1. Project 개요

1.1 Project 개발 필요성

1.2 Project 개발 배경 또는 동향

image는 여러 색상의 집합인데 우리는 각 image들에 있어서 어떠한 색상이 어느정도의 비중을 차지하고 있는지를 알고 싶었다. 육안으로는 정확한 비중을 판단하는 데에 한계가 있다고 생각했다. 또한 인터넷에 image를 등록하면 색상을 추출해주는 사이트는 상당히 많지만 우리는 단순히 색상을 얻는 것으로 끝나는 것이 아니라 한 가지 색상만을 사용해 image를 재구성하고 싶었다. 따라서 필터링을 통해 image 내사용 빈도가 높은 색상부터 나타내고, 더 나아가 하나의 색상만을 추출해 image를 재구성하는 것을 목표로 하였다.

1.3 기본적인 동작 및 기능

1.4 Project 개발 방법

a. 참고 MATLAB project 출처

https://kr.mathworks.com/matlabcentral/fileexchange/49898-image-color-filtering
https://kr.mathworks.com/matlabcentral/fileexchange/69538-image2palette-simple-k-means-color-clustering

b. 수정 및 개선 필요성, 범위 및 방법

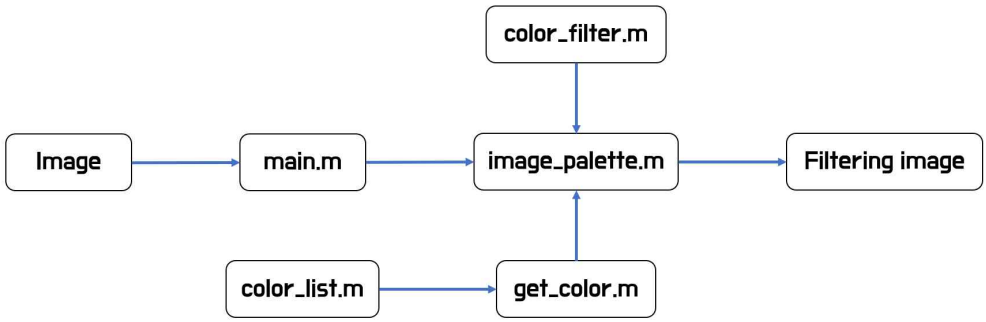
색상 범주를 조금 더 구체화할 필요가 있었고 나누어져있는 6가지의 색상들은 그 색상으로 정해진 명확한 이유가 없다고 생각했다. 따라서 우리는 hsv 원그래프를 활용하여 R,G,B와 RGB중 하나에 포함시키기 애매하다고 생각한 YELLOW로 총 4가지 색상으로 분류하여 각각의 색깔별로 추출할 수 있도록 설정하였다.

c. Project에서 본인들의 기여도 %로 측정 및 근거 제시

이름	역할	상세내용	기여도
이건욱	알고리즘 분석	알고리즘 및 블록다이어그램 기술	33%
김승준	코드 분석	각 함수 기능 설명	33%
신창우	코드 구현	색상 범주별 추출 코드 작성	33%

2. Project 수행 내용

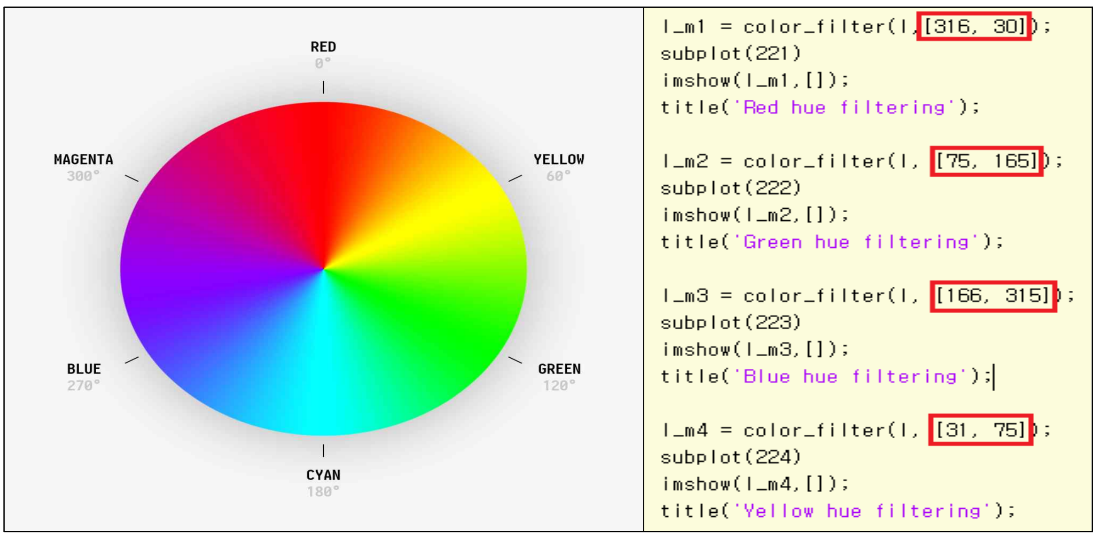
2.1 알고리즘 설명



위 그림은 우리가 원래의 코드를 역할별로 스크립트를 나눈 것을 보여준다. 위 그림에서는 main 에서 image를 읽어와 image_palette 함수에서 image를 구성하고 있는 색상 요소들을 구별하고 color_filter 함수에서는 image를 특정한 색상으로 필터링하여 재구성된 모습을 새로운 창에 표시해준다.

get_color 함수와 color_list 함수는 서로 상호 작용하여 추출된 색상을 RGB color list와 일치하는 색상의 이름을 보여주는 데 사용되는 모습을 보여준다.

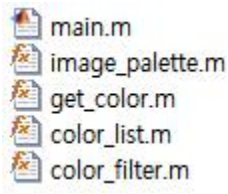
다음은 우리가 언급한 RGB 와 Yellow에 대한 hsv 원그래프에서 각도를 구하기 위해 찾아냈던 원그래프와 색깔별로 각도를 맞춘 코드이다.



여러 번의 시도 끝에 위의 코드에 표시된 각도들이 우리가 원했던 결과를 얻는데 가장 적합한 각도들이였고 270~330도 정도의 영역들은 R,B영역에 나누어 넣을 수 있었지만 30~90도 정도의 영역인 YELLOW는 RGB에 포함시키기 애매하다고 판단하여 따로 분리하는 판단을 하였다.

