Aynur Süne - 21041

EE417 First Post-Lab Report

1) Linear Scaling

```
function newimg = lab1linscale(img)

[row,col,color] = size(img);

if(color==3)
img = rgb2gray(img);
end

newimg = zeros(size(img));
img = double(img);

imgmax = max(img(:));
imgmin = min(img(:));
Gmax = 255;
a = -imgmin;
b = Gmax/(imgmax-imgmin);


newimg = b.*(img+a);

newimg = uint8(newimg);
imshow(newimg)
end
```

Linear scaling is a point operator that linearly scales values of pixels in an image between u_{min} and u_{max} using a gradation function. For this task, final image's u_{min} and u_{max} values must be 0 and 255 respectively. Output of the above function is as follows:

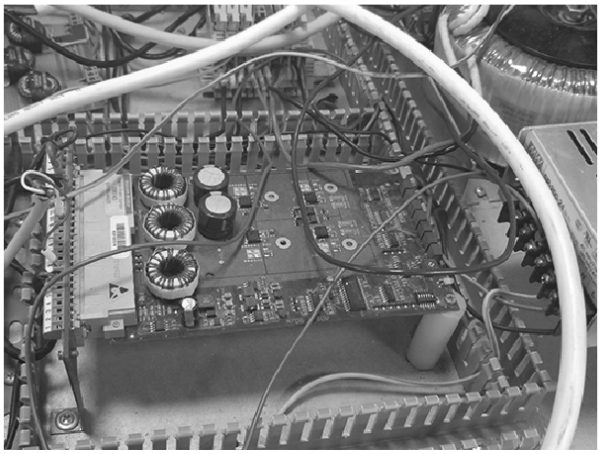
2) Conditional Scaling

An image is mapped into a new one that has the same mean and variance with a reference image. New pixel values are computed using a gradation function.

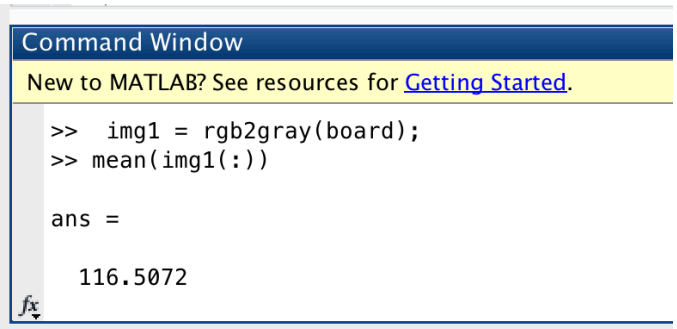
```
function [newimg] = lab1condscale(img1,img2)

[row,col,color] = size(img1);
if(color==3)
    img1 = rgb2gray(img1);
end

[row2,col2,color2] = size(img2);
if(color2==3)
    img2 = rgb2gray(img2);
end
newimg = zeros(size(img2));
img1=double(img1);
img2=double(img2);

meanref = mean(img1(:)); % img1 reference image I
stdref =std(img1(:));

meanimg = mean(img2(:)); % img2 our image
stdimg =std(img2(:));

a = meanref*(stdimg/stdref)-meanimg;
b = stdref/stdimg;

newimg = b.*(img2+a);

newimg = uint8(newimg);
img1 = uint8(img1);
imshow(newimg)
imshow(img1)
end
```

Output of the above function is as follows:

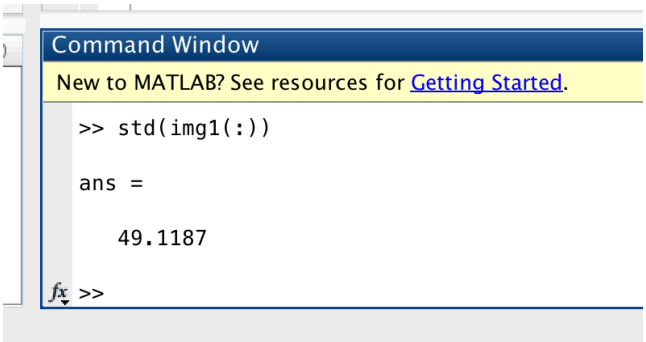
Reference Image:



Reference Image Mean:



Reference Image Standard Deviation:



Current Image:

Current Image Mean:

```
Command Window
New to MATLAB? See resources for Getting Started.
>> mean(city(:))
ans =
   35.7225
fx >>
```

Current Image Standard Deviation:

```
Command Window
New to MATLAB? See resources for Getting Started.
>> std(city(:))
ans =
   15.0345
fx >>
```

Processed Image:

Processed Image Mean:

```
Command Window
New to MATLAB? See resources for Getting Started.
>> mean(cityref(:))
ans =
  116.5056
fx >>
```

Processed Image Standard Deviation:

```
Command Window
New to MATLAB? See resources for Getting Started.
>> std(cityref(:))
ans =
   49.1361
fx >>
```

As It can be observed, image gets brighter since reference image has higher pixel values(brighter) thus mean and standard deviation is higher than current image. When

current image's values change with reference image's mean and standard deviation, it gets brighter.

