수치해석

과제 5 : 이미지 잘라내기

201310622 기계공학과 최홍성

201511182 소프트웨어 김성헌

201411196 소프트웨어 김영현

201711346 소프트웨어 정우혁

제출일: 2018.06.09

일차 시.

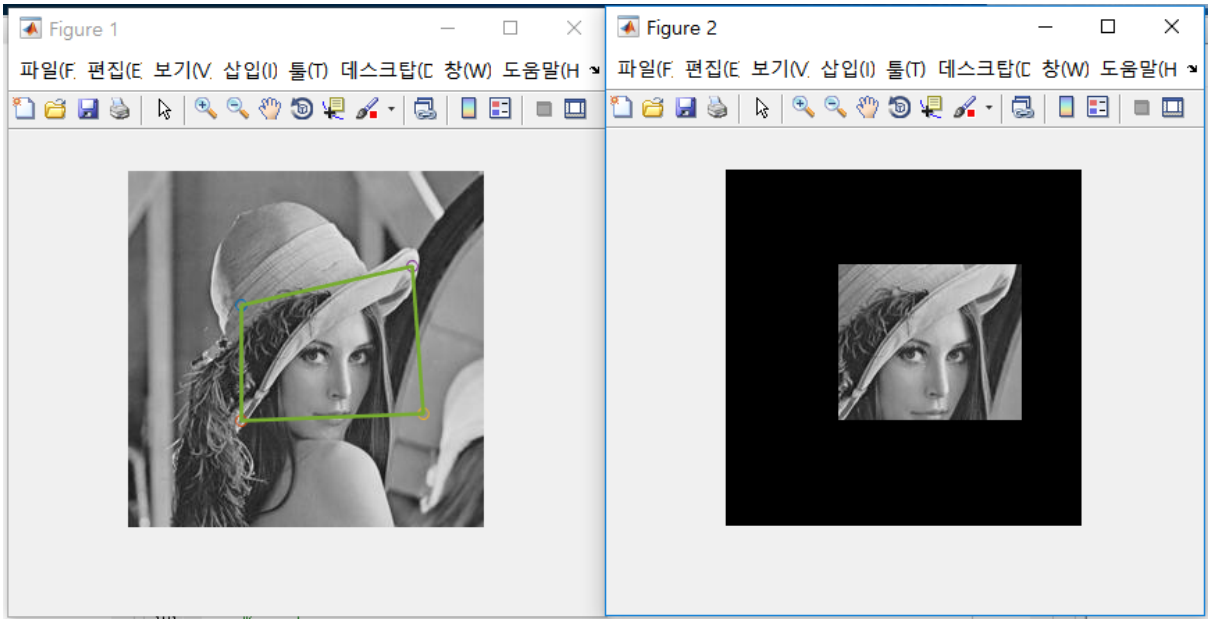
방법

- 우선 네 개의 점을 찍었을 때, 점의 제일 작은 x, y 좌표 값과 제일 큰 x, y 좌표 값을 min, max 함수를 이용하여 구하고, 그것을 이용하여 사각형을 만들어 그 부분 좌표의 값 만을 img에서 추출하여 출력하였다.

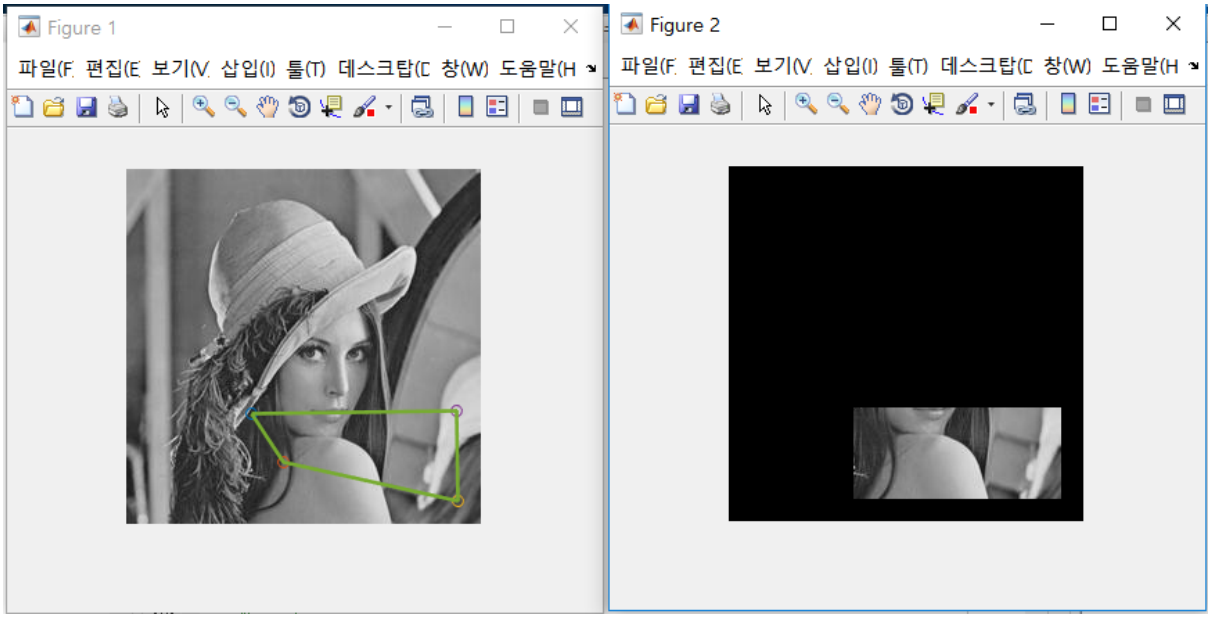
소스코드

```
clear
img = imread('lena.jpeg');
imshow(img)
hold on;
ptData = pData(4)



ptData2 = zeros(5,2);
ptData2 = ptData(:,:)
ptData2(5,:) = ptData(1,:)

hold on;
plot(ptData2(:,1),ptData2(:,2),'linewidth',2)
xd = ptData(:,1)
yd = ptData(:,2)
xd(5,1) = xd(1,1);
yd(5,1) = yd(1,1);

minx = min(xd);
maxx = max(xd);
miny = min(yd);
maxy = max(yd);

img2 = uint8(zeros(256,256));
 for i=minx:maxx
     for j=miny:maxy
         img2(j,i) = img(j,i);
     end
 end

 figure(2)
 imshow(img2);


figure(2)
plot(X1(1,:),Y1(1,:))
hold on;
plot(X2(1,:),Y2(1,:));
hold on;
plot(X3(1,:),Y3(1,:))
hold on;
plot(X4(1,:),Y4(1,:))
```