2.2 MATLAB 구현

– 함수별 설명



▶ **main.m**: image 파일에서 추출하고 싶은 색 성분 개수를 Cluster의 개수를 선언하여 조정 가능하다.
불러온 image를 image_palette.m 함수 호출하여 넘겨준다.

```
%% 이미지를 읽어 image_palette 함수 호출
nCluster = 5;           % 추출하고 싶은 색 성분 개수
fname ='test.png';      % 이미지 파일

image_palette( nCluster, fname );
```

▶ **color_list.m**: RGB 표에 의한 색상 정보

```
function [rgb, labels] = color_list()
labels = {...
    'Pink'
    'LightPink'
    'HotPink'
    'DeepPink'
    'PaleVioletRed'
    'MediumVioletRed'
    'LightSalmon'
    'Salmon'
    'DarkSalmon'
    'LightCoral'
    'IndianRed'
    'Crimson'
    'Firebrick'
    'DarkRed'
    'Red'
    'OrangeRed'

rgb = [255  192 203
       255 182 193
       255 105 180
       255 20  147
       219 112 147
       199 21  133
       255 160 122
       250 128 114
       233 150 122
       240 128 128
       205 92  92
       220 20  60
       178 34  34
       139 0   0
       255 0   0
       255 69  0
       255 99  71
       255 127 80
       255 140 0
       255 165 0
       255 255 0
```

〈약 150개의 color 정보〉

▶ **get_color.m**: color_list.m으로부터 색상 이름 전달받아 추출에 사용될 컬러를 매칭

```
%% color_list.m으로부터 색상 이름 전달받음
function [matching_label, min_error] = get_color( input )
%입력 받을 RGB (0~255)
if size(input,1)>size(input,2)
    input=input';
end
[rgb, labels] = color_list();
input_rep = repmat( input, [size(rgb,1), 1]);
[min_error,matchIdx] = min( mean((input_rep-rgb).^2, 2) );
matching_label = labels{ matchIdx };
end
```

▶ **color_filter.m**: figure(2)에 사용되는 함수, HSV 범위를 이용하여 특정 색상 외의 나머지 색을 필터링

```
% 범위에 따라 마스크 부여
if(range(1) > range(2))
    mask = (I(:,:,1)>range(1) & (I(:,:,1)<=1)) + (I(:,:,1)<range(2) & (I(:,:,1)>=0));
else
    mask = (I(:,:,1)>range(1)) & (I(:,:,1)<range(2));
end

I(:,:,2) = mask .* I(:,:,2);
```

▶ **Image_palette.m**: main.m에서 image를 받아온 후 색상 추출

```
for clusterIdx = 1:nCluster                % 값이 작은 것부터 정렬해서 저장
    id = G==order(clusterIdx);
    LAB = ( img_parse(:, id) );
    mean_LAB = reshape( nanmean(LAB,2)', [1, 1, 3]);   % LAB 색상 저장

    % lab 색상을 rgb 색상표에 따라 이름 부여
    [color_label] = get_color(255*lab2rgb(mean(LAB,2)'));
    if plotOption
        subplot(2,nCluster,nCluster+clusterIdx);
        % LAB색상을 rgb 색상으로 변환후 표시
        imshow( lab2rgb( mean_LAB ) );
        title(['C_{' num2str(clusterIdx) '} (' num2str(round(100*p(order(clusterIdx))/100)) ' %)' ]);
        xlabel(color_label);
    end
end
```

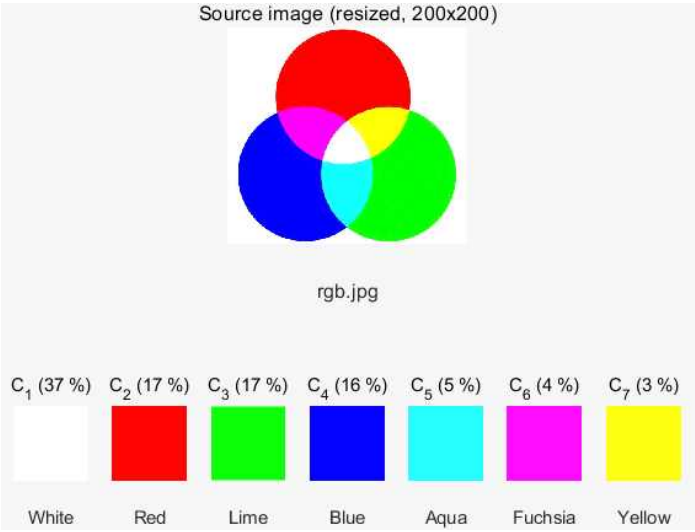
⇒ figure(1)에 image에서 추출된 색상을 Cluster의 개수에 따라 다르게 올린다.

```
I_m1 = color_filter(I,[316, 30]);
subplot(221)
imshow(I_m1,[]);
title('Red hue filtering');

I_m2 = color_filter(I, [75, 165]);
subplot(222)
imshow(I_m2,[]);
title('Green hue filtering');

I_m3 = color_filter(I, [166, 315]);
subplot(223)
imshow(I_m3,[]);
title('Blue hue filtering');

I_m4 = color_filter(I, [31, 75]);
subplot(224)
imshow(I_m4,[]);
title('Yellow hue filtering');
```

⇒ figure(2)에 image를 필터링하여 RED, GREEN, BLUE, YELLOW 영역을 나타낸다.

