楊宗翰
101062142
2015/3/23

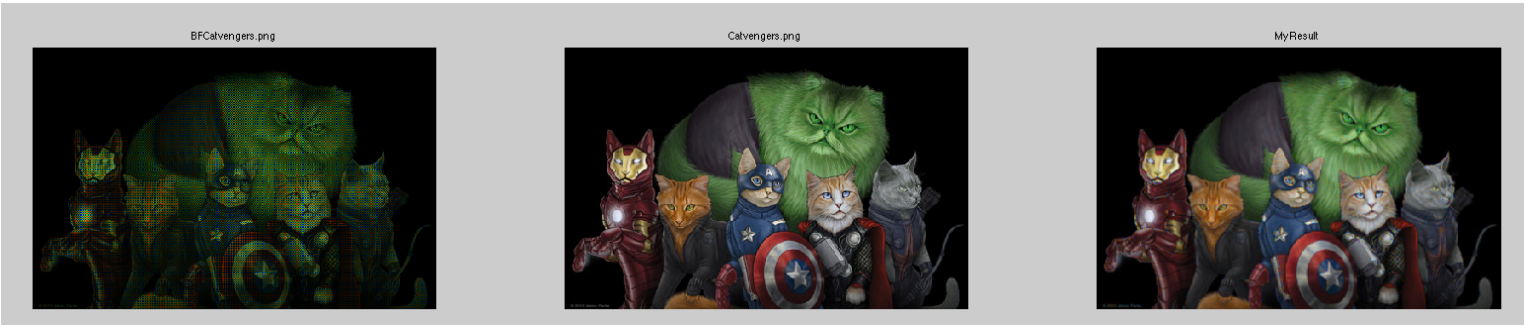
Introduction to Multimedia Homework 1 Report

• Problem 1 - Demosaicing

In this problem we aim to simulate the process of digitalizing a picture with component constraint. Because the component we use can only detect on color at each site, leaving 2 other channels blank, we want to recover the empty channels form its neighboring sites. The core of the solution is as below.

```
% nearest neighbor algorithm
for i = 2:w-1
    for j = 2:h-1
        for k = 1:3
            % we only care about the missing color
            if bffilter_img(i, j, k) == 0
                % get resonable range
                i1 = i-1; i2 = i+1;
                j1 = j-1; j2 = j+1;
                ori = bfcat_img(i1:i2, j1:j2, k); % filtered image
                fil = bffilter_img(i1:i2, j1:j2, k); % filter
                s =  ori .* fil;
                avg_color = sum(sum(s)) / sum(sum(fil));
                result_img(i, j, k) = avg_color;
            end
        end
    end
end
```

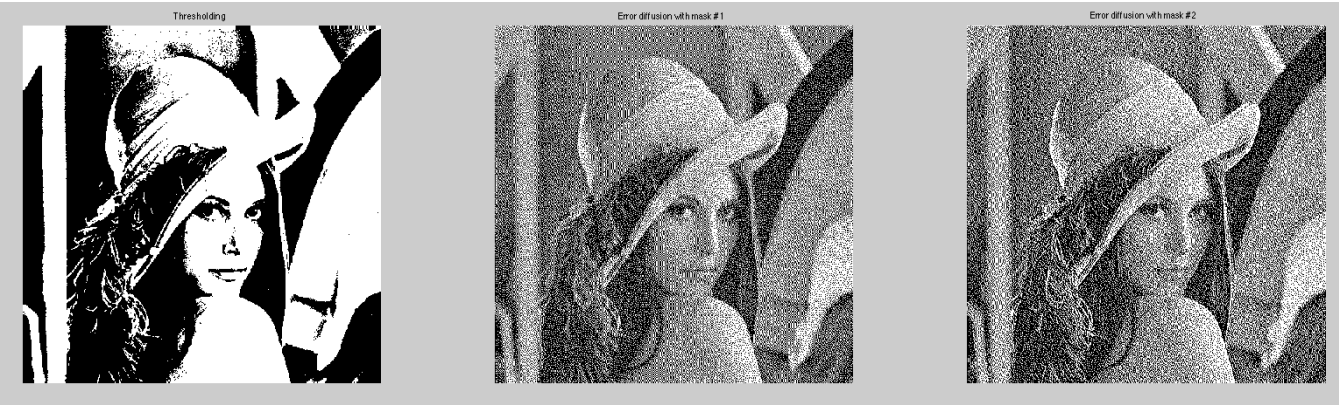
Here, I neglect the border so as to simplify my implementation. I enumerate each hollow sites and recover them from their neighbors. I find dot product very useful here, simply fetch a 3x3 sub-matrix of the image and dot them together we can calculate their average color at ease.



