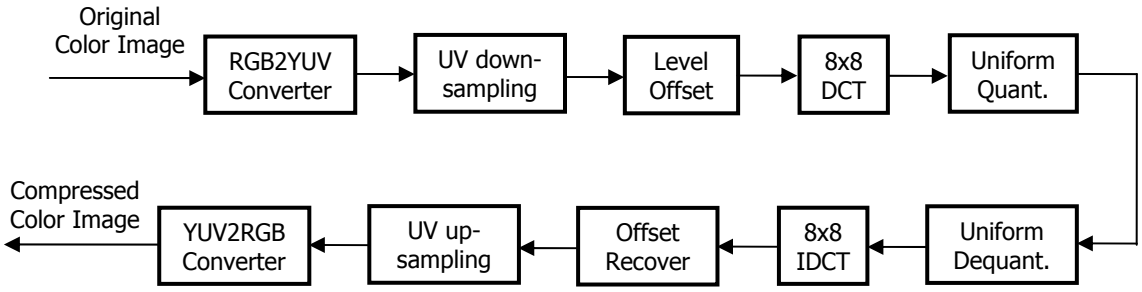
Lab Assignment
Quality Enhancement for Compressed Images and Videos
June 27th, 2008

I. JPEG Compression:



The code for the above JPEG compression algorithm is given in the function jpeg.m with syntax:

out_img=jpeg(in_img, scaling_factor);

where in_img is the original color image in bmp format and out_img is compressed color image in bmp format.

Purpose: write the Matlab file makeJPEG.m to:

- Compress the original image using JPEG standard.
- Calculate the PSNR.
- Verify the quality of the compressed image.

I.1. Read the input image: use the following code

```
%read the original image
[FileName,PathName] = uigetfile('*.bmp','Select the uncompressed image');
in_img(:,:,:)=double(imread(strcat(PathName,FileName),'bmp'));
```

I.2. Assign the quality_factor=25 and write the code to find the scaling_factor as in the following equation

$$scaling\_factor = \begin{cases} 50/quality\_factor & if \quad quality\_factor \le 50 \\ 2-50/quality\_factor & if \quad quality\_factor > 50 \end{cases}$$

I.3. Use the function out_img=jpeg(in_img, scaling_factor) to compress the in_img input image. Open the function to verify the detailed procedure of the JPEG compression as in Fig. 1.

I.4. Write the out_img compressed image by using the code:

```
%write the compressed image
imwrite(uint8(out_img),strcat(PathName,FileName(1:end-4),'jpeg.bmp'),'bmp');
```

I.5. Calculate the PSNR of the compressed image as in the following equation:

$$MSE = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N}\sum_{p=1}^{3}\left(in\_img(m,n,p) - out\_img(m,n,p)\right)^2}{M \times N \times 3}$$

$$PSNR = 10\log\frac{255^2}{MSE}$$

I.6. Run the composed code for the image mobile003.bmp.

I.7. Display both the original and compressed images by the following code:

```
%show the images
figure(1);
subplot(2,2,1);
imshow(uint8(in_img));
title('Original')
subplot(2,2,2);
imshow(uint8(out_img));
title('JPEG')
```

I.8. Adjust the quality_factor to see the blocking and ringing artifacts and to determine the range of quality_factor where these artifacts are imperceptible.

I.9. Repeat I.8. for other original images in the database to see that different kinds of images will be affected differently by JPEG compression.

II. MJPEG Compression:

Purpose: write the Matlab file makeMotionJPEG.m to compress the sequence using the MJPEG compression.

II.1. Set the avi parameter and create the avi files for the original and compressed video sequences:

```
fpsec=30;
aviobj_org =…
avifile(strcat('mobile_org.avi'),'fps',fpsec,'compression','None','quality',100);
aviobj_mpjeg =…
avifile(strcat('mobile_mjpeg.avi'),'fps',fpsec,'compression','None','quality',100);
```

II.2. Set the quality_factor=25 and write the code to calculate the scaling_factor. Read each frame of the original video sequence and compress using JPEG:

```
No_Frame=50;
for k=1:No_Frame
   k
   thousandth=fix(k/1000);
   rem=k-fix(k/1000)*1000;
```

```
    hundredth=fix(rem/100);
    rem=rem-hundredth*100;
    tenth=fix(rem/10);
    unit=rem-tenth*10;
    in_video_frame(:,:,1:3)=double(imread(strcat('database_directory \mobile'...
       ,char(hundredth+48),char(tenth+48),char(unit+48),'.bmp'),'bmp'));

    %compress the original frame
    out_video_frame=jpeg(in_video_frame,scaling_factor);

    %add frame to the video sequences
    aviobj_org = addframe(aviobj_org,uint8(in_video_frame));
    aviobj_mpjeg = addframe(aviobj_mpjeg,uint8(out_video_frame));
  end
  aviobj_org = close(aviobj_org);
  aviobj_mpjeg = close(aviobj_mpjeg);
```

II.3. Run the composed code and adjust the quality_factor to see the mosquito and flickering artifacts.