실험 결과



직사각형으로 찍었을 때, 출력되어지는 모습



직사각형이 아닌 다른 사각형으로 찍어도 직사각형 모양으로 출력됨..

결과 분석

- 단순히 네 개의 점만의 좌표를 사용하니까, 단조롭게 직사각형 밖에 추출되지 않았다. 사용자가 찍은 점에 따른 다각형의 모양에 따라서 이미지를 출력해야 한다.

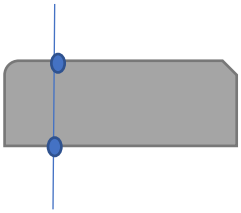
개선 방안

- 보간법을 이용하여 점과 점사이를 보간하여 정보를 받아와서 그 정보를 이용하여 사각형 안의 좌표정보를 추출하여야 한다.

이차 시.

방법

- 스플라인 보간법을 이용하여 점과 점사이를 보간한다. 보간한 x좌표와 y좌표의 정보는 각각 X,Y 에 저장한다. 그렇게 두 점 사이의 점들의 x좌표와 y좌표 정보를 알고나서, minx< x < maxx 부분에 x축과 수직인 직선을 긋는다. 그러면 다각형의 성질 상, 다각형과 직선이 두 점에서 만나게 된다. 그 성질을 이용하여 만나는 두 점의 정보를 저장하자. 그러면 그 두 점사이의 점들을 maxx – minx 만큼씩 추출해주면 된다. 이 두 점사이의 좌표들을 x가 변함에 따라 추출하면 사각형에 해당하는 좌표들이 나오고, 이 좌표들에 해당하는 img 값들을 추출하여 출력해주었다.



소스코드

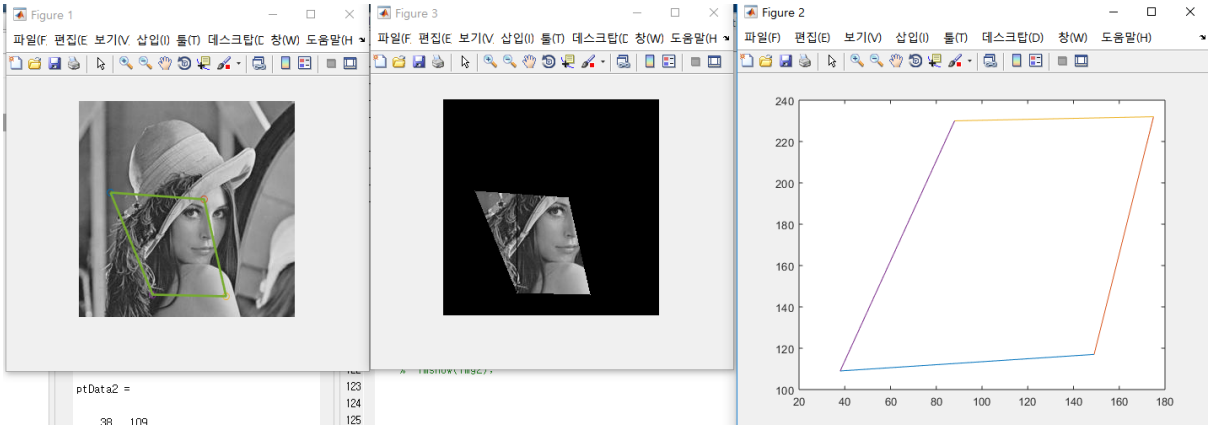
```
clear
img = imread('lena.jpeg');
imshow(img)
hold on;
ptData = pData(4)



ptData2 = zeros(5,2);
ptData2 = ptData(:,:)
ptData2(5,:) = ptData(1,:)

hold on;
plot(ptData2(:,1),ptData2(:,2),'linewidth',2)
xd = ptData(:,1)
yd = ptData(:,2)
xd(5,1) = xd(1,1);
yd(5,1) = yd(1,1);

 j=1;
```

```
 xd1 = [xd(1,1),xd(2,1);];
 yd1 = [yd(1,1),yd(2,1);];

 if(xd(1,1)<xd(2,1))
for i1=xd(1,1):1:xd(2,1)
   X1(1,j) = i1;
   Y1(1,j) = spline(xd1,yd1,i1);
   j=j+1;
end

 else
    for i1=xd(1,1):-1:xd(2,1)
   X1(1,j) = i1;
   Y1(1,j) = spline(xd1,yd1,i1);
   j=j+1;

end
 end

 j=1;
  xd2 = [xd(2,1),xd(3,1);];
 yd2 = [yd(2,1),yd(3,1);];

 if(xd(2,1)<xd(3,1))

    for i1=xd(2,1):1:xd(3,1)
   X2(1,j) = i1;
   Y2(1,j) =  spline(xd2,yd2,i1);
   j=j+1;

end

 else
    for i1=xd(2,1):-1:xd(3,1)
   X2(1,j) = i1;
   Y2(1,j) = spline(xd2,yd2,i1);
   j=j+1;

end
  end

  j=1;
   xd3 = [xd(3,1),xd(4,1);];
 yd3 = [yd(3,1),yd(4,1);];
  if(xd(3,1)<xd(4,1))

    for i1=xd(3,1):1:xd(4,1)
   X3(1,j) = i1;
   Y3(1,j) =  spline(xd3,yd3,i1);
   j=j+1;

end

 else
    for i1=xd(3,1):-1:xd(4,1)
   X3(1,j) = i1;
   Y3(1,j) =  spline(xd3,yd3,i1);
```

