학번: 2017320215

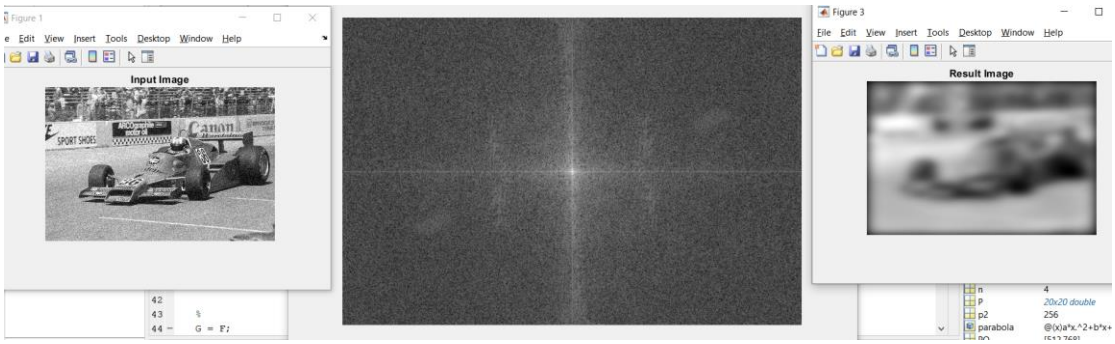
이름: 임준상

myLPF:

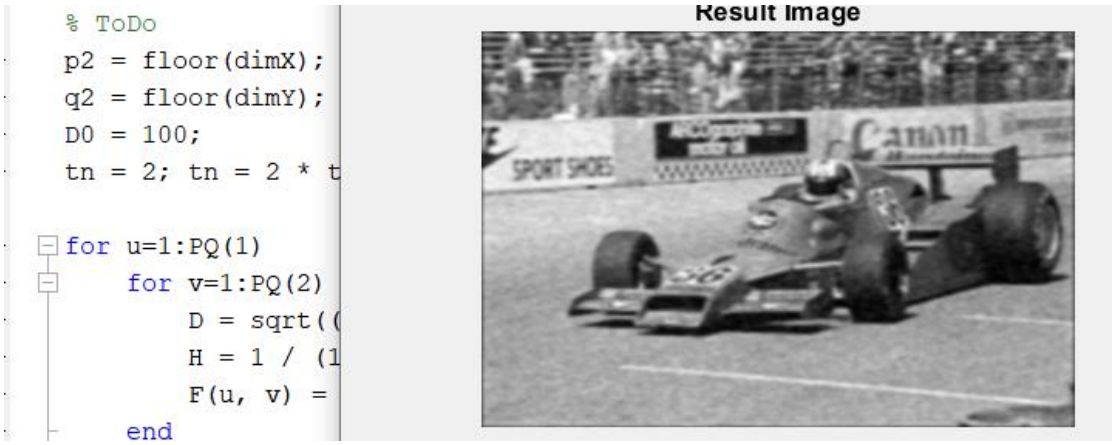
아래는 Butterworth LPF 를 구현한 코드이다.

```
28      % TODO
29 -    p2 = floor(dimX);      % P/2
30 -    q2 = floor(dimY);      % Q/2
31 -    D0 = 15;               % cutoff freq.
32 -    tn = 2; tn = 2 * tn;   % two n, 2n
33
34 -    for u=1:PQ(1)
35 -        for v=1:PQ(2)
36 -            D = sqrt((u-p2)^2 + (v-q2)^2);   % D(u, v)
37 -            H = 1 / (1+(D/D0)^tn);           % H(u, v)
38 -            F(u, v) = H * F(u, v);
39 -        end
40 -    end
41
42      %
```