III. Deblocking the compressed JPEG image:

Purpose: use the provided function fuzzydeblock to reduce the blocking artifacts of the compressed JPEG images.

This function has the syntax:

```
Output=fuzzydeblock(Input, Img_height, Img_width, NBlocksize,sigma);
```

where NBlocksize is the block size of the separately processed blocks, which is equal to 8 for JPEG.

The main idea of this algorithm is to apply the fuzzy filter to the pixels near to the horizontal and vertical block borders between blocks. The fuzzy filter uses the Gaussian membership function with spread parameter sigma.

III.1. In the Matlab file makeJPEG.m, apply the function fuzzydeblock to remove the blocking artifact for R, G, B component separately. Write the deblocked image and find its PSNR. Verify the blocking reduction in the deblocked image.

III.2. Similar to III.1., but the deblocking algorithm is applied to Y, U, V component of the compressed image. The code to convert RGB to YUV and vice versa can be found in the function jpeg.m:

```
RGBtoYUV=[0.299  0.587  0.114
      -0.169 -0.332  0.500
       0.500 -0.419 -0.081];
%Convert from RGB to YUV
```

```
for i=1:M
   for j=1:N
      tempi(1:3,1)=in_img(i,j,:);
      YUV(i,j,:)=RGBtoYUV*tempi;
   end
end
%convert YUV to RGB
for i=1:M
   for j=1:N
      tempi(1:3,1)=YUVjpeg(i,j,:);
      out_img(i,j,:)=(RGBtoYUV)^(-1)*tempi;
   end
end
```

Write the deblocked image and find its PSNR. Compare the visual quality and PSNR value to between the simulation results in III.1. and III.2.

III.3. Find the optimal sigma to get the highest PSNR for the deblocked image.

III.4. Choose the very high value of sigma to see the blurry effects.

IV. Deringing the compressed JPEG image:

Purpose: use the provided function fuzzydering to reduce the ringing artifacts of the compressed JPEG images after deblocking.

This function has the syntax:

Output=fuzzydering(Input, Img_height, Img_width, H_length, W_length, sigma);

where H_length, W_length are the window sizes of pixels centered by the pixel of interest.

The main idea of this algorithm is to apply the fuzzy filter to the set of pixels centered by the pixels of interest to reduce the ringing artifact. The fuzzy filter uses the Gaussian membership function with spread parameter sigma.

IV.1. From the code of III.2., apply the function fuzzydering only to the Y component with sigma=12 to reduce the ringing artifacts. Write the deringed image and find its PSNR. Verify the quality improvement of the enhanced image after deblocking and dernging.

IV.2. Find the optimal sigma for deringing to achieve the highest PSNR. Then set sigma=30 to see the blurry effect. Are the blurry effects different for different areas, such as flat and detail areas?

IV.3. If both the U and V components are deranged, will we get further improvement? Hint: use imshow function to show the U and V components, compare them with its corresponding Y component.

IV.4. Develop the algorithm to enhance the quality for the MJPEG sequence using the code in II.2.

V. Demosquito and deflickering the compressed MJPEG sequence:

Purpose: use the provided function fuzzydemosquito to reduce the mosquito and flickering artifacts. The syntax of this function is:

```
Output=fuzzydemosquito(Input, Img_height, Img_width, …
T_lenght, H_length, W_length, sigma);
```

where T_lenght, H_length, W_length are the cubic sizes of the spatio-temporally surrounding pixels centered by the pixel of interest. The fuzzy filter in this case is applied to the 3D surrounding pixels instead of only spatially surrounding pixels for deblocking or deringing.

V.1. In the new Matlab file 3DEnhance.m, set the cubic size to 5x5x5, sigma=8 and read T_lenght frames from the MJPEG avi video sequence.

V.2. Deblock only the current frame using the function fuzzydeblock.

V.3. Demosquitos and deflickering using the function fuzzydemosquito.

V.4. Develop the code for enhancing one JPEG frame to the code for enhancing the MJPEG sequence by using the code in II. Play the original, MJPEG and enhanced sequences to verify the mosquito and flickering artifact reduction of the enhanced sequences.

V.5. Compare the result of IV.4 and V.4. in mosquito and flickering artifact.