Function to Output Mathematics Function Image

Bingcheng Guo

2019 年 12 月 6 日

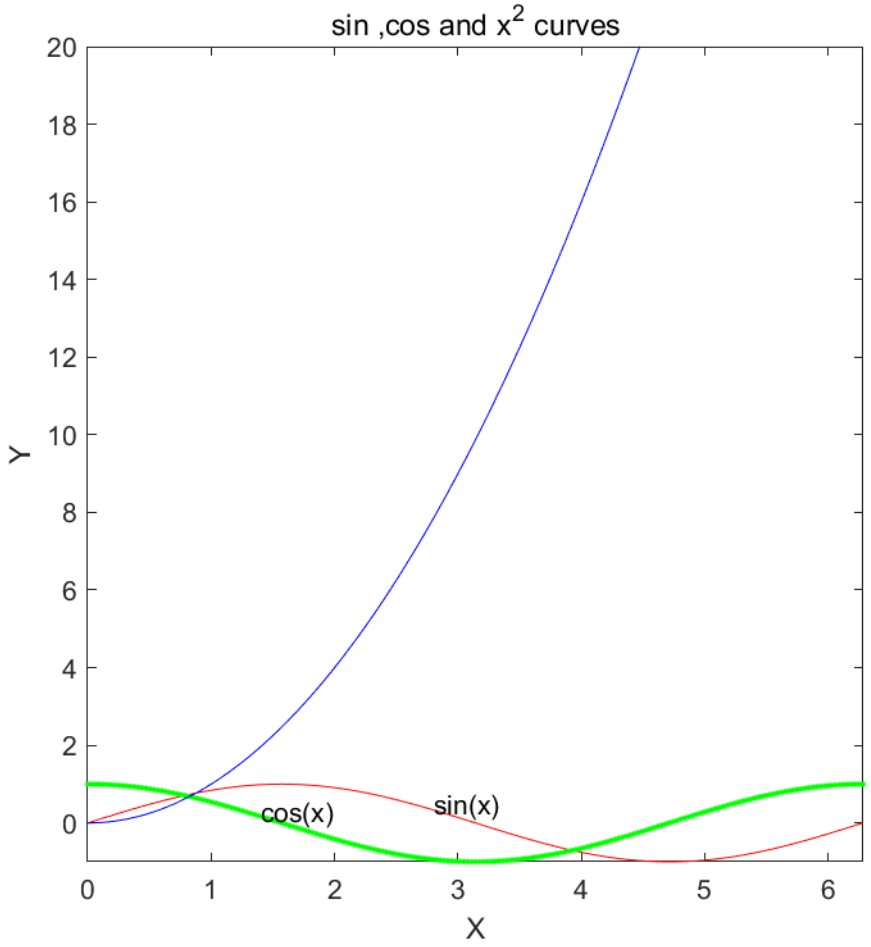
Matlab code listing :

```
%return figure
%references : width and height of the figure
function figure = generateFigure(imgW,imgH)
    x = 0:pi/100:2*pi;  %domain
    y1= sin(x);
    y2= cos(x);
    y3 = x.^2;

    set(gcf,'Position',[300 50 imgW imgH]);
    %set position and size of figure

    plot(x,y1,'r-',x,y2,'g.',x,y3,'b-');
    axis([0,2*pi,-1,20]);
    %set range of independent and
    %dependent variables to print

    title('sin ,cos and x^2 curves');
    xlabel('X');
    ylabel('Y');
    text(2.8,0.5,'sin(x)');
    text(1.4,0.3,'cos(x)');
```

```
22      print(gcf,'-dpng','res.png');
23       %print the figure to res.png
24
25      figure = imread('res.png');
26       %read the img and set the return value
```