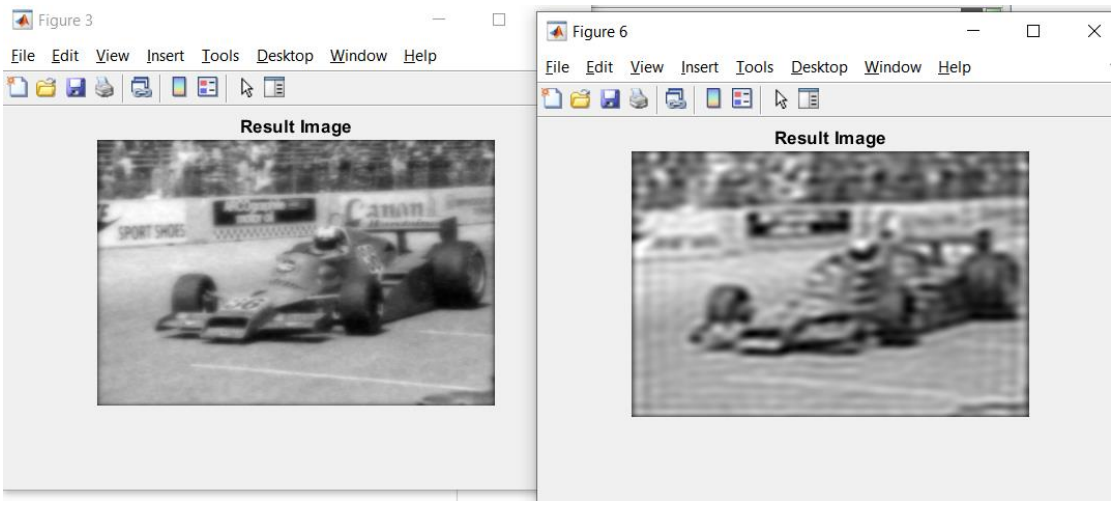
D0 이 15 이고 n 은 2 일 때 나오는 그림은 아래와 같다.



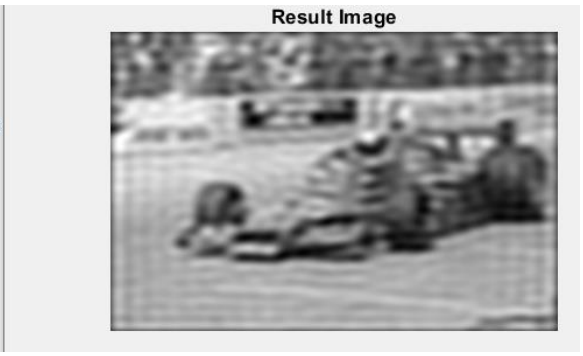
D0 의 값을 100 으로 바꿔서 그림이 두렷하게 되었다. Cut 되지 않은 내용이 많아지기 때문이다.



아래 그림 중 위에 있는 두 그림은 D0 이 50 이고 n 값이 1(왼쪽), 20(오른쪽)인 그림이며, 밑에는 Ideal LPF 를 사용해서 나오는 그림이다. n 이 작을 수록 loss 된 data 가 많으며, 클수록 Ideal LPF 와 비슷하다(직사각형).



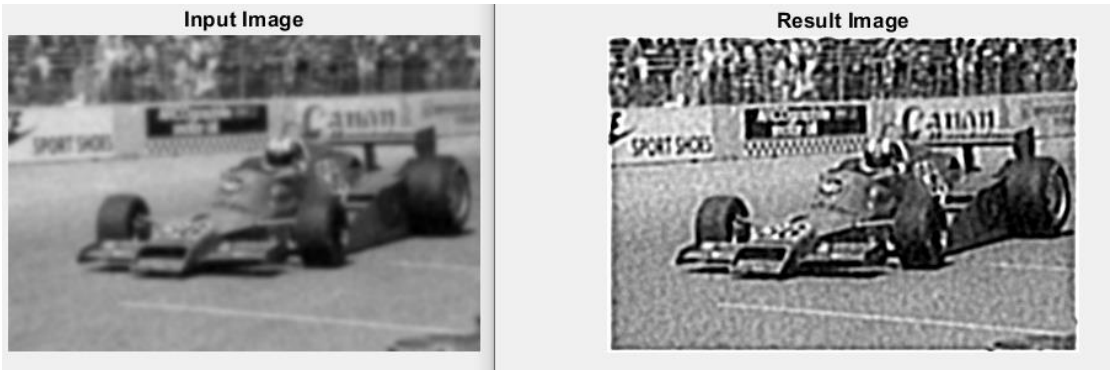

```
25 -    tn = 4;tn = 2 * tn;    % two n, 2n
26
27 -  for u=1:PQ(1)
28 -      for v=1:PQ(2)
29 -          D = sqrt((u-p2)^2 + (v-q2)^2);
30 -          if D < D0
31 -              H = 1;
32 -          else
33 -              H = 0;
34 -          end
35            %H = 1 / (1+(D/D0)^tn);
36 -          F(u, v) = H * F(u, v);
37 -      end
```