3) Local Mean Filter

Local Mean Filter is used to reduce noise in an image by convolution with a sliding window. Each pixel is replaced by the mean(average) of the pixel values in the window. Window size is computed as (2k+1)x(2k+1) where k is a user input. To achieve sliding window, we need to traverse the image pixels with a double for loop.

```
function [newimg] = lab1locbox(img,k)
[r,c,color] = size(img);

if(color==3)
    img = rgb2gray(img);
end
newimg = zeros(size(img));
img = double(img);

for i=k+1:1:r-k-1
    for j= k+1:1:c-k-1
        subimg = img(i-k:i+k,j-k:j+k);
        newimg(i,j) = mean(subimg(:));
    end
end

newimg = uint8(newimg);
imshow(newimg)
end
```

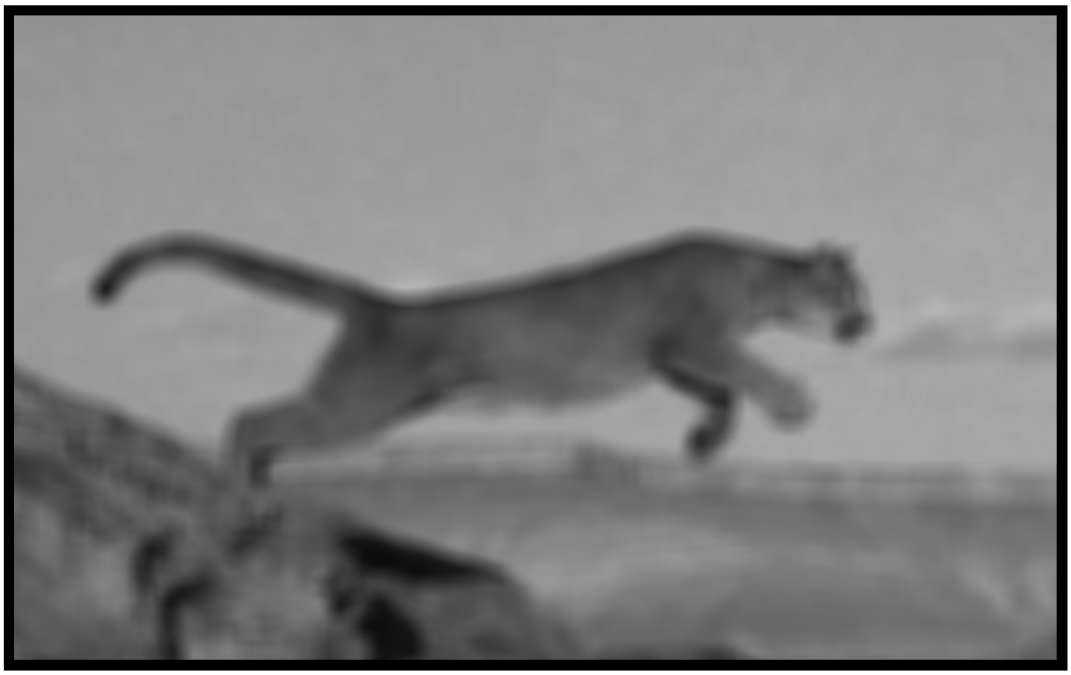
Results of the above function with different k values are as follows:

Box Filtered Image, k=3

Box Filtered Image, k=5

Box Filtered Image, k= 8

Box Filtered Image, k=15



As it can be observed, thickness of black frame increases when k value increases since outer k number of pixels are not traversed or computed. When k (smoothing) increases too much, image gets blurry. When k is large, pixel values are more generalized thus image is less sharp and blurry.

4) Local Max and Min Filters

Convolution with sliding window is used as in third part; instead of taking average of the pixels, each pixel value is replaced by minimum and maximum values of the window in the double for loop and output is two images.

```
function [newimgmin, newimgmax] = lab1locmaxmin(img,k)
[r,c,color] = size(img);

if(color==3)
    img = rgb2gray(img);
end

newimgmin = zeros(size(img));
newimgmax = zeros(size(img));
img = double(img);

for i=k+1:1:r-k-1
    for j= k+1:1:c-k-1
        subimg = img(i-k:i+k,j-k:j+k);
        newimgmin(i,j) = min(subimg(:));
        newimgmax(i,j) = max(subimg(:));
    end
end
```

```
newimgmin = uint8(newimgmin);
newimgmax = uint8(newimgmax);
imshow(newimgmin);
imshow(newimgmax);
end
```

Local Min Filtered Image, k=3

Local Max Filtered Image, k=3

Local Min Filtered Image, k=5

Local Max Filtered Image, k=5

Local Min Filtered Image, k=10

Local Max Filtered Image, k=10

As it can be seen from output images, Min filters are darker than original image and Max filters are brighter. As k value increases, darkness and brightness increase because probability of having a minimum or maximum value that differs entirely from rest of the window pixel's values increase. Only one pixel value may be enough to change a part of the image completely. This can be observed clearly from the last max filter where k equals to 10. Obvious square shaped (window) white/grey parts exist. Also images get blurry as in third part.