영상처리 9주차 과제

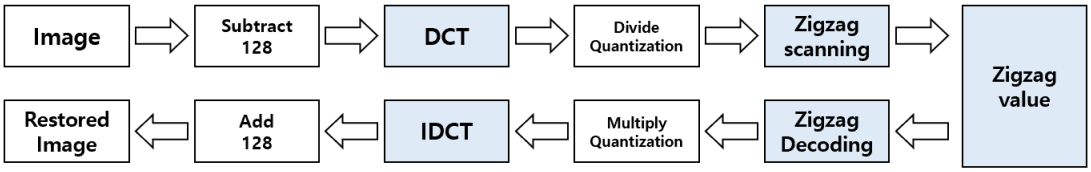
학번: 201404376

이름: 정성욱

1. **과제 내용**

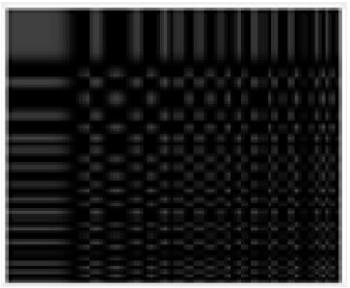
목표 JPEG에 대한 encoding구현

Jpeg 압축 과정:



Encoding은 zigzag value를 도출해 내는것임

1. **우선 이미지를 8x8의 블록으로 나눔**
2. **나누어진 블록에 대해 8x8 DCT를 구해야하는데 그냥 dct필터를 씌우는 것임**
   각 나눠진 8x8 하나의 블록에 대해



이 64개의 필터를 하나씩 씌워서 더하는 것임. (어두운 이유는

$$F(u,v) = C(u)C(v)\sum_{y=0}^{n-1}\sum_{x=0}^{n-1} f(x,y)\cos\left(\frac{(2x+1)u\pi}{2n}\right)\cos\left(\frac{(2y+1)v\pi}{2n}\right)$$

에서 f(x,y)가 빠짐(이미지 픽셀에 대한 계산은 나중에 따로 계산하게 하려고 함

3. **DCT를 적용한 이미지에 대해서 이미 존재하는 DCT 테이블로 각 픽셀을 나눠 준 후**

round를 취해 준다.

4. 이후에 나온 픽셀들에 대해 지그제그 스캐닝을 해서 1차원 리스트로 만들어줌
5. 이 과정을 이미지를 8x8로 나눈 후 모든 픽셀에 적용 한다.

코드 설명

```
function C=set_C(w,n)
    if(w==0)
        C=sqrt(1/n);
    else
        C=sqrt(2/n);
    end
end

function [until_zero_list,zero_index] = get_zero_idx(ttt)
     [ti,tl] =size(ttt);
    this_index = 1;
    for ki = 1:tl
        if(ttt(ki)~=0)
            this_index = ki;
        end
    end
   zero_index=this_index;
   until_zero_list = ttt(1:this_index);
end
```

set_C는 필터를 만드는데 쓰이는 함수로 매개변수의 값이 0 일경우 sqrt(1/n)를 리턴하고 그외의 경우엔 sqrt(2/n)을 리턴함

$$C(w) = \begin{cases} \sqrt{1/n} & if\ w = 0 \\ \sqrt{2/n} & otherwise \end{cases}$$

get_zero_idx는 리스트 형태로 만드는 것입니다. 이 스캐닝을 해서 리스트가 어디부터 0만이 존재하는 지를 1차원 리스트로 만들어서 추가한 리스트와 어디서부터 시작되는지에 대한 인덱스값을 리턴합니다.

```
function zz= make_zigzag(dct_list)
    list = [];
idx=1;
idy=1;
count=1;
decrease = false;
increase = true;
mode_change=false;
```

지그제그 스캐닝을 해서 1차원 리스트로 만들어주는 함수입니다.