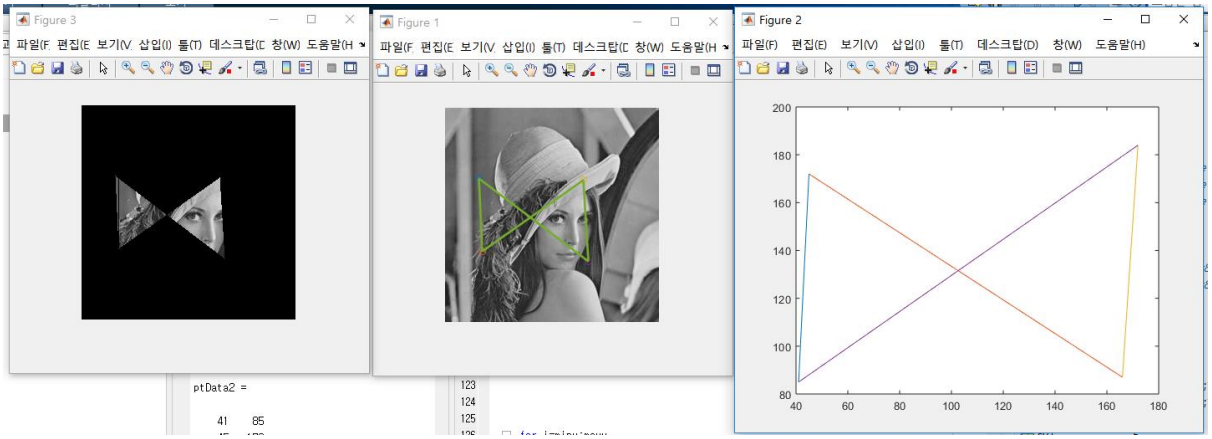
```
    j=j+1;

end
  end

  j=1;
  xd4 = [xd(4,1),xd(5,1);];
 yd4 = [yd(4,1),yd(5,1);];
  if(xd(4,1)<xd(5,1))

     for i1=xd(4,1):1:xd(5,1)
    X4(1,j) = i1;
    Y4(1,j) = spline(xd4,yd4,i1);
    j=j+1;

end

 else
     for i1=xd(4,1):-1:xd(5,1)
    X4(1,j) = i1;
    Y4(1,j) = spline(xd4,yd4,i1);
    j=j+1;

 end
  end

   [a,sy1] = size(Y1);
 [a,sy2] = size(Y2);
 [a,sy3] = size(Y3);
 [a,sy4] = size(Y4);

 minx = min(xd);
 maxx= max(xd);
 maxy= max(yd);
 miny = min(yd);

%  img2 = uint8(zeros(256,256));
%  for i=minx:maxx
%     for j=miny:maxy
%        img2(j,i) = img(j,i);
%     end
%  end
%
%  figure(2)
%  imshow(img2);



 for i=minx:maxx
    a1 = 0;
    a2=0;
    a3=0;
    a4=0;


    for j= 1:sy1
       if(i==X1(1,j))
```

```
            a1 = Y1(1,j);
        end
    end
    for j= 1:sy2
        if(i==X2(1,j))
            a2 = Y2(1,j);
        end
    end
    for j= 1:sy3
        if(i==X3(1,j))
            a3 = Y3(1,j);
        end
         end
for j= 1:sy4
        if(i==X4(1,j))
           a4 = Y4(1,j);
        end
end
ah = [a1 a2 a3 a4;];
    ah = sort(ah,'descend');

bx(1,i-minx+1)=i;
by1(1,i-minx+1)=ah(1,1);
by2(1,i-minx+1)=ah(1,2);
 end

 img2 = uint8(zeros(256,256));
 [r,c] = size(bx);
for j = 1: c
 for z=miny:1:maxy
    if(by2(1,j)<=z && z<=by1(1,j))
        img2(z,bx(1,j)) = img(z,bx(1,j));
    else
        img2(z,bx(1,j)) = 0;
    end

 end
end
figure(3)
 imshow(img2);

  figure(2)
plot(X1(1,:),Y1(1,:))
hold on;
plot(X2(1,:),Y2(1,:));
hold on;
plot(X3(1,:),Y3(1,:))
hold on;
plot(X4(1,:),Y4(1,:))
```

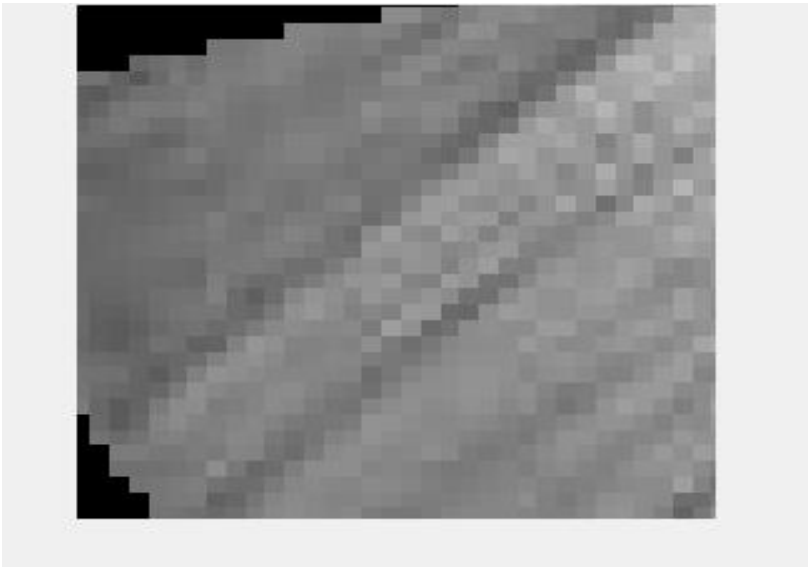
실험결과



직사각형이 아닌 사각형이어도 정확히 잘려서 추출되어지는 모습.



