Programming Assignment 6

NikhilTank: 13110070

1. Develop a gradient domain HDR compression technique to compress the dynamic range of a given HDR image. Display the final tone mapped LDR image and evaluate its quality using dynamic range independent quality metric available online.

Preview of HDR Images:

```
%%Matlab code for Gradient Domain HDR compression
% Sequence
% taking log of luminance
% taking gradient of log(luminance)
% obtaning the phi
% cal gradient attenuation function for each level of gauss
% scaling factor at every level
% cal gradient attenuation function for each level of gauss
% obtaning cap Phi
% multiplying gradx,y to phi and solving poisson equation
% equating lap I = div G
% make verticle source matrix
% convert verticle into required
% transform the HDR color according to the new calculated luminance
clc
clear all
count=0;
hdr = hdrread('H.hdr');
hdr = imresize(hdr, [2*64 2*128]);
lab = rgb2lab(hdr);
```

```
L_k = lab(:, :, 1); % Extract the L image
A = lab(:, :, 2);   % Extract the a image
B = lab(:, :, 3);   % Extract the b image
GDHDR = hdr;
% taking log of luminance
lnL=abs(log(L_k));

% taking gradient of log(luminance)
[gradx,grady] = gradient(lnL);

%% obtaning the phi
% make gauss pyr till d
n = log2(size(L_k));
d = min(n)-5;   % to make course matrix with min height or width = 32
G = lnL;
gauss = {G};
for i = 1:d
   G = impyramid(G,'reduce');
   gauss = [gauss;G];
end

% cal gradient attenuation function for each level of gauss
gradH = {};
for k = 1:d+1
   temp = gauss{k};
   [x,y] = size(temp);
   value = zeros(x, y, 2, 'double');
      for i = 1:x
         for j = 1:y
            if i+1>x
               value(i,j,1) = (0-temp(i-1,j))/(2^(k+1));
               %value(i,j,1) = (0)/(2^(k+1));
            elseif i-1<1
               value(i,j,1) = (temp(i+1,j)-0)/(2^(k+1));
               %value(i,j,1) = (0)/(2^(k+1));
            else
               value(i,j,1) = (temp(i+1,j)-temp(i-1,j))/(2^(k+1));
            end
            if j+1>y
               value(i,j,2) = (0-temp(i,j-1))/(2^(k+1));
               %value(i,j,2) = (0)/(2^(k+1));
            elseif j-1<1
               value(i,j,2) = (temp(i,j+1)-0)/(2^(k+1));
               %value(i,j,2) = (0)/(2^(k+1));
            else
               value(i,j,2) = (temp(i,j+1)-temp(i,j-1))/(2^(k+1));
            end
         end
      end
```

```
        gradH = [gradH;value];
end

% scaling factor at every level
% cal gradient attenuation function for each level of gauss
phi = {};
for k = 1:d+1
    temp1 = gradH{k};
    [x,y,z] = size(temp1);
    value = zeros(x, y, 'double');
    alpha = 0.1*(mean2(abs(temp1(:,:,1))) + mean2(abs(temp1(:,:,2))) )/2;
    beta = 0.88;
    for i = 1:x
        for j = 1:y
            norm_gradH = sqrt(temp1(i,j,1)^2 + temp1(i,j,2)^2);
            value(i,j) = (alpha/norm_gradH)*(norm_gradH/alpha).^beta;
        end
    end
    phi = [phi;value];
end

% obtaning cap Phi
 P=phi{d+1};
for k = 1:d
     app = phi{d+1-k};
     org = imresize(P,2);
     [x,y] = size(app);
     for i = 1:x
        for j = 1:y
            P(i,j) = org(i,j)*app(i,j);
        end
     end
end

%% multiplying gradx,y to phi and solving poisson equation
[x,y] = size(P);
Gx = zeros(x, y, 'double');
for i = 1:x
    for j = 1:y
            Gx(i,j) = gradx(i,j)*P(i,j);
    end
end

Gy = zeros(x, y, 'double');
for i = 1:x
    for j = 1:y
            Gy(i,j) = grady(i,j)*P(i,j);
    end
end
```

```
%% equating lap I = div G
[x,y] = size(Gx);
divG = zeros(x, y, 'double');
for i = 1:x
   for j = 1:y
      if i-1<1
         divG(i,j) = 0;
      elseif j-1<1
         divG(i,j) = 0;
      else
         divG(i,j) = Gx(i,j) - Gx(i-1,j) + Gy(i,j) - Gy(i,j-1);
      end
   end
end

poi = speye(x*y,x*y);
for i =1:x*y %row variable %start & endpt included
   poi(i,i) = (-4);
   if i-y >= 1
      poi(i,i-y) = 1;
   end
   if i-1 >= 1
      if rem((i-1),y) ~= 0
         poi(i,i-1) = 1;
      end
   end
   if i+y<=x*y
      poi(i,i+y) = 1;
   end
   if i+1<=x*y
      if rem(i,y) ~= 0
         poi(i,i+1) = 1;
      end
   end
end

%% make verticle source matrix
[x,y] = size(Gx);
b = zeros(x*y,1);
r=0;
for i = 1:x %row variable
   for j = 1:y
      b(i+j+r-1,1) = divG (i,j);
   end
   r = r + y - 1;
end
```

```
vertI=poi\b;

%% convert verticle into required
I = zeros(x,y);
r=0;
for i = 1:x %row variable
   for j = 1:y
      I(i,j) = vertI(i+j+r-1,1) ;
   end
   r = r + y - 1;
end
 L_j = exp(I);
lab(:, :, 1) = exp(I);
HDR = lab2rgb(lab);
[x,y,z] = size(hdr);
s=.25;
GDHDRC = zeros(x,y,z,'double');

%% transform the HDR color according to the new calculated luminance
for k=1:z
   for i = 1:x
       for j = 1:y
          GDHDR(i,j,k) = L_j(i,j).*(hdr(i,j,k)/L_k(i,j)).^s;
       end
   end
end
imwrite(GDHDR,'Compressed_HDR.jpg'); %Gradient domain compressed HDR image
imwrite(HDR,'HDR.jpg'); %untoned hdr image
```