Original image and my result. This yields "*Peak-SNR value is 30.8207, SNR value is 18.6611*" — pic1.1

- # Problem 2 - Dithering

The task here is to transfer a grayscale image into a binary one. The core of the problem is to determine which is 1 which is 0, and we utilize thresholding and error diffusion dithering techniques to determine its value. As for the thresholding value, I choose to use the **average value of all pixels**. This is the very value we would instinctively choose.

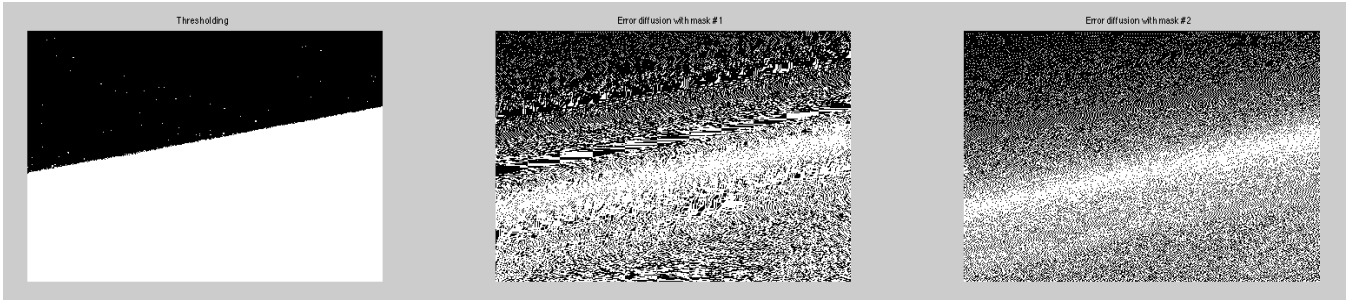


Snapshot of the result. — pic2.1

From pic2.1 we can easily notice that thresholding will result in a clear border of black and white. On the other hand, error diffusion dithering gives us less contrast images.

To better demonstrate how diffusion technique affects the image, I choose one of my Mac's default desktop background because of the gradient from earth atmosphere to the space. As a result, the thresholding technique gives a striking contrast image with a clear border, and error diffusion dithering technique gives less contract results. I also notice that with larger mask, error diffusion result will be more natural.

Part of my desktop background. — pic2.2



Snapshot of pic2.2's result. — pic2.3

• Problem 3 - Interpolation

Here, we want to enlarge a picture with 2 different interpolation functions.

- **Nearest-neighbor (NN) interpolation**

```
% nearest-neighbor (NN) interpolation
for i = 1:4*w
    for j = 1:4*h
        % make sure ii, jj in range [1, size]
        ii = min(w, round((i-1)/4)+1);
        jj = min(h, round((j-1)/4)+1);
        nni_img(i, j, :) = cat_img(ii, jj, :);
    end
end
```

Nearest-neighbor interpolation is very naïve. Simply use *round* to find where where that position belongs to will do the trick.

- **Bilinear interpolation**

```
% bilinear interpolation
% we neglect the border here
for i = 4:4*w-4
    for j = 4:4*h-4
        a = i/4; b = j/4;
        x = floor(a); y = floor(b);
        A = a-x;
        B = b-y;
        % assigning weights
        bi_img(i, j, :) = ...
            (cat_img(x, y, :)*(1-A)*(1-B) ...
            + cat_img(x+1, y, :)*(A)*(1-B) ...
            + cat_img(x, y+1, :)*(1-A)*(B) ...
            + cat_img(x+1, y+1, :)*(A)*(B));
    end
end
```

Bilinear interpolation involves the concept of assigning weights. First, we have to find it's four neighboring pixels. Then, assign the weights to four pixels according to the distance from (a, b).

Nearest-neighbor result. — pic3.1

Bilinear result. — pic3.2

Compare with two results, we can clearly see artifacts in pic3.1 and more natural result in pic3.2.