점의 순서를 다르게 해주어 나비넥타이 모양을 만들어도 출력 됨.



확대하였을 때, 매끄럽지 못한 부분도 생긴다.

결과 분석

- 두 점과 점 사이를 직선으로 보간하여서 확대하였을 때, 매끄럽지 못한 부분이 나타남

해결방안

- 스플라인 곡선을 이용하여 두 점 사이의 정보를 획득한 후, 위의 방식과 똑 같은 방식을 사용(x축과 수직인 직선을 그었을 때, 만나는 점은 두 점이다.)라는 방식으로 해준다.

삼차 시.

방법

- 한점과 한점을 스플라인 보간법을 사용하는 것이 아니라, 전체 점을 스플라인 보간법으로 보간한다. 그러면 그 점에 해당하는 곡선으로서 도형이 그려지게 된다. 두 번째 시도와 같은 방식으로 하려면 점과 점사이의 정보를 얻어와야 하기 때문에, for문을 사용하여 점과 점 사이의 x값과 y값을 따로 저장하여 주고, minx 와 maxx 사이의 x축과 수직인 직선을 그어서 만나는 두 점을 구하고, 그 사이의 값들을 img값의 값에서 추출하여 출력한다. 두 번째 시도와 같은 방식이지만 두번째 방식은 직선을, 세번째 방식은 곡선을 사용했다는 점에서 차이가 있다. 그리고 스플라인 보간법을 사용할 경우, 구해지는 x값이 소수점이 주로 나오기 때문에, 소수점에서 가장 가까운 정수를 찾아주는 함수 round 를 사용하여, x축과 수직인 직선을 그었을 때, 그에 해당하는 y값을 추출할 수 있게 하였다.

소스코드

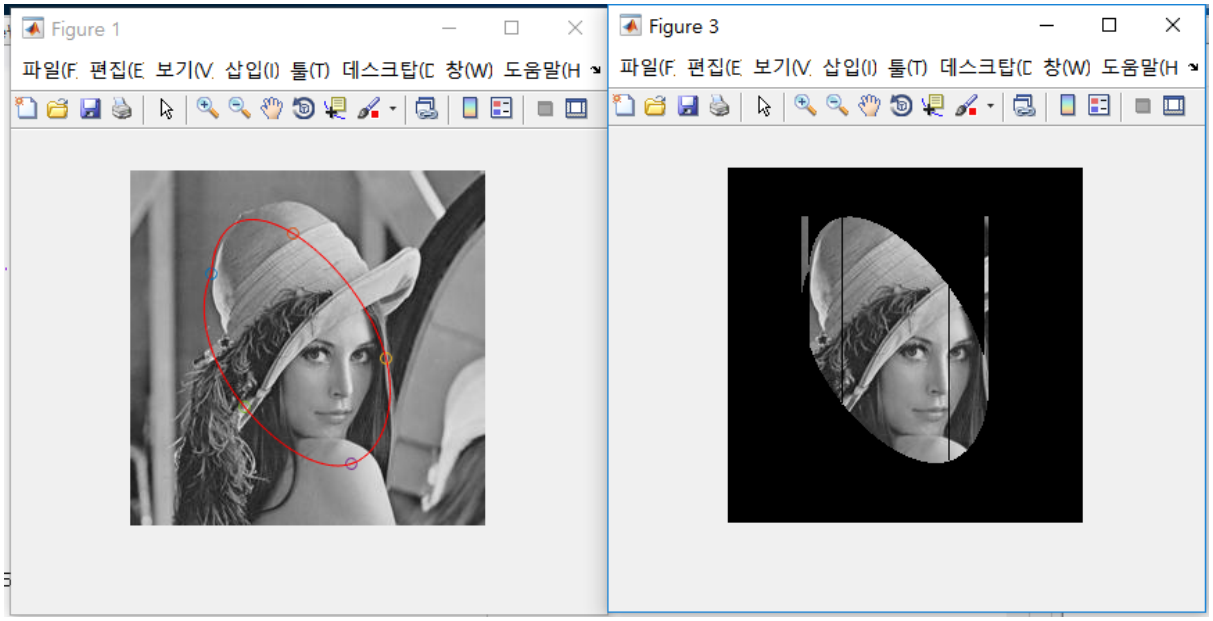
```
clear
img = imread('lena.jpeg');
[size1,size2] = size(img);
num = 5;
imshow(img);
hold on;
ptData = pData(num);
x = zeros(6,1);
y = zeros(6,1);
x(1:5,1) = ptData(:,1);
x(6,1) = ptData(1,1);
```

```
y(1:5,1) = ptData(:,2);
y(6,1) = ptData(1,2);
a = 1:num+1;
sxy = 1:1/400:num+1;
s = spline(a,x,sxy);
s2 = spline(a,y,sxy);
plot(s,s2,'-r')

[row,col] =size(s)
mins = min(s)
maxs = max(s)
mins2 = min(s2)
maxs2 = max(s2)

for i1=1:col
    if(x(2,1) == s(1,i1))
        break;
    end
end

for j=1:i1
    X1(1,j) =round(s(1,j))
    Y1(1,j) = s2(1,j);
end

for i2=i1:col
    if(x(3,1) == s(1,i2))
        break;
    end
end
z=1
for j=i1:i2
    X2(1,z) = round(s(1,j));
    Y2(1,z) = s2(1,j);
    z=z+1
end

for i3=i2:col
    if(x(4,1) == s(1,i3))
        break;
    end
end
z=1
for j=i2:i3
    X3(1,z) =round(s(1,j));
    Y3(1,z) = s2(1,j);
    z=z+1
end

for i4=i3:col
    if(x(5,1) == s(1,i4))
        break;
    end
end
z=1
for j=i3:i4
    X4(1,z) = round(s(1,j));
    Y4(1,z) = s2(1,j);
    z=z+1;
```