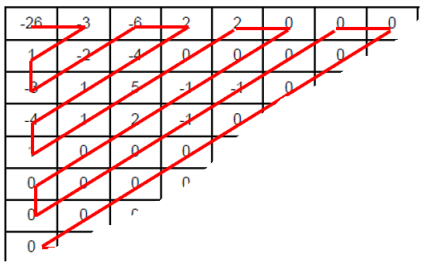
우선 코드가 길어서 변수부분만 짜르고 오르는 부분 내려가는 부분 그리고 중앙을 기준으로 값이 점점 줄어드는 모드 체인지로 구현했습니다.

```
33 –    while true
34 –        [chkx,chky] =  size(list);
35 –        if(chky==64) break
36 –        end
37
38 –        if(mode_change==false)
39 –          if(idy==1 && idx==1)
40 –             list(count) = dct_list(idx,idy);
41 –             count=count+1;
42 –            idy=idy+1;
43 –            decrease = true;
44 –            increase = false;
45 –          elseif(decrease)
46 –            idx=1;
47 –            for iy = idy:-1:1
48 –              list(count) = dct_list(idx,iy);
49 –              idx=idx+1;
50 –              count=count+1;
51 –            end
52 –            decrease = false;
53 –            increase = true;
```

첫번째 인덱스일 경우 지그제그의 첫번째 값에 첫번째 값을 넣고 시작합니다.

그후 다시 루프를 돌아서 하나씩 추가를 하되 일부러 오른쪽으로 가서 아래로 내려오도록 설계했습니다. Decrease true 먼저 실행함 끝나면 decrease false로 두고 increase를 true로 두므로써 여기서도 위로 올라가는 모드 아래로 내려가는 모드로 나눔 increase 끝날시 자기자신은 false로 두고 decrease를 true로 두도록 변경함.

```
54 –              elseif(increase)
55 –                idy=1;
56 –                for ix = idx:-1:1
57 –                  if(ix>8)
58 –                      idy=2;
59 –                      idx=8;
60 –                      mode_change=true;
61 –                      decrease = false;
62 –                      increase = true;
63 –                      break;
64 –                  end
65 –                  list(count) = dct_list(ix,idy);
66 –                  count=count+1;
67 –                  idy=idy+1;
68 –                end
69 –              if(mode_change==false)
70 –              decrease = true;
71 –              increase = false;
72 –              end
73 –            end
```



위의 코드는 여기까지 함 이후 모드 체인지로 바꿈

```
74 -          else
75 -          if(decrease)
76 -              laci=0;
77 -              ix =  idx;
78 -              for aci =idy:-1:idx
79 -                list(count) =  dct_list(ix,aci);
80 -                ix=ix+1;
81 -                count=count+1;
82 -                laci=aci;
83 -              end
84
85 -              if(idx>8) idx=8; end
86 -                tmp = idx+1;
87 -                idx = 8;
88 -                idy = tmp;
89 -              decrease = false;
90 -              increase = true;
91
92 -          elseif(increase)
93 -              laci = 0;
94 -              iy=idy;
95 -            for aci =idx:-1:idy
96 -                list(count) =  dct_list(aci,iy);
97 -                iy=iy+1;
98 -                count=count+1;
99 -                 laci=aci;
100 -           end
```

모드 체인지로 바뀐 루프는 이전에 했던 반대의 increase, decrease 모드에 대한 반대를 수행함 여기서는 8x8크기에 대한 지그제그 스캐닝만을 고려하였음 위와는 반대로 이미 x,y 8 8 까지 온 상태이므로 일부러 x의 값을 2로 잡아준다음에 increase할때는 2:8까지 y는 그 반대로 8:2까지 이런식으로 3 8, 4 8, 5 8 로 8 8 로 점점 다가간다. 하다보면 8 을 넘을때도 있는데 그때마다 8 로 다시 보정해줍니다.