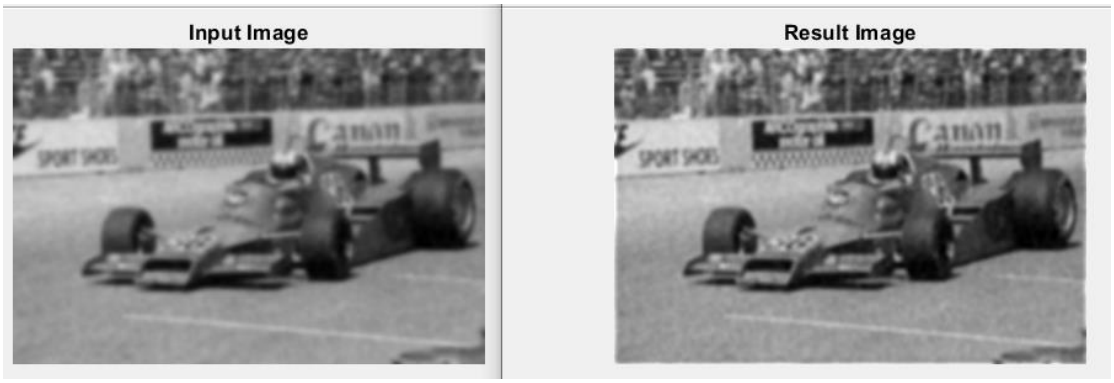
myHBF:

```
29      % ToDo
30 -    p2 = floor(dimX);      % P/2
31 -    q2 = floor(dimY);      % Q/2
32 -    D0 = 70;               % cutoff freq.
33 -    tn = 2; tn = 2 * tn;   % two n, 2n
34 -    k = 10;                % boosting weight
35
36 -  for u=1:PQ(1)
37 -      for v=1:PQ(2)
38 -          D = sqrt((u-p2)^2 + (v-q2)^2);  % D(u, v)
39 -          H = 1 / (1+(D/D0)^tn);          % H(u, v)
40 -          Hhp = 1 - H;                    % Highpass Filters
41 -          F(u, v) = (1+k*Hhp)*F(u, v);
42            %F(u, v) = Hhp*F(u, v);
43 -      end
44 -  end
45
46      %
47 -    G = F;
```

D0 이 70 이고 n 은 2 인 Butterworth LPF 를 사용해서 Highpass Filters 를 구한다. 아래 그림은 k 가 10 으로 조정되었을 때 나오는 그림이다.



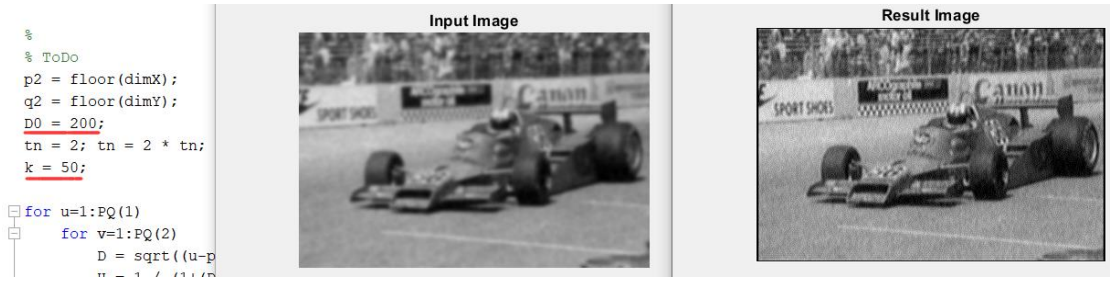
k 가 1 로 조정되면 나온 결과는 다음과 같다.