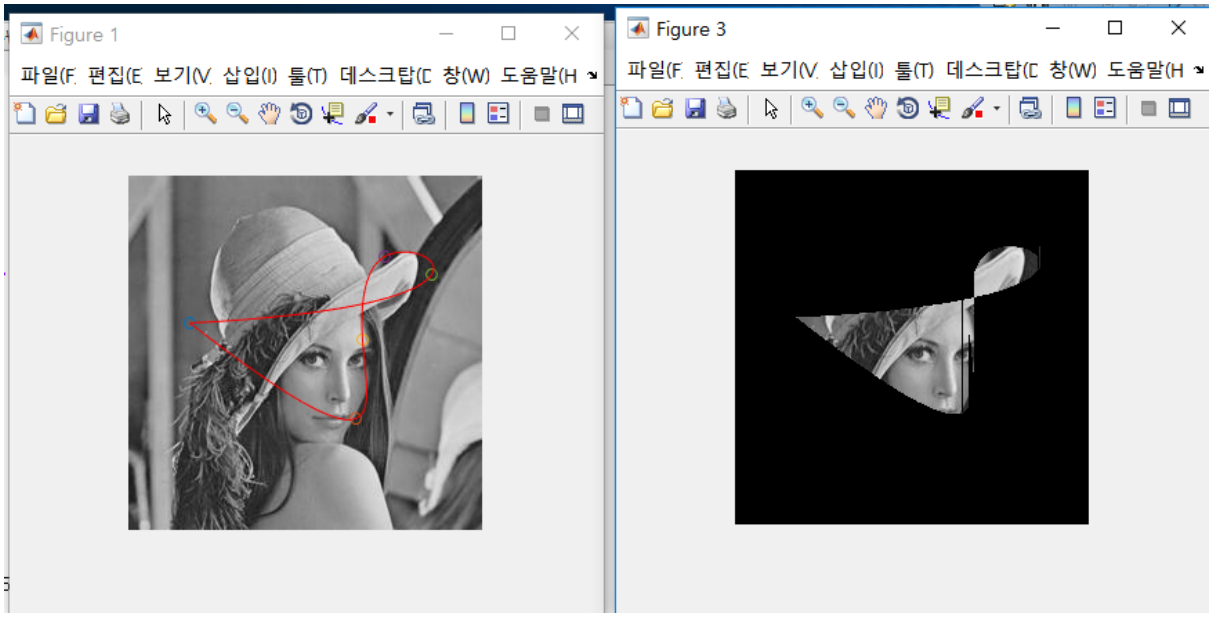
```
end

for i5=i4:col
    if(x(6,1) == s(1,i5))
        break;
    end
end
z=1
for j=i4:i5
    X5(1,z) = round(s(1,j));
    Y5(1,z) = s2(1,j);
    z=z+1;
end

[a,sy1] = size(Y1);
 [a,sy2] = size(Y2);
 [a,sy3] = size(Y3);
 [a,sy4] = size(Y4);
 [a,sy5] = size(Y5);


  z=1;



for i=round(mins):round(maxs)

    a1= 0;
     a2=0;
     a3=0;
     a4=0;
     a5 = 0;

    for j= 1:sy1
        if(i==X1(1,j))
            a1 = Y1(1,j);
        end
     end
     for j= 1:sy2
        if(i==X2(1,j))
            a2 = Y2(1,j);
        end
     end
     for j= 1:sy3
        if(i==X3(1,j))
            a3 = Y3(1,j);
        end
     end
for j= 1:sy4
        if(i==X4(1,j))
           a4 = Y4(1,j);
        end

end

for j= 1:sy5
        if(i==X5(1,j))
           a5 = Y5(1,j);
```

```
        end
end
ah = [a1 a2 a3 a4 a5;];
    ah = sort(ah,'descend');

bx(1,z)=i;
by1(1,z)=ah(1,1);
by2(1,z)=ah(1,2);

z=z+1;


 end

 img2 = uint8(zeros(256,256));
 [r,c] = size(bx);
for k = 1: c
 for z=round(mins2):1:round(maxs2)

    if(by2(1,k)<=z && z<=by1(1,k))
       img2(z,round(bx(1,k))) = img(z,round(bx(1,k)));
    else
        img2(z,round(bx(1,k))) = 0;
    end

 end
end
figure(3)
 imshow(img2);
```