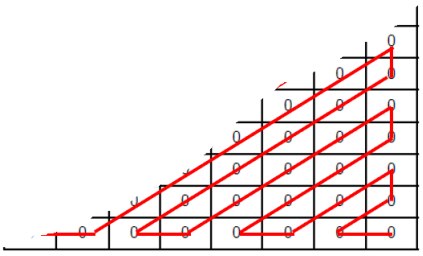
```
101 –            if(idy>8) idy=8; end
102 –                tmp = idx;
103 –                idx = idy+1;
104 –                idy = 8;
105 –              decrease = true;
106 –              increase = false;
107
108 –          end
109 –      end
110 –      end
111 –      zz=list;
112 –      end
```

루프를 쭉 돌다가

```
[chkx,chky] =  size(list);
if(chky==64) break
end
```

를 만족하면 즉 8,8까지 왔을 시에 chky값은 64가됩니다 그러면 break로 무한루프를 종료합니다.

1차원 리스트기 때문에 chkx는 무조건 1, 원소가 추가될 때, 마다 chky의 크기가 증가합니다.



에 대한 수행이었음. 물론 지금까지 한 연산에 대한 값은 list에 추가하고 그것을 리턴함.

```
[x,y] =size(img);
rx =(mod(x,8));
ry =(mod(y,8));
x_padded=0;
y_padded=0;

if(not (rx==0))
    x_padded=(8-rx);
    rx = floor(x_padded/2);
end
if(not (ry==0))
    y_padded=(8-ry);
    ry = floor(y_padded/2);
end

pad_img = zeros(x+x_padded,y+y_padded);

if((x_padded==0)&&(y_padded==0))
    pad_img = img;
elseif((x_padded~=0&&y_padded~=0))
    pad_img(1:x,1:y) = uint8(img);
elseif(x_padded~=0)
    pad_img(1:x,1:y) = uint8(img);
elseif(y_padded~=0)
    pad_img(1:x,1:y) = uint8(img);
end
```

이미지를 입력받았는데 8로 안눠질경우 8로 패딩하는 과정임

```
% Subtract 128
s_img =  double(pad_img) - 128;

DCT_table = ([[16,11,10,16,24,40,51,61];
              [12,12,14,19,26,58,60,55];
              [14,13,16,24,40,57,69,56];
              [14,17,22,29,51,87,80,62];
              [18,22,29,51,87,80,51,62];
              [24,35,55,64,81,104,113,92];
              [49,64,78,87,103,121,120,101];
              [72,92,95,98,112,100,103,99];]);
% Apply DCT
[px,py] = size(s_img);
step_x = px/8; step_y = py/8;
F_uv = zeros(64,64);
```

128빼고 dct테이블 미리정의함.

```
for i = 1:step_x:px
    u= floor(i/step_x)+1;
    for j= 1:step_y:py
        v= floor(j/step_y)+1;
        for lx = i:step_x+i-1;
            for ly = j :step_y+j-1;
                this_sum = zeros(8,8);
                for ix = 0:7
                    for iy = 0:7
                     this_sum(ix+1,iy+1) = this_sum(ix+1,iy+1) +cos((2*(ix+lx-1)+1)*(u-1)*pi/(2*8))*cos((2*(iy+ly-1)+1)*(v-1)*pi/(2*8));
                    end
                end
                F_uv((u-1)*8+1:(u)*8,(v-1)*8+1:(v)*8)=set_C(u,8)*set_C(v,8)*this_sum;
            end
        end
    end
end
```

Dct를 씌우기위한 필터를 만듬

```
DCT_res = zeros(px,py);
for ix = 1:8:px
    for iy = 1:8:py
        this_sum = zeros(8,8);
        for fi  = 1:8:64
            for fj = 1:8:64
            this_sum =this_sum+s_img(ix:ix+7,iy:iy+7).*F_uv(fi:fi+7,fj:fj+7);
            end
        end
        DCT_res(ix:ix+7,iy:iy+7) = round(this_sum./DCT_table);
    end
end
```

필터를 씌우는 동시에 디시티 테이블로 합에 대한 값을 round로 나눔

- -76 = 52 – 128

2. DCT
3. Quantization + Thresholding
   - -26 = round[-415/16]
   - -3 = round[-29/11]

이과정을 한번에 수행한 것임.