k 가 클 수록 F(u, v)에 더하는 숫자가 커지고 효과가 분명하게 나오는 반면, k 가 작을 수록 input 의 차이가 작다.

```
F(u, v) = (1+k*Hhp)*F(u, v);
```

D0 값이 200 으로, k 가 50 으로 조정되면 나오는 그림은 다음과 같다. D0 값이 크기 때문에 높은 frequence 만(예: edge) 사용하며 k 가 크기 때문에 효과가 많이 강화될 것이다. 그래서 결과그림이 input 그림보다 많이 두렷하게 나왔다. Edge 가 더 명확해졌기 때문이다.

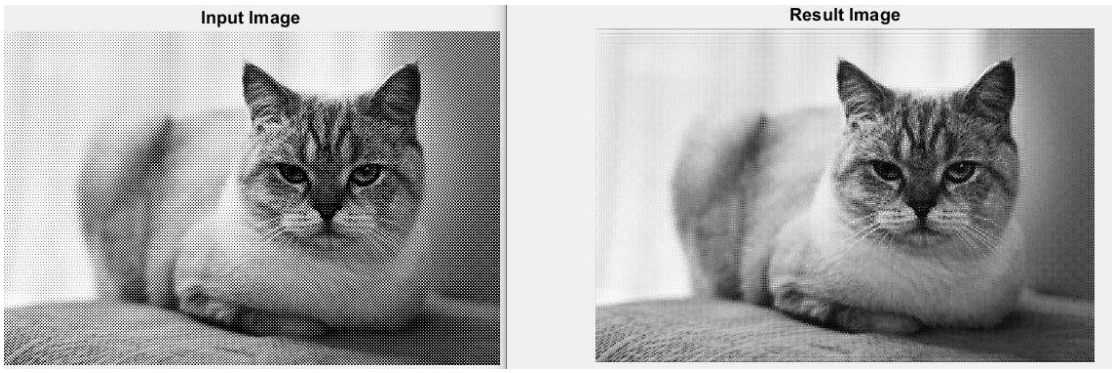


myNotch:

```
    % TODO
    p2 = floor(dimX);      % P/2
    q2 = floor(dimY);      % Q/2
    D0k = [45, 45, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30, 30];              % cutoff freq.
    tn = 6; tn = 2 * tn;  % two n, 2n

    % Coordinates of each notch
    uk = [213, -213, 259, 0, 258, -258, 45, -45, 213, -213, 213, -213, 303, -303, 167];
    vk = [319, 319, 0, 386, 386, 386, 318, 318, 68, 68, 454, 454, 319, 319, 0];
    len = size(uk, 2);

for u=1:PQ(1)
    for v=1:PQ(2)
        for i=1:len
            Dk = sqrt((u-p2-uk(i))^2 + (v-q2-vk(i))^2);  % Dk(u, v)
            Dnk = sqrt((u-p2+uk(i))^2 + (v-q2+vk(i))^2);  % D-k(u, v)
            Hk = 1 / (1+(D0k(i)/Dk)^tn);           % Hk(u, v)
            Hnk = 1 / (1+(D0k(i)/Dnk)^tn);           % H-k(u, v)
            Hnr = Hk*Hnk;                      % Highpass Filters
            F(u, v) = Hnr*F(u, v);
            %F(u, v) = 1*F(u, v);
        end

    end
end

    %
    G = F;
```

Notch 의 위치가 uk 와 vk 에 저장되어 있고 각 notch 의 D0 또 D0k 에 저장되어 있다.

위의 코드를 실행하면 나오는 그림은 아래와 같다.



그리고 아래는 Input 그림과 result 그림의 Fourier spectrum(스펙트럼?)이다.