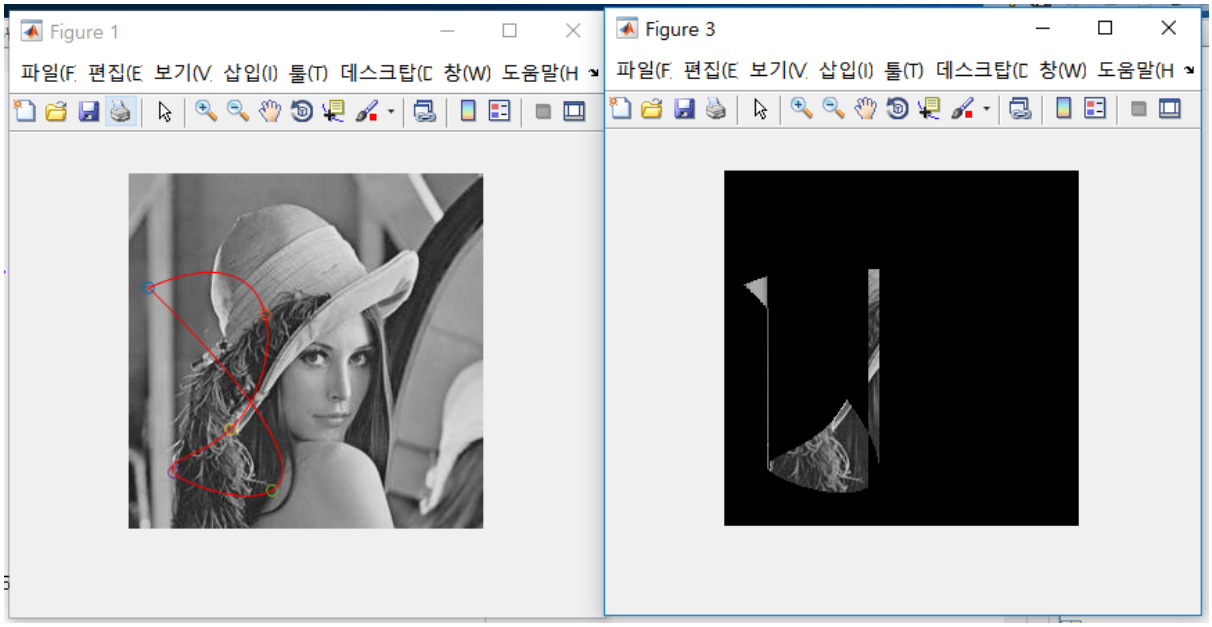
실험 결과



일반적으로 잘랐을 때의 추출되는 모습



꼬았을 때도, 추출되는 모습



만나는 점이 두 점보다 많을 경우, 제대로 추출되어지지 않는 모습.

결과분석

- 점을 이을 경우에 무조건 x축과 수직인 직선과 두 점에서 만날 거라 생각 했었는데, 위와 같을 때, 두 점보다 많이 만나는 경우가 있어서 제대로 좌표 값을 얻어내지 못하였다. 그리고 아무래도 round함수값을 써서 원래 값이 아닌 근사값을 썼기 때문에 오차가 발생하였다. 또 sxy를 좀 더 촘촘하게 하면 더 나은 보간 값을 가지고 더 매끄럽게 사진을 추출할 수 있었다.

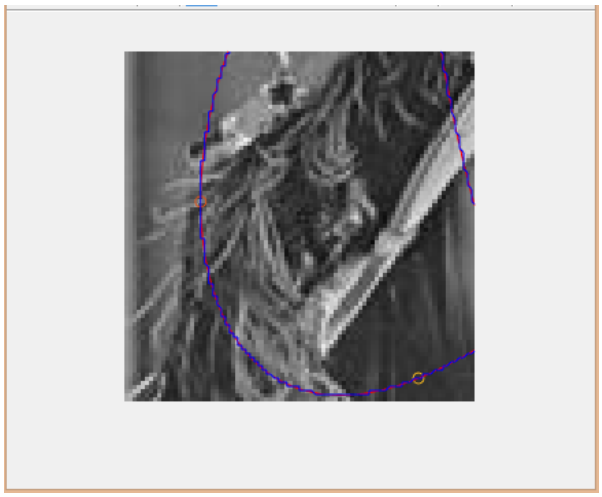
해결방안

- 애초에 저런 식으로 나비 넥타이 모양이 나오게 하는 것이 아니라, 점마다 순서를 지정해주어서 무조건 다각형이 생성되도록 해야 한다.

사차 시.

방법

- 이미지는 정수형 이차원 배열로 저장된다는 점을 이용하여 선택한 점을 이용해 x, y값을 각각 따로 생각해 스플라인 보간법을 이용해 점을 값을 구해준 다음, round 함수를 이용해 그 값들을 전부 정수로 바꿔준다. 그 점에 해당하는 곡선으로서 도형이 그려지게 된다. (해당 부분은 밑에 사진과 같이 큰 차이가 없다.) 그 이후 구해준 x 값을 오름차순 순으로 정렬해주고 for문을 사용하여 i번째 점과 i+1번째 점의 y 값을 이용해 해당 곡선 안에 있는 점들을 모두 옮겨주는 방식을 이용하였다.



(파란색이 원래 값, 빨간색이 round 함수를 이용해서 구해준 값이다.)

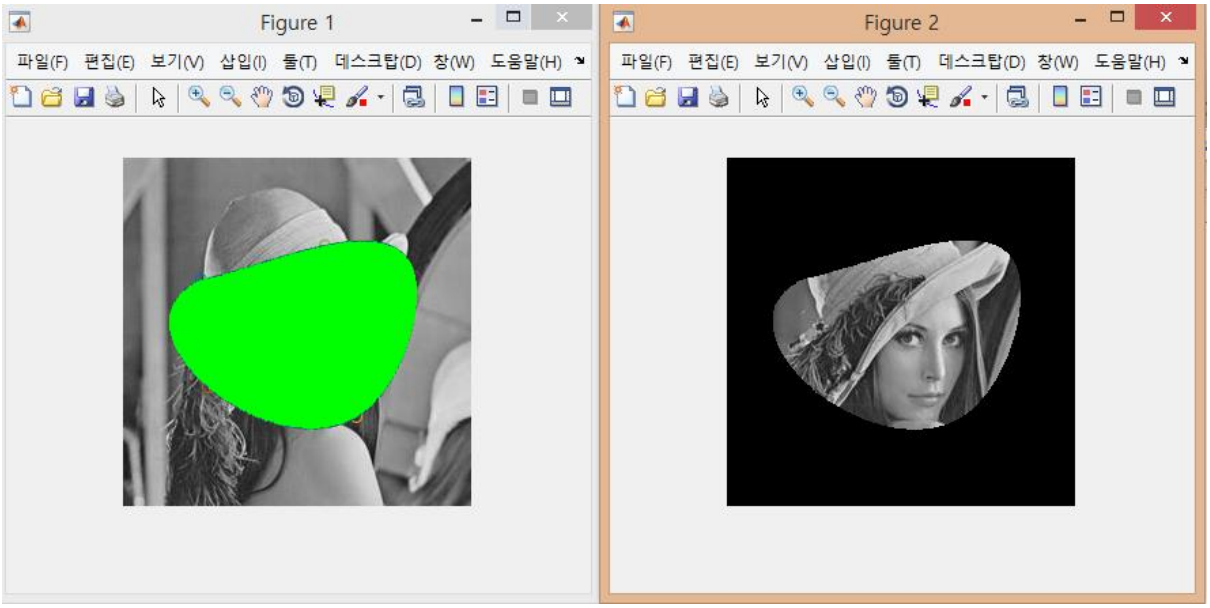
소스코드

```
img = imread('lena.jpeg');
[size1,size2] = size(img);
num = 5;
imshow(img);
hold on;
ptData = getPt(num);
x = zeros(6,1);
y = zeros(6,1);
x(1:5,1) = ptData(:,1);
x(6,1) = ptData(1,1);
y(1:5,1) = ptData(:,2);
y(6,1) = ptData(1,2);
a = 1:num+1;
sxy = 1:1/256:num+1;
s = spline(a,x,sxy);
s2 = spline(a,y,sxy);
plot(s,s2,'-r')
hold on;
round_s = round(s)
round_s2 = round(s2)
plot(round_s, round_s2, '-b')
[round_s_size_x, round_s_size_y] = size(round_s)
```