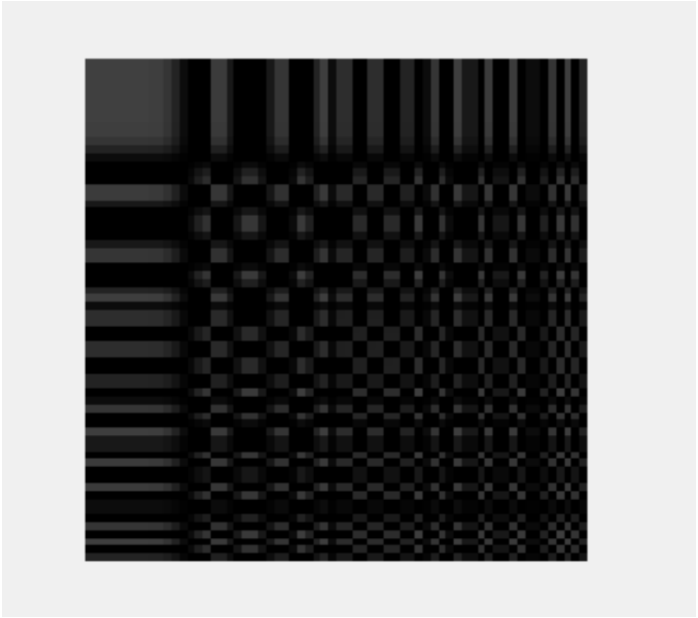
```
list_zigzag= {};
% Zigzag scanning
aa=1;
for i = 1: 8:px
    for j = 1:8:py
    [no0_list,index] = get_zero_idx(make_zigzag(DCT_res(i:i+7,j:j+7)));
    list_zigzag{aa} = no0_list;
    aa=aa+1;
    list_zigzag{aa} = index;
    aa=aa+1;
    end
end
```

```
zigzag= list_zigzag;
end
```

EOB를 표현하기 위해 가장 적절한 것이 메트랩의 배열 {}이라 판단하고 { 리스트, 하나의 int값, double, 문자열 } 등 파이썬의 리스트같이 여러 타입의 변수들을 원소로 닮을 수 잇는 판단때문에 메트랩 배열을 씀 한곳엔 EOB 바로옆엔 언제부터 0이 존재하는가에 대한 인덱스 값을 저장하도록 함.

구현 결과

8x8 DCT 필터



리스트

열 2023 ~ 2028

[1x27 double]	[27]	[1x21 double]	[21]	[1x16 double]	[16]

열 2029 ~ 2034

[1x23 double]	[23]	[1x21 double]	[21]	[1x43 double]	[43]

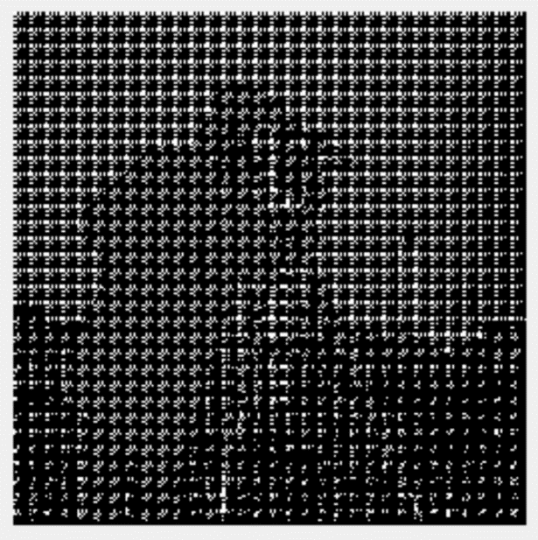
열 2035 ~ 2040

[1x39 double]	[39]	[1x25 double]	[25]	[1x25 double]	[25]

열 2041 ~ 2046

[1x23 double]	[23]	[1x43 double]	[43]	[1x29 double]	[29]

디시티 결과:



2. 느낀 점

:: 구현하면서 느낀 점, 어려웠던 점, 혹은 설명이 필요하다고 느낀 부분

아무리해도 디시티는 구현이 어려워서 디시티빼고는 다 했습니다.