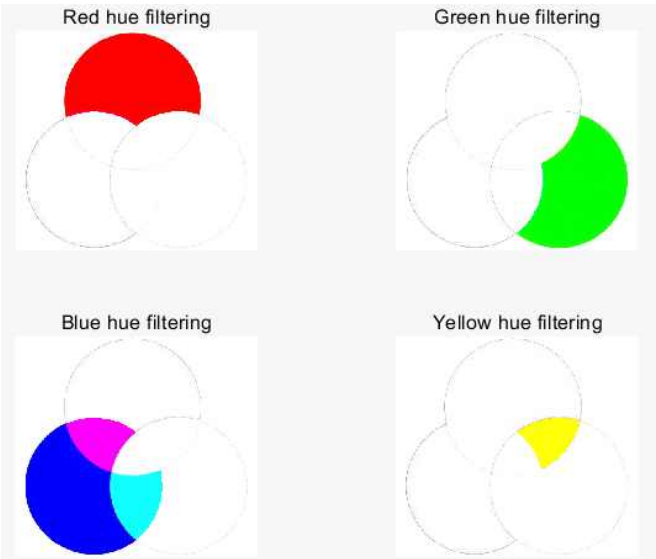
– 예상 출력값

① 지정된 image의 컬러 구성에 맞는 색상들이 추출



⇒ 일정한 비율로 색상이 추출된 것을 확인할 수 있다.

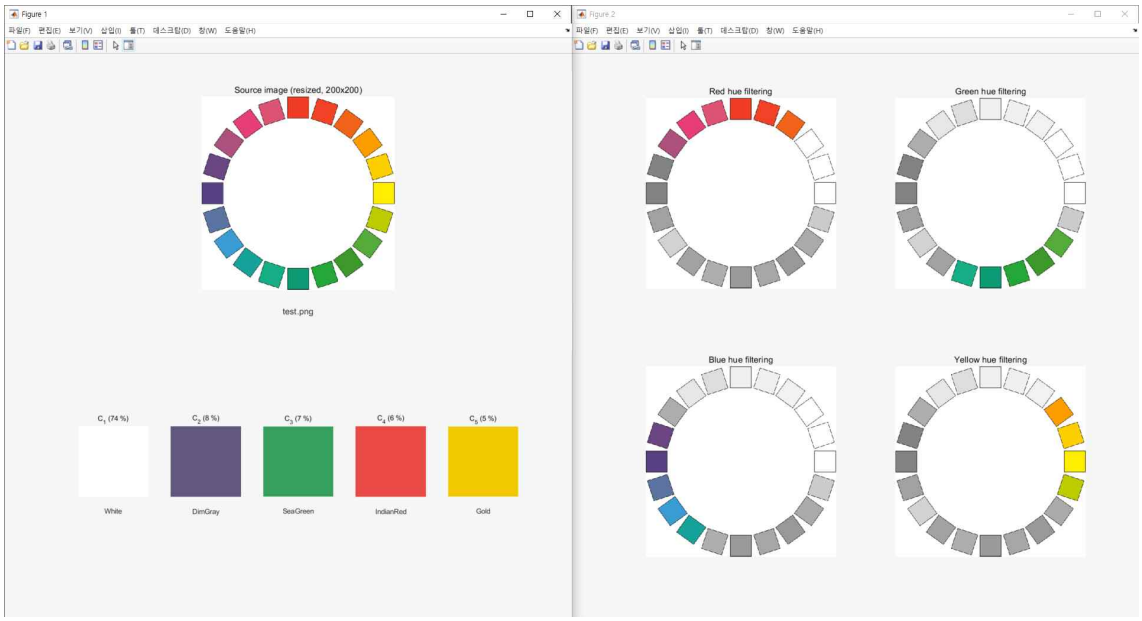
② 미리 지정한 HSV의 H 영역 범위를 이용,
특정 색상을 제외한 나머지 색상은 필터링하여 image 재구성



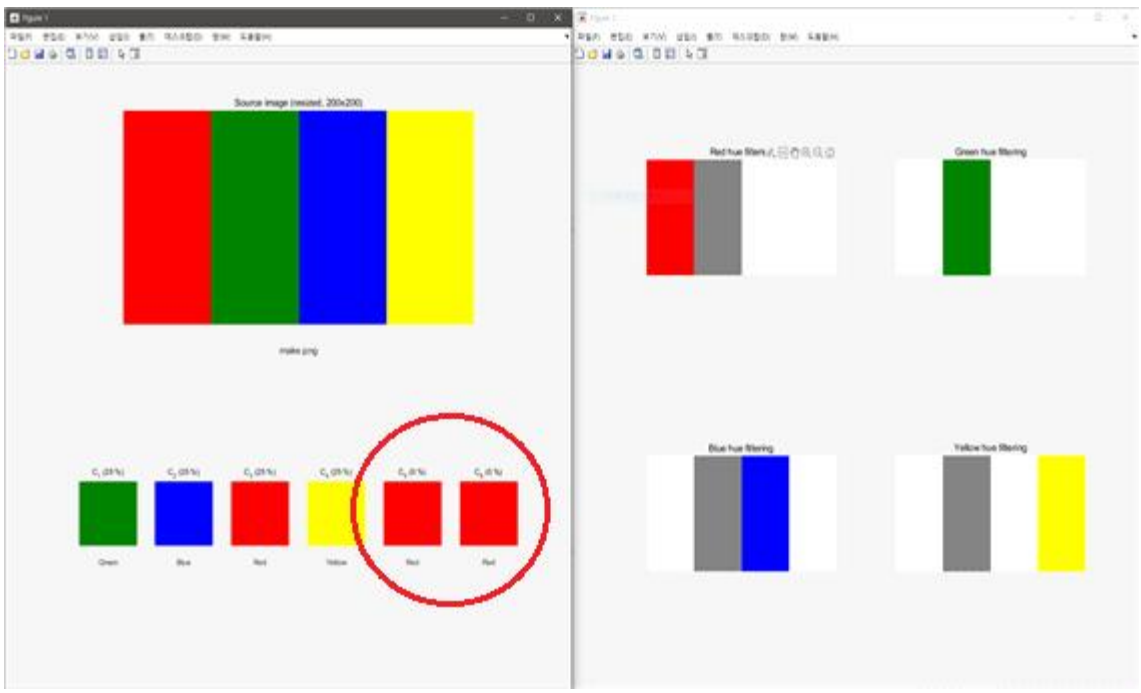
⇒ HSV(범위)와 RGB(지정된 값)는 색 구성 방법부터 다르기 때문에 양쪽 색상 호환 문제가 중요한 이슈다.