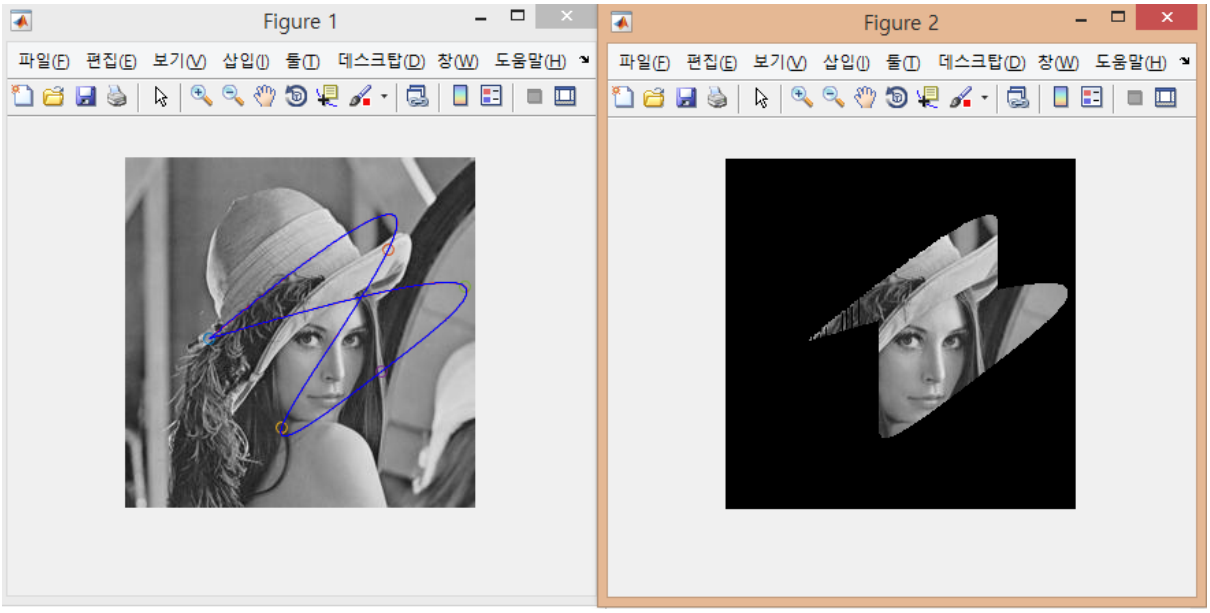
```
for i=1: round_s_size_y
    for j=1:round_s_size_y
        if(round_s(i)<round_s(j))
            tempa=round_s(i);
            round_s(i)=round_s(j);
            round_s(j)=tempa;
            tempb=round_s2(i);
            round_s2(i)=round_s2(j);
            round_s2(j)=tempb;
        end
    end
end
hold on;
plot(round_s, round_s2, '-g')
img2 = zeros(size1, size2); img2_t = uint8(img2);
for i=1: round_s_size_y-1
    if (round_s2(i) < round_s2(i+1))
            min_y = round_s2(i);
            max_y = round_s2(i+1);
            for j = min_y:max_y
             img2_t([min_y:max_y], round_s(i)) = img([min_y:max_y],
round_s(i));

            end
    else
         max_y = round_s2(i);
         min_y = round_s2(i+1);
        for j = min_y:max_y
            img2_t([min_y:max_y], round_s(i)) = img([min_y:max_y],
round_s(i));

        end
    end
end
figure
imshow(img2_t)
```

실험 결과

일반적으로 잘랐을 때의 추출되는 모습



만나는 점이 두 점보다 많을 경우, 제대로 추출되어지지 않는 모습.



결과분석

- 번째 사진처럼 점을 찍어주면 보간된 점들을 모두 이은 초록색 영역처럼 사진을 잘라낼 수 있었으나, 두 번째 사진처럼 점들을 찍어준 경우에는 선택하지 않은 영역의 값을 가져오는 것을 볼 수 있다. 이는 만나는 점이 두 점보다 많은 경우, 가장 작은 값과 큰 값의 사이에 있는 값을 선택하기 때문에 생긴 문제이다.

해결방안

- 을 그었을 때, 두 점보다 많은 경우를 조건을 줘서 나눠줘야한다.

오차 시.