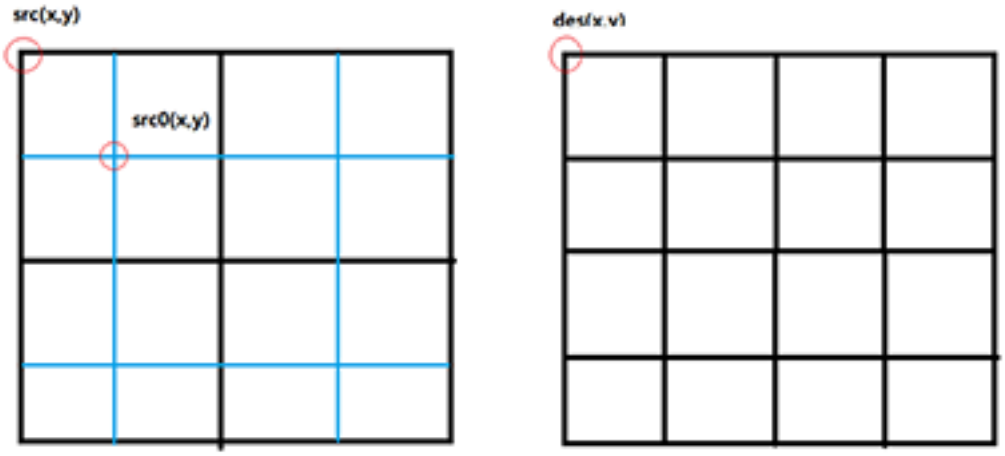
Command window:
img = generateFigure(500,500);
imshow(img);

Image as follows:



Bilinear interpolation Algorithm:

src(x,y) represent the value of the old image;des(x,y) represents the value of the new image; src1(x,y)represents the Pixel of the new image mapping to the old image;



Bilinear interpolation: the new image mapping to the old image, src0(x+u,y+v).then use the old four Pixel neighbor with the (x+u,y+v): src(x,y),src(x+1,y),src(x,y+1),src(x+1,y+1) to calculate the Pixel of the position (x+u,y+v);
Before X,Y axis convert,we need mapping the new image position (x_d,y_d) to the old position ($_c$,$_c$)(which is (x+u,y+v)):
We assume that the old image is M * N Pixels and the new image is W * H Pixels;

$$\begin{aligned} x_c &= x_d * ((M-1)/(W-1)) \\ y_c &= y_d * ((N-1)/(H-1)) \end{aligned} \tag{1}$$

In X axis:

$$\begin{aligned} src(x+u,y) &= (1-u) * src(x,y) + u * src(x+1,y) \\ src(x+u,y+1) &= (1-u) * src(x,y+1) + u * src(x+1,y+1) \end{aligned} \tag{2}$$

In Y axis:

$$src(x+u,y+v) = (1-v) * src(x+u,y) + v * src(x+u,y+1) \tag{3}$$

代码演示如下：

```
1    function figure = bilinear(imgW,imgH)
```

```
%UNTITLED2 此处显示有关此函数的摘要
%    此处显示详细说明
img = imread('Lena_L.png');
Temp = zeros(imgH,imgW);
S = size(img);
for x=0:imgH-1
    h=x*(S(1)-1)/(imgH-1);      %源图像的行号
    mh=floor(h);
    hf=mod(x*(S(1)-1),(imgH-1));
    for y = 0:imgW-1
    w=y*(S(2)-1)/(imgW-1); %原图像的列号
    mw=floor(w);
    wf=mod(y*(S(2)-1),(imgW-1));
        if (hf==0 && wf ==0)
                Temp(x+1,y+1)=img(h+1,w+1);
        elseif(hf==0 && wf ~=0)
            Temp(x+1,y+1)=(1-w+mw)*img(h+1,mw+1)+(
                w-mw)*img(h+1,mw+2); %同一行
        elseif(hf~=0 && wf ==0)
            Temp(x+1,y+1)=(1-h+mh)*img(mh+1,w+1)+(
                h-mh)*img(mh+2,w+1); %同一列
        else
            Temp(x+1,y+1)=(1-w+mw)*((1-h+mh)*img(
                mh+1,mw+1)+(h-mh)*img(mh+2,mw+1))+(
                w-mw)*((1-h+mh)*img(mh+1,mw+2)+(h-
                mh)*img(mh+2,mw+2));
        end
    end
end
imwrite(uint8(Temp),'res_4.png');
figure = imread('res_4.png');
end
```

具体思路就是先映射回原图像，再找原图像的周围四个点计算该点的值，但

是结果与 matlab 自带的结果有差异，算是简单理解。

可以查看目录下的 Lena_L.png 和 res_4.png 查看详情。