2.3 입력 및 출력

1. 색 구분이 뚜렷한 경우



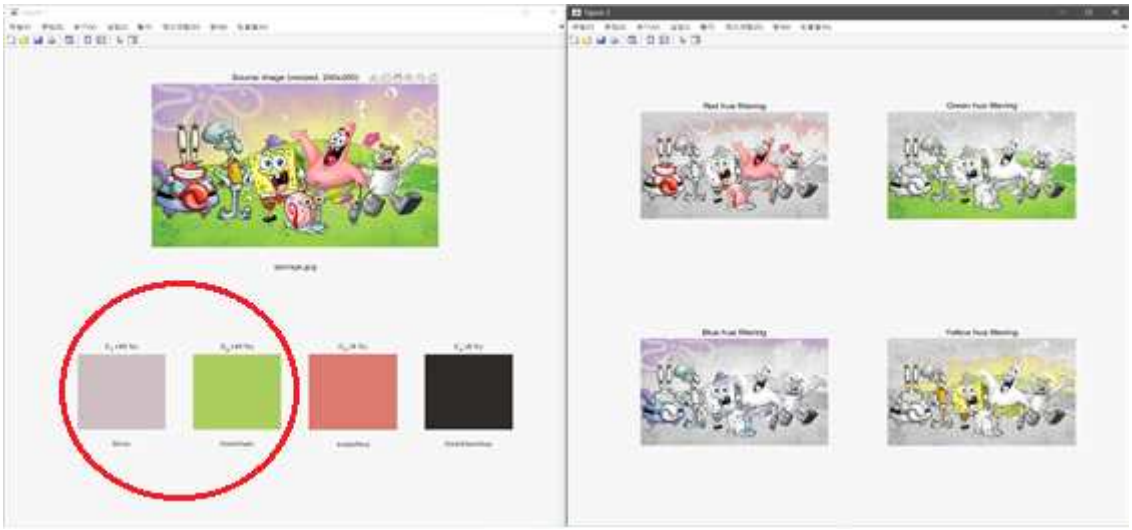
〈color chart. 1〉

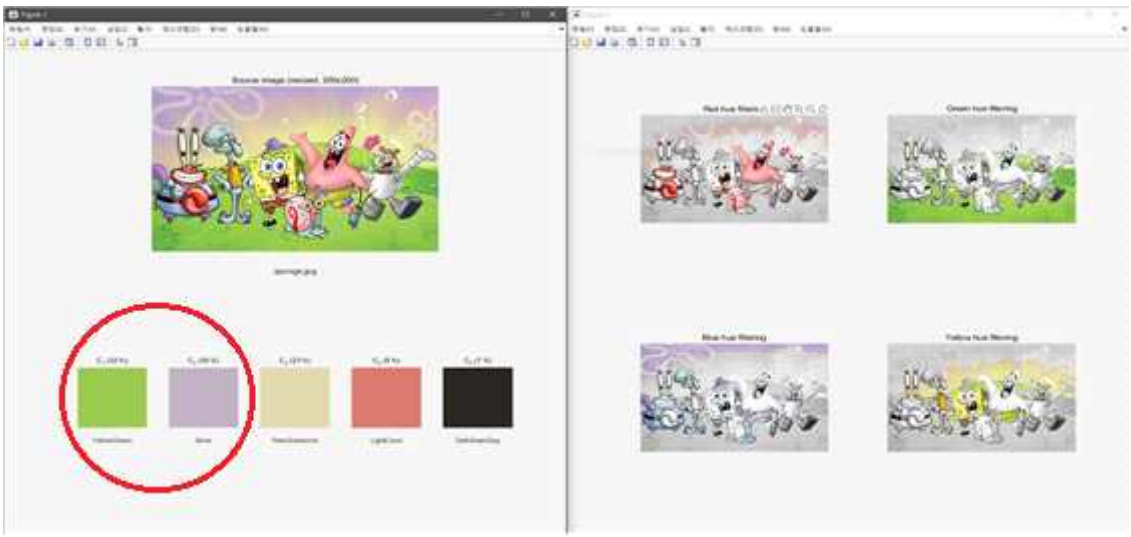


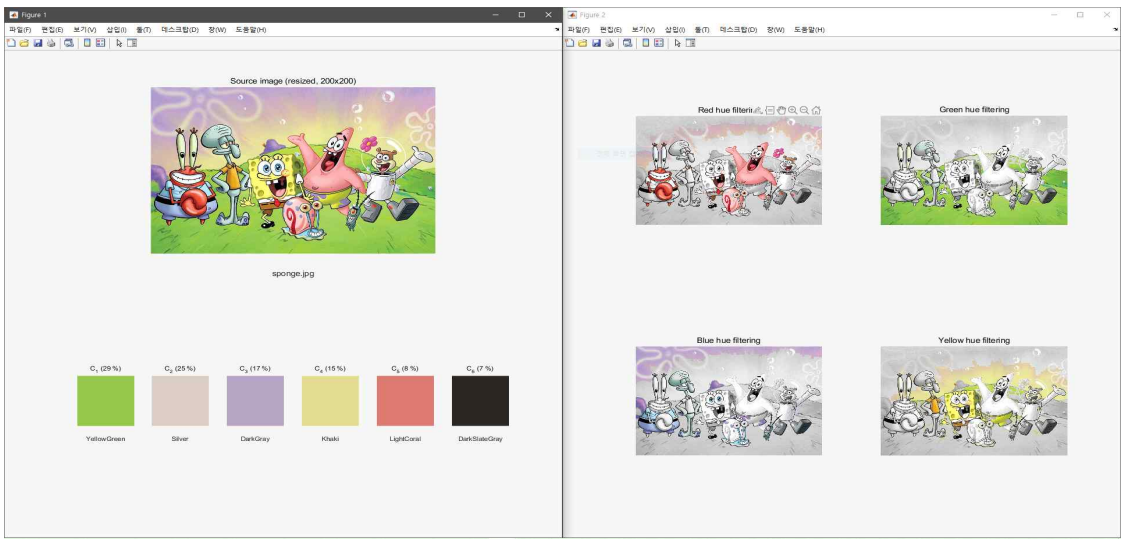
〈color chart. 2〉

2. 색 구분이 복잡한 경우

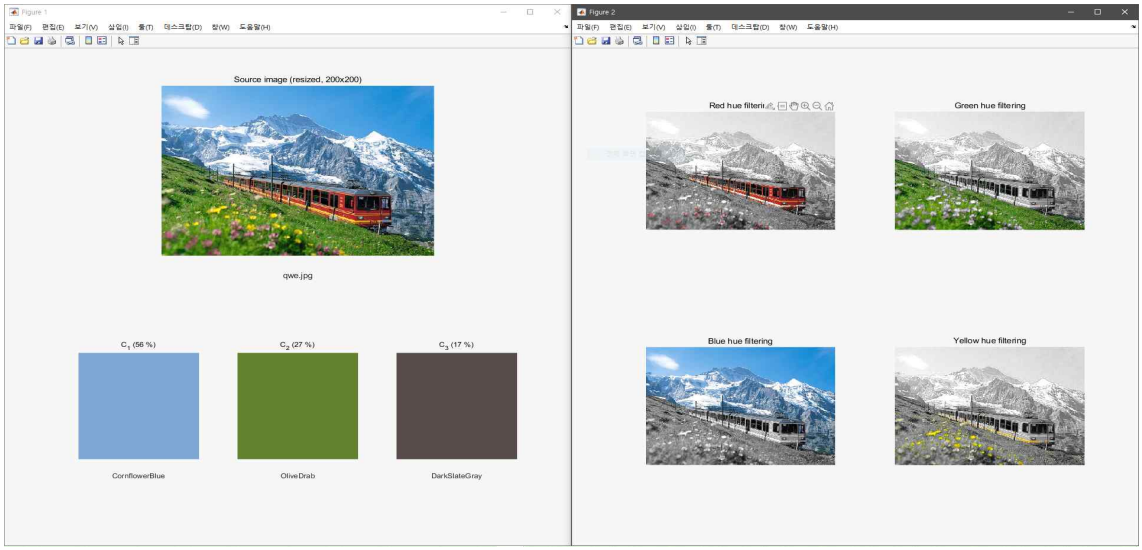
〈complex. 1〉

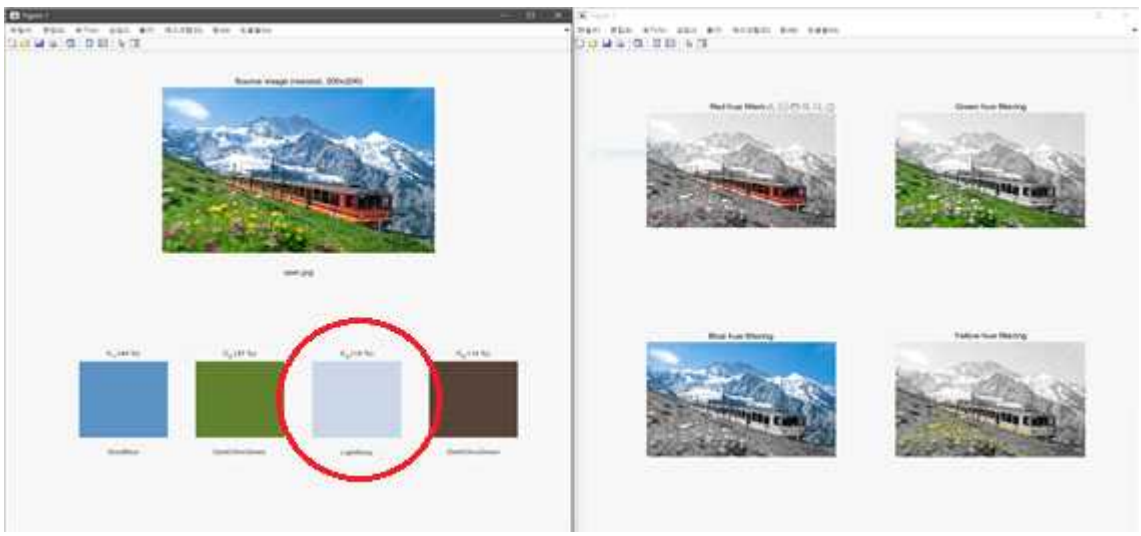


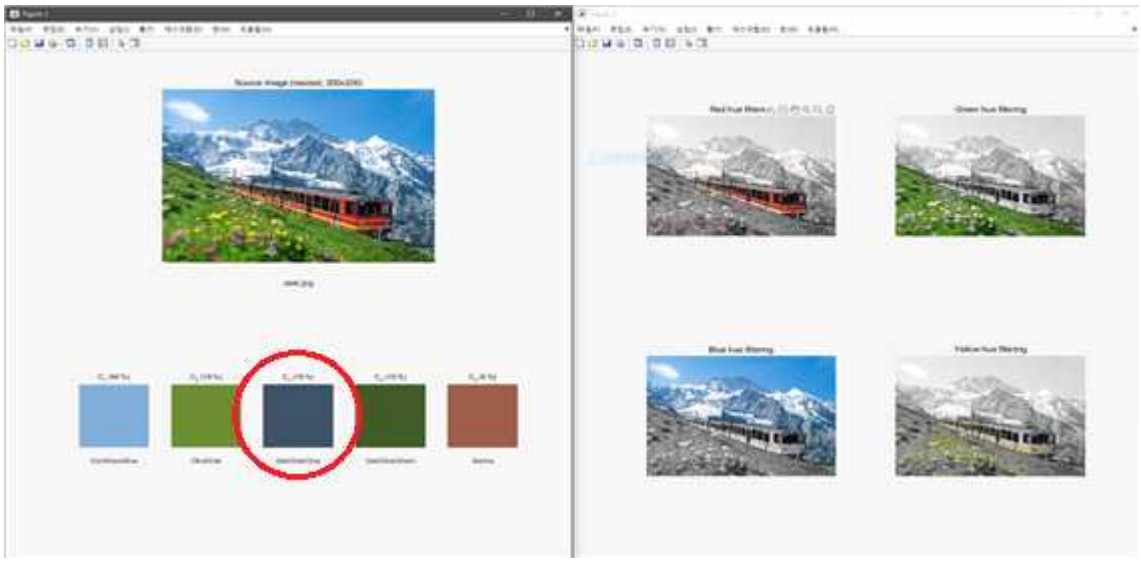




〈complex. 2〉


Source image (resized, 200x200)
qwe.jpg
C1 (56 %)
C2 (27 %)
C3 (17 %)
CornflowerBlue
OliveDrab
DarkSlateGray
Red hue filtering
Green hue filtering
Blue hue filtering
Yellow hue filtering





☞ 결과물에 따른 알고리즘 성능 분석

① 색 구분이 뚜렷한 경우

색 구분이 뚜렷한 사진을 이용해 필터링을 해본 결과 〈color chart. 1〉과 같은 경우는 배경색인 흰색을 제외하고 원하는 필터값인 RED, GREEN, BLUE, YELLOW계열의 색상을 추출하는 것을 볼 수 있다. 색 구분이 뚜렷한 사진을 필터링한 결과로 다른 사진도 적절한 영역을 필터링 할 수 있다는 추론을 할 수 있다.