Rendered Output:

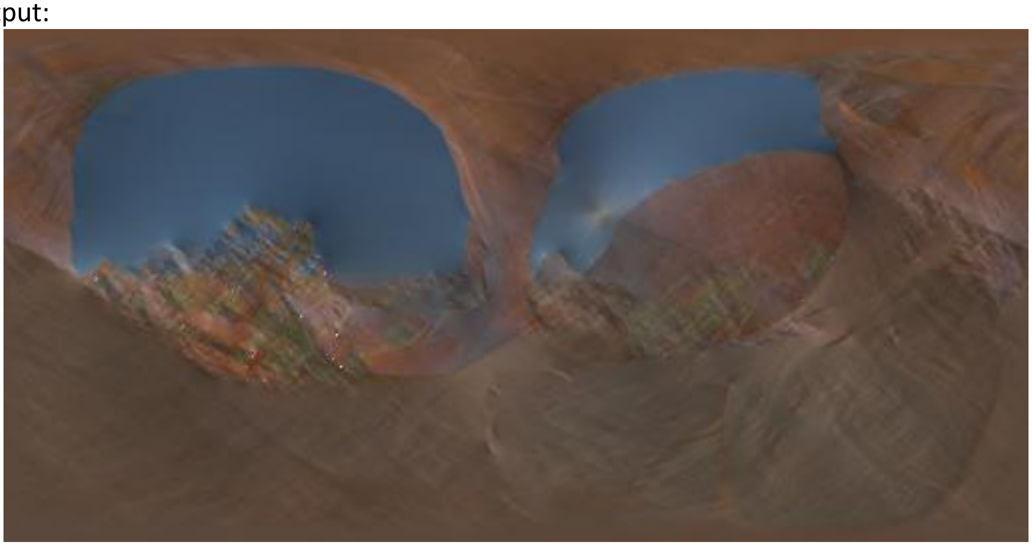

Fig: ‘Compressed_HDR.jpg’

2. Develop an algorithm to perform matting from an image of a foreground captured with two different background colors (blue, green).

Original Images:

Fig: 'Blue.png'

Fig: 'Green.png'

Note: There is a finite region where alpha belongs (0, 1) where flames are present. We want to extract the information of FG, i.e. Dragon and flames, then we want to transfer the same effect to a new back ground of choice.

```
%%Matlab code for Image matting
clc
clear all
count=0;
gre = imread('Green.png'); %RGB order of color
gre = imresize(gre, .5);
blu = imread('Blue.png');
blu = imresize(blu, .5);
BG_new = imread('BG.png');
BG_new = imresize(BG_new, .5); % new backgroung
[x,y,z]=size(gre);
alpha = zeros(x, y);
FG = gre;
IMG = BG_new;
a = (blu-gre);
b = (gre-blu);
c = 254;  %c = (gre(1,1,2)-blu(1,1,2))
for k = 1:z % calc alpha only for green channel
   for i = 1:x
      for j = 1:y
         if k ==2
            alpha(i,j,k) = (254-b(i,j,k));
         elseif k ==3
            alpha(i,j,k) = (254-a(i,j,k));
         end
      end
   end
end
Alpha = alpha./c;
Alpha(:,:,1) = Alpha (:,:,2);
Beta = ones(x,y,z) - Alpha; %(1 - Alpha(i,j,k)) at every pt
```

```
BG = zeros(x, y, z);
BG(:,:,2) = 254;
for k = 1:z % calc alpha only for green channel
   for i = 1:x
      for j = 1:y
         FG(i,j,k) = (gre(i,j,k)-Beta(i,j,k).*BG(i,j,k))./Alpha(i,j,k);
      end
   end
end
for k = 1:z % ransfer the same effect to a new back ground of choice
   for i = 1:x
      for j = 1:y
         IMG(i,j,k) = Alpha(i,j,k).*FG(i,j,k) + Beta(i,j,k).*BG_new(i,j,k);
      end
   end
end
imwrite(FG,'FG.png');
imwrite(IMG,'Image_matted.png');
```

Extracted FG:

Fig: ‘FG.png’

Transferring this FG to a new BG of choice, using same alpha matt:

Fig: ‘Image_matted.png’

Note: Here you can see the alpha matting is still retained and the flames transparent remains as it was on blue screen and green screen.

Reference:

[1] http://www.hdrlabs.com/sibl/archive.html

[2] http://in.mathworks.com/help/images/examples/color-based-segmentation-using-the-l-a-b-color-space.html

[3] Raanan Fattal, Dani Lischinski and Michael Werman , Gradient Domain High Dynamic Range Compression, School of Computer Science and Engineering The Hebrew University of Jerusalem