방법

- 모두 찍은 후, 현재 점을 i, 점과 점사이의 거리를 비교하는 점을 j로 놓고 for문을 사용하여 각각의 점에서 최단거리를 순서대로 구한다. 단, 최단거리를 구할 때에는 이미 최단거리를 구한 점은 제외하며, 가장 처음의 최솟값은 이미지 사이즈에서 절대 나올 수 없는 거리인 sqrt(size1^2+size2^2)+1으로 사용한다. 한 점의 최단거리 검색이 끝나면, 그 점의 바로 다음 인덱스로 x와 y좌표를 재배열하고, 최솟값을 다시 초기의 최솟값으로 초기화 시켜준다.

소스코드

```
img = imread('lena.jpeg');
[size1,size2] = size(img);
num = 5;
imshow(img);
hold on;
ptData = getPt(num);
x = zeros(6,1);
y = zeros(6,1);
x(1:5,1) = ptData(:,1);
x(6,1) = ptData(1,1);
y(1:5,1) = ptData(:,2);
y(6,1) = ptData(1,2);
mini = sqrt(size1^2+size2^2)+1;
for i = 1:num-1
```

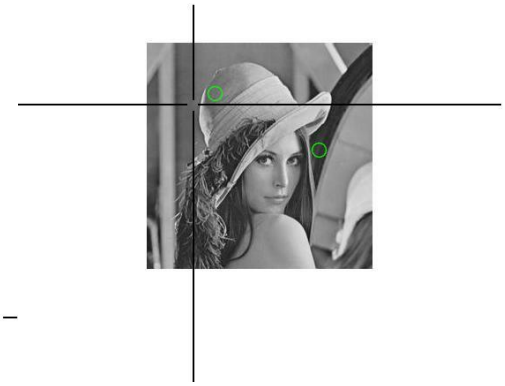
```
    for j = i+1:num
        if(mini > sqrt((x(j,1)-x(i,1))^2+(y(j,1)-y(i,1))^2))
            mini = sqrt((x(j,1)-x(i,1))^2+(y(j,1)-y(i,1))^2);
            swap_num = j;
        end
    end
    tmp = x(i+1,1);
    x(i+1,1) = x(swap_num,1);
    x(swap_num,1) = tmp;
    tmp = y(i+1,1);
    y(i+1,1) = y(swap_num,1);
    y(swap_num,1) = tmp;
    mini = sqrt(size1^2+size2^2)+1;
end
a = 1:num+1;
sxy = 1:1/256:num+1;
s = spline(a,x,sxy);
s2 = spline(a,y,sxy);
plot(s,s2,'-r')
hold on;
round_s = round(s);
round_s2 = round(s2);
plot(round_s, round_s2, '-b')
[round_s_size_x, round_s_size_y] = size(round_s);
for i=1: round_s_size_y
    for j=1:round_s_size_y
        if(round_s(i)<round_s(j))
            tempa=round_s(i);
            round_s(i)=round_s(j);
            round_s(j)=tempa;
            tempb=round_s2(i);
            round_s2(i)=round_s2(j);
            round_s2(j)=tempb;
        end
    end
end
hold on;
plot(round_s, round_s2, '-g')
img2 = zeros(size1, size2); img2_t = uint8(img2);
```

```
for i=1: round_s_size_y-1
      if (round_s2(i) < round_s2(i+1))
                  min_y = round_s2(i);
                  max_y = round_s2(i+1);
                  for j = min_y:max_y
                    img2_t((min_y:max_y), round_s(i)) = img((min_y:max_y), round_s(i));

                  end
      else
            max_y = round_s2(i);
            min_y = round_s2(i+1);
           for j = min_y:max_y
                img2_t((min_y:max_y), round_s(i)) = img((min_y:max_y), round_s(i));

           end
      end
end
figure
imshow(img2_t)
```

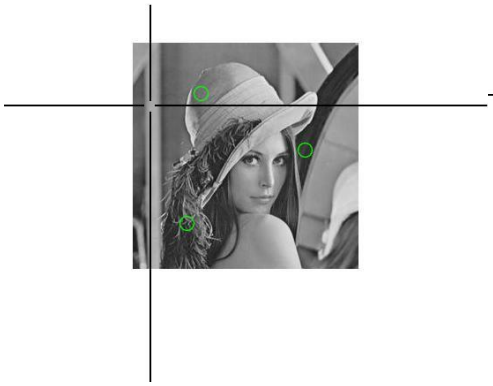
실험 결과

1. 첫번째 점을 찍었을 때 사진

2. 두번째 점을 찍었을 때 사진



3. 세번째 점을 찍었을 때 사진



4. 네번째 점을 찍었을 때 사진

5. 다섯번째 점을 모두 찍었을 때 사진

6. 깔끔하게 잘라진 이미지

하지만, 최단거리가 반드시 다각형을 만들지는 않는다.

깔끔하게 잘리기는 하지만 반드시 다각형을 만들지는 않는 모습이다.

결과분석

- 거리를 사용하여 주변의 점들 중에 가장 거리가 가까운 점을 다음 점으로 삼음으로써, 오류를 범하지 않도록 하였다. 하지만 두번째 예제와 같이 최단거리를 사용해도 다각형을 만들지 않는 경우의 수가 존재한다.

해결방안

- 형을 만들지 못하는 경우의 수를 없애는 새로운 조건이 필요하다.

후기

- 팀원들과 협동하여 팀 프로젝트를 진행하면서, 나와는 다른 사고방식을 가진 팀원들과 의견을 공유하고, 그 안에서 문제를 해결하기 위해 다양한 알고리즘을 구현 해 보았다. 그 속에서 서로가 서로에게 조언을 해주며, 그 조언들은 팀원 모두의 능력을 향상 시킬 수 있는 계기가 되었으며, 조금 더 나은 해결방안을 가져와 줄 수 있었다. 비록 완벽한 알고리즘을 구현하지 못했지만, 이번 과제를 통해 색다른 경험을 할 수 있었음에 감사함을 느끼는 바이다.