두 번째로, 직접 만든 〈color chart. 2〉처럼 색이 잘 구분되어 있는 사진일수록 정확도가 높았다. 성능테스트를 위해 직접 각각의 RED, GREEN, BLUE, YELLOW의 RGB를 맞춘 4가지 색상표를 만들어 테스트 해보았고 cluster의 값이 사진이 가지고 있는 경우를 테스트해보기 위해 4개가 아닌 6개의 cluster 개수를 지정하여 테스트를 해보았다. 그 결과 6개 중 4가지 색상을 제외한 2개의 cluster는 아무런 의미 없는 값이 나오는 것을 확인할 수 있었다.

② 여러 색들이 모여 있어 색 구분이 복잡한 경우

색 구분이 복잡할 경우의 image도 이용해 필터링을 해보았다. 이러한 사진들은 색 구분이 어려워 색상을 색 구분이 뚜렷한 image에 비해 성능이 낮을 것이라 예상했다. 우리가 예상했던 대로 색 구분이 뚜렷한 image에 비해 색상의 %에 대한 cluster에서 문제점이 발견되었다.
〈complex. 1〉의 경우 cluster 4개와 5개의 경우에서 빨간원에 표시된 색깔인 YellowGreen과 Silver의 순위가 바뀐 것을 확인할 수 있었다.
〈complex. 2〉의 경우에는 cluster의 개수가 증가할수록 중간에 색상이 하나씩 추가됐지만 결과적으로 cluster의 개수가 증가할수록 우리가 예상하는 결과와 비슷해진다는 것을 알게되었다.

※ 종합해보면 색 구분이 뚜렷하거나 cluster 수를 증가시킨다면 성능이 향상된다는 것을 예측할 수 있었다.

3. 참여 인원별 역할 및 수행 소감

▶ 이건욱

대학교에 와서 C, JAVA, Python 등 여러 언어들을 공부해보았지만 신호처리라는 과목을 들으면서 처음으로 MATLAB이라는 언어를 배우게 되었다. 교수님께서 간단한 예제들과 실용적인 몇 가지 문법들을 알려주실때는 되게 재밌겠다고 생각했고 프로젝트가 있단 말씀을 하셨을때도 재밌을 거란 기대가 반 이상을 차지했다. 하지만 막상 여러 가지 Project들을 찾아보고 code를 분석해보니 기본이 없어 Project가 버겁게 느껴졌었다. 하지만 이해되지 않는 것을 하나하나 검색해보고 교수님께서 알려주신 명령어 중 하나인 help를 사용하여 알아내고 직접 부딪히다 보니 결국 이해할 수 있게 되었고 나의 흥미는 다시 살아나기 시작했던 것 같다. 매번 느끼지만 이게 우리 전공의 묘미가 아닐까 싶다. 결과적으로 이렇게 처음엔 버겁다 느낀 프로젝트로 인해서 나에게 또 한가지의 스택이 쌓이고 전공공부에 대한 흥미가 늘어가는 것 같아 정말 유익한 프로젝트이자 정말 유익한 강의들이였다고 생각한다.

▶ 김승준

지난 학기 멀티미디어 시스템 수업을 들으면서 멀티미디어의 기본 지식을 이해하여서 이번 신호처리 과목이 크게 낯설지는 않았다고 생각합니다.
MATLAB 수업을 듣고 프로젝트를 진행하면서 처음 써보는 프로그램이라 사용하는데 조금 어려움을 겪긴 했었지만, 팀원들과 함께 찾아보고 공부하면서 실력이 조금씩 늘어가는 게 재미있었고, 팀 프로젝트를 완성하는 원동력이 되었습니다.
팀 프로젝트를 하기 앞서서 어떤 주제로 시작을 해야 좋을까 선택하는데 시간이 꽤 걸렸습니다. 교수님이 추천해 주신 Mathwork 사이트와 구글, 깃허브 등 서칭을 하면서 많은 양의 자료를 보았습니다. 그중 마음에 드는 예제가 몇몇 있었지만, 난이도가 높은 코드가 섞여있어 이해하기 어려워 진행에 차질을 겪었습니다. 팀원과 많은 고난 끝에 선택한 것이 이번 프로젝트인 Image color filtering입니다.
range를 맞추는 과정에서 RGB 표와 HSV 표를 보며 여러 사진의 결과를 비교하며 RED, GREEN, BLUE, YELLOW 네 영역을 원하는 만큼 추출 할 수 있어서 뿌듯했습니다.

▶ 신창우

신호처리 과목을 공부하면서 MATLAB을 처음 접했다. 이미지 프로세싱과 머신러닝까지 할 수 있다는 점이 매우 흥미로웠다. 기존에는 C, C++, JAVA, Python등 다른 언어들을 배웠었는데 MATLAB은 굉장히 직관적이라는 점이 Python과 비슷하다. 최근 Python을 많이 다루고 있어서 MATLAB을 접근하는 것은 어렵지 않았다. 하지만 다른 점도 있기에 매우 헷갈리는 것도 많았다. 멀티미디어 시스템 과목을 2년전에 들어서 이번 신호처리 과목은 다 처음 배우는 느낌이 많이 들지만 오히려 새로 배우는 것이라는 점이 더 흥미롭고 재미있게 만들어줬다. 프로젝트에서 코드를 재구성하는 역할을 맡았다. 처음 프로젝트의 기반인 코드를 봤을 때 엄청 어렵고 복잡하다고 생각이 들었다. 하지만 직관적인 언어이기 때문에 읽으면 읽을수록 코드를 이해하기 쉬워졌다. 처음 보는 함수가 있으면 help를 통해 무슨 함수인지 알아보면서 익혔다. 그래서 어떤 코드가 어떤 역할을 하는

지 쉽게 파악할 수 있게 되었고, 프로젝트에 어떤 코드가 필요 없고 외부함수 구성을 쉽게 진행할 수 있었다. 항상 Python을 주로 써왔기 때문에 MATLAB을 이해하기 쉬웠다. 앞으로 머신러닝을 할 때 Python뿐만 아니라 MATLAB으로도 해야겠다고 생각이 들었다.

