, Newton interpolation and Lagrange interpolation have the same methods to construct the fitting function, so the curve would be the same if we choose the same points.
- We all know, when using Newton interpolation, it would be easier and more flexible if we need to add one more point.

Comparison(Cont.)

- Because Newton interpolation has no need to count the derivative, if f(x) is made up of discrete points or its derivative don't exist, Newton interpolation is still useful.
- An advantage of using Newton's method instead of the Lagrange formula is that one usually obtains smaller discontinuities in the calculated interpolating polynomial.(Because of the round-off error of computer)

Hermite Interpolation

Hermite Interpolation

To obtain the coefficients of the Hermite interpolating polynomial $H(x)$ on the $(n+1)$ distinct numbers $x_0, \ldots, x_n$ for the function f:

INPUT numbers $x_0, x_1, \ldots, x_n$; values $f(x_0), \ldots, f(x_n)$ and $f'(x_0), \ldots, f'(x_n)$.

OUTPUT the numbers $Q_{0,0}, Q_{1,1}, \ldots, Q_{2n+1,2n+1}$ where

$$\begin{aligned} H(x) = Q_{0,0} &+ Q_{1,1}(x - x_0) + Q_{2,2}(x - x_0)^2 + Q_{3,3}(x - x_0)^2(x - x_1) \\ &+ Q_{4,4}(x - x_0)^2(x - x_1)^2 + \cdots \\ &+ Q_{2n+1,2n+1}(x - x_0)^2(x - x_1)^2 \cdots (x - x_{n-1})^2(x - x_n). \end{aligned}$$

Step 1 For $i = 0, 1, \ldots, n$ do Steps 2 and 3.

Step 2 Set $z_{2i} = x_i$;
$z_{2i+1} = x_i$;
$Q_{2i,0} = f(x_i)$;
$Q_{2i+1,0} = f(x_i)$;
$Q_{2i+1,1} = f'(x_i)$.

Step 3 If $i \neq 0$ then set

$$Q_{2i,1} = \frac{Q_{2i,0} - Q_{2i-1,0}}{z_{2i} - z_{2i-1}}.$$

Step 4 For $i = 2, 3, \ldots, 2n+1$

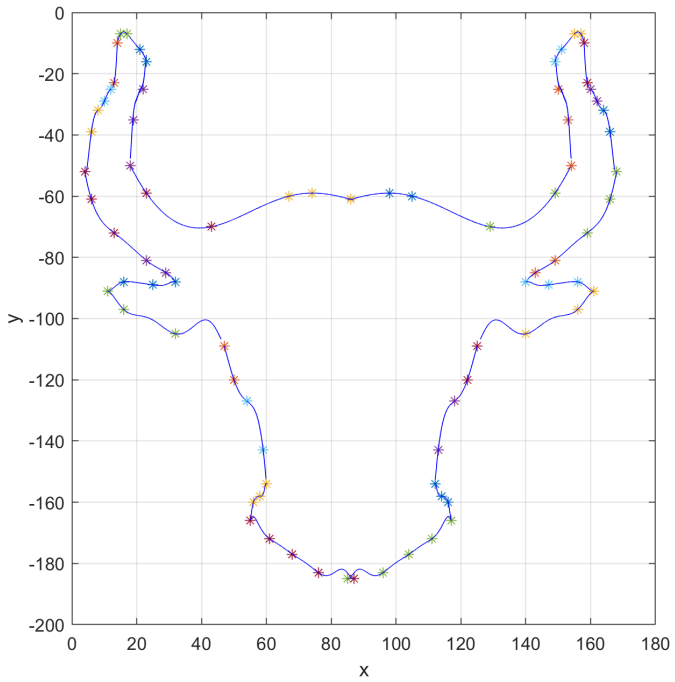
for $j = 2, 3, \ldots, i$ set $Q_{i,j} = \dfrac{Q_{i,j-1} - Q_{i-1,j-1}}{z_i - z_{i-j}}$.

Step 5 OUTPUT $(Q_{0,0}, Q_{1,1}, \ldots, Q_{2n+1,2n+1})$;
STOP

■

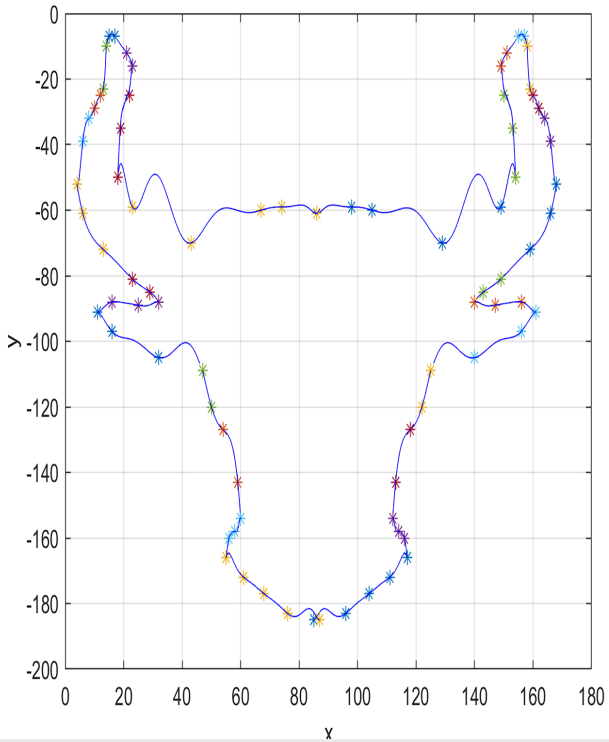
Result

- The derivative is obtained by Newton interpolation method
- The mean value of the derivative of two sections of function is used at the end point



Problem

Hermite Interpolation needs to divide the data into more segments and use more accurate derivatives, otherwise the over fitting phenomenon will be more serious.



Cubic Spline Interpolation

- Extract as many points as possible
- The set of pixels $Points$ corresponding to the extracted contour.
- Execute Cubic spline interpolation algorithm on $Points$.

Result

- Due to the nature of Cubic spline interpolation itself, there is no need to segment the point set.
- For the closed contour, making $M_0 = M_n$ yields n equations