4. Project 후기

대학을 다니면서 MATLAB을 처음 접했다. 기본 사용법만 배우고 프로젝트를 진행하는 것이 우리에겐 상당히 어려운 일이었다. 그래서 다소 긴장한 상태에서 프로젝트를 시작했다. 처음에는 face recognition을 주제로 프로젝트를 진행하려고 했다. 하지만 코딩 이해에 어려움이 있어 우리 팀은 많은 고민 끝에 image filtering으로 주제를 바꾸었다.

주제를 바꾸고 처음 접한 것이 특정 색을 제외한 나머지 색은 흑백으로 처리하는 알고리즘을 접했다. 그렇게 어려운 난이도가 아니기에, 다른 image processing 알고리즘과 접목시키고 싶었다. Mathwork 사이트에서 계속 검색해본 결과 사진에서 사용된 색을 추출하여 추출한 색을 가지고 image filtering을 하자는 의견이 나왔다. 이것을 토대로 시작을 해서 현재의 결과물을 만들게 되었다. 의견은 좋았지만 두 개의 코딩을 합치는 것 자체가 어려웠고 계속적인 시행착오가 있었다. 하지만 팀원들끼리 합심하여 성공했고 처음 접한 프로그램에서 이정도의 결과물을 만들었다는 사실에 굉장히 만족스러웠다. 또한 MATLAB은 오픈소스가 다양하고, 이를 사용할 경우 다른 프로그램에 비해 상대적으로 오류가 적었다. 따라서 처음 접하는 프로그램임에도 불구하고 조금 더 효율적으로 프로젝트를 성공시킬 수 있었다.

5. Appendix (Project MATLAB source code)

※ code를 실행하기 전, MATLAB Image Processing ToolBox설치 필요

≪ main.m ≫

```
%% image를 읽어 image_palette 함수 호출
nCluster = 6;            % 추출하고 싶은 색 성분 개수
fname ='color_chart.png';       % image 파일

image_palette( nCluster, fname );
```

≪ get_color.m ≫

```
%% color_list.m으로부터 색상 이름 전달받음
function [matching_label, min_error] = get_color( input )

if size(input,1)>size(input,2)
    input=input';
end
[rgb, labels] = color_list();
input_rep = repmat( input, [size(rgb,1), 1]);
[min_error,matchIdx] = min( mean((input_rep-rgb).^2, 2) );
matching_label = labels{ matchIdx };
end
```

≪ color_filter.m ≫

```
%% 지정한 HSV의 H 범위를 이용하여 특정 색상 외의 나머지 색 필터링
function I = color_filter(image, range)
    % RGB를 HSV로 변환
    I = rgb2hsv(image);

    % 0 ~ 1 사이 범위로 normalization
    range = range./360;

    % 마스크 생성
    if(size(range,1) > 1), error('Error. Range matriz has too many rows.'); end
    if(size(range,2) > 2), error('Error. Range matriz has too many columns.'); end

    % 범위에 따라 마스크 부여
    if(range(1) > range(2)) % 붉은 색조 케이스
        mask = (I(:,:,1)>range(1) & (I(:,:,1)<=1)) + (I(:,:,1)<range(2) & (I(:,:,1)>=0));
    else % 일반 케이스
        mask = (I(:,:,1)>range(1)) & (I(:,:,1)<range(2));
    end
    % 채도는 마스크에 따라 수정
    I(:,:,2) = mask .* I(:,:,2);

    % HSV를 다시 RGB로 변환
    I = hsv2rgb(I);
end
```

≪ image_palette.m ≫

```
%% main.m에서 받은 image를 바탕으로 색상 추출
function image_palette(nCluster, fname, plotOption)

  % 함수 인자가 3개 미만이면 plotOption 값 true
if nargin<3, plotOption = true; end
sz = [200 200];                    % 사진 크기 초기화

%% Load image file
img = rgb2lab(imread(fname));       % image 파일 읽어오기
img = single(img);
```

```
L = img(:,:,1);                          % image의 lab 값을 각각 변수에 저장
a = img(:,:,2);
b = img(:,:,3);

%% Visualize image
if plotOption                            % 불러온 image 표시
    figure(1);
    subplot(2,1,1);
    imshow( lab2rgb( img ) );
    title(['Source image (resized, ' num2str(sz(1)) 'x' num2str(sz(2)) ')'])
    xlabel(fname);
end

%% Parse and visualize Lab values
img_parse = [];
img_parse(1,:) = L(:);
img_parse(2,:) = a(:);
img_parse(3,:) = b(:);


%% K-means Clustering
[G] = kmeans( img_parse', nCluster);    % k-평균 군집화 수행
p = [];
for clusterIdx = 1:nCluster                % 퍼센트 할당
    p(clusterIdx) = 100*length(find(G==clusterIdx)) / length(G);
end
[~,order]=sortrows(p'); order=order(end:-1:1);

% Palette visualization
for clusterIdx = 1:nCluster                % 값이 작은 것부터 정렬해서 저장
    id = G==order(clusterIdx);
    LAB = ( img_parse(:, id) );
    mean_LAB = reshape( nanmean(LAB,2)', [1, 1, 3]);    % LAB 색상 저장

     % lab 색상을 rgb 색상표에 따라 이름 부여
    [color_label] = get_color(255*lab2rgb(mean(LAB,2)'));
    if plotOption
        subplot(2,nCluster,nCluster+clusterIdx);
        % LAB색상을 rgb 색상으로 변환후 표시
        imshow( lab2rgb( mean_LAB ) );
```

```
        title(['C_{' num2str(clusterIdx) '} (' num2str(round(100*p(order(clusterIdx))/100)) ' %)' ]);
        xlabel(color_label);
    end
end


%% 6가지 색상별로 추출하여 image 재구성
if plotOption
    figure(2)
            I = imread(fname);
            I = im2double(I); % image I를 배정밀도로 변환

            I_m1 = color_filter(I, [316, 30]);
            subplot(221)
            imshow(I_m1,[]);
            title('Red hue filtering');

            I_m2 = color_filter(I, [75, 165]);
            subplot(222)
            imshow(I_m2,[]);
            title('Green hue filtering');

            I_m3 = color_filter(I, [166, 315]);
            subplot(223)
            imshow(I_m3,[]);
            title('Blue hue filtering');

            I_m4 = color_filter(I, [31, 75]);
            subplot(224)
            imshow(I_m4,[]);
            title('Yellow hue filtering');

end
end
```

≪ color_list.m ≫

```
function [rgb, labels] = color_list()
labels = {...
    'Pink'
    'LightPink'
    'HotPink'
    'DeepPink'
    'PaleVioletRed'
    'MediumVioletRed'
    'LightSalmon'
    'Salmon'
    'DarkSalmon'
    'LightCoral'
    'IndianRed'
    'Crimson'
    'Firebrick'
    'DarkRed'
    'Red'
    'OrangeRed'
    'Tomato'
    'Coral'
    'DarkOrange'
    'Orange'
    'Yellow'
    'LightYellow'
    'LemonChiffon'
    'LightGoldenrodYellow'
    'PapayaWhip'
    'Moccasin'
    'PeachPuff'
    'PaleGoldenrod'
    'Khaki'
    'DarkKhaki'
    'Gold'
    'Cornsilk'
    'BlanchedAlmond'
    'Bisque'
    'NavajoWhite'
    'Wheat'
    'Burlywood'
    'Tan'
    'RosyBrown'
    'SandyBrown'
    'Goldenrod'
    'DarkGoldenrod'
    'Peru'
    'Chocolate'
    'SaddleBrown'
    'Sienna'
```

```
'Brown'
'Maroon'
'DarkOliveGreen'
'Olive'
'OliveDrab'
'YellowGreen'
'LimeGreen'
'Lime'
'LawnGreen'
'Chartreuse'
'GreenYellow'
'SpringGreen'
'MediumSpringGreen'
'LightGreen'
'PaleGreen'
'DarkSeaGreen'
'MediumAquamarine'
'MediumSeaGreen'
'SeaGreen'
'ForestGreen'
'Green'
'DarkGreen'
'Aqua'
'Cyan'
'LightCyan'
'PaleTurquoise'
'Aquamarine'
'Turquoise'
'MediumTurquoise'
'DarkTurquoise'
'LightSeaGreen'
'CadetBlue'
'DarkCyan'
'Teal'
'LightSteelBlue'
'PowderBlue'
'LightBlue'
'SkyBlue'
'LightSkyBlue'
'DeepSkyBlue'
'DodgerBlue'
'CornflowerBlue'
'SteelBlue'
'RoyalBlue'
'Blue'
'MediumBlue'
'DarkBlue'
'Navy'
'MidnightBlue'
'Lavender'
```

```
    'Thistle'
    'Plum'
    'Violet'
    'Orchid'
    'Fuchsia'
    'Magenta'
    'MediumOrchid'
    'MediumPurple'
    'BlueViolet'
    'DarkViolet'
    'DarkOrchid'
    'DarkMagenta'
    'Purple'
    'Indigo'
    'DarkSlateBlue'
    'SlateBlue'
    'MediumSlateBlue'
    'White'
    'Snow'
    'Honeydew'
    'MintCream'
    'Azure'
    'AliceBlue'
    'GhostWhite'
    'WhiteSmoke'
    'Seashell'
    'Beige'
    'OldLace'
    'FloralWhite'
    'Ivory'
    'AntiqueWhite'
    'Linen'
    'LavenderBlush'
    'MistyRose'
    'Gainsboro'
    'LightGray'
    'Silver'
    'DarkGray'
    'Gray'
    'DimGray'
    'LightSlateGray'
    'SlateGray'
    'DarkSlateGray'
    'Black'
    };
rgb = [255   192   203
    255   182   193
    255   105   180
    255   20   147
    219   112   147
```

```
199  21  133
255  160  122
250  128  114
233  150  122
240  128  128
205  92  92
220  20  60
178  34  34
139  0  0
255  0  0
255  69  0
255  99  71
255  127  80
255  140  0
255  165  0
255  255  0
255  255  224
255  250  205
250  250  210
255  239  213
255  228  181
255  218  185
238  232  170
240  230  140
189  183  107
255  215  0
255  248  220
255  235  205
255  228  196
255  222  173
245  222  179
222  184  135
210  180  140
188  143  143
244  164  96
218  165  32
184  134  11
205  133  63
210  105  30
139  69  19
160  82  45
165  42  42
128  0  0
85  107  47
128  128  0
107  142  35
154  205  50
50  205  50
0  255  0
124  252  0
```

```
127   255   0
173   255   47
0   255   127
0   250   154
144   238   144
152   251   152
143   188   143
102   205   170
60   179   113
46   139   87
34   139   34
0   128   0
0   100   0
0   255   255
0   255   255
224   255   255
175   238   238
127   255   212
64   224   208
72   209   204
0   206   209
32   178   170
95   158   160
0   139   139
0   128   128
176   196   222
176   224   230
173   216   230
135   206   235
135   206   250
0   191   255
30   144   255
100   149   237
70   130   180
65   105   225
0   0   255
0   0   205
0   0   139
0   0   128
25   25   112
230   230   250
216   191   216
221   160   221
238   130   238
218   112   214
255   0   255
255   0   255
186   85   211
147   112   219
138   43   226
```

```
    148    0    211
    153    50    204
    139    0    139
    128    0    128
    75    0    130
    72    61    139
    106    90    205
    123    104    238
    255    255    255
    255    250    250
    240    255    240
    245    255    250
    240    255    255
    240    248    255
    248    248    255
    245    245    245
    255    245    238
    245    245    220
    253    245    230
    255    250    240
    255    255    240
    250    235    215
    250    240    230
    255    240    245
    255    228    225
    220    220    220
    211    211    211
    192    192    193
    169    169    169
    128    128    128
    105    105    105
    119    136    153
    112    128    144
    47    79    79
    0    0    0
    ];
end
```