

SD 575 Image Processing

Fall 2018

Lab 5: Image Compression and Segmentation
Due Friday November 30, 2018 at 11:59 PM (Midnight)

Note: All lab reports will be submitted on Learn to group dropboxes.

For help on how to use the Matlab functions mentioned throughout the lab, please type ‘help’ followed by the function (e.g., `help imread`). Function information can also be found through Google.

1 Overview

The goal of this lab is to provide some hands-on experience with fundamental image compression concepts and techniques. Due to the exponential growth in usage and storage of digital graphic media, image compression has been very important in helping reduce storage and transmission bandwidth problems. Many real-world applications depend heavily on image compression, such as digital photography, video games, digital movie archival/streaming, and medical imaging. For this lab, we will study some fundamental image compression techniques such as chroma subsampling, image transform, and quantization.

The following images will be used for testing purposes:

- `lena.tif`
- `peppers.png`

These images can be found on the course website.

2 Chroma Subsampling

First, we will study the effects of chroma subsampling on image quality and how it can be used to provide image compression. For this study, we will use the peppers image. First, let us convert the image from the RGB colorspace into the YCbCr colorspace using the `rgb2ycbcr` function. Plot each of the image channels (Y, Cb, and Cr) separately.

1. Describe the Cb and Cr channel images. Why do they appear this way?
2. Compare the level of image detail in the Cb and Cr images with the Y channel image. Which contains more fine details? What does that say about the luma (Y) and chroma (Cb and Cr) channels?

Now, reduce the resolution of the chroma channels by a factor of 2 in both the horizontal and vertical directions and then upsample them back to the original resolution using bilinear interpolation. The `imresize` function will come in handy. Also, you will need to separate the color image into three separate images so you can operate on them independently. Recombine the original Y channel image and the two upsampled Cb and Cr images to create a new color image. This can be considered a simple way of reducing network bandwidth for image/video transfer – that is, downscaling the chroma images on the server, sending them over a network to a client, and rescaling on the client.

1. Compare the resulting image from chroma sub-sampling with the original image. How large are the visual differences?
2. Based on the resulting image, what can you say about chroma sub-sampling and its effect on image quality?

Let us study the effects of luma sub-sampling. Reduce the resolution of the luma (Y) channel by a factor of 2 in both the horizontal and vertical directions and then upsample it back to the original resolution using bilinear interpolation. Recombine the upsampled Y channel image and the original Cb and Cr images to create a new color image.

1. Compare the resulting image from luma sub-sampling with the original image. How large are the visual differences?
2. Based on the resulting image, what can you say about luma sub-sampling and its effect on image quality?
3. Compare the resulting image from luma sub-sampling with the image produced using chroma sub-sampling. Which method performs better? Which is better for reducing network bandwidth while preserving visual acuity? Why?

3 Colour Segmentation

Image segmentation is useful in identifying regions of interest in an image to gain a more meaningful representation of the image. In this section, colour segmentation is explored using k -means classification to segment various colours in an image using the RGB and $L^*a^*b^*$ colour spaces. The $L^*a^*b^*$ colour space consists of a luminosity dimension (L^*) and two colour dimensions called a^* and b^* . The a^* channel indicates colour which falls along the red-green axis while the b^* channel indicates colours which fall along the blue-yellow axis.

Load the `peppers.png` image in Matlab and convert the image to the $L^*a^*b^*$ colour space. (Hint: `applycform` is useful here.)

Then, classify the colours in the a^* and b^* colour spaces using k -means clustering for $k = 2$ and $k = 4$. Use `kmeans()` to do colour segmentation with squared Euclidean distance and the following starting point matrix, μ .

```
% K = 2;
% row = [55    200];
% col = [155    400];

K = 4;
row = [55    130    200    280];
col = [155    110    400    470];

% Convert (r,c) indexing to 1D linear indexing.
idx = sub2ind([size(f,1) size(f,2)], row, col);

% Vectorize starting coordinates
for k = 1:K
    mu(k,:) = ab(idx(k,:),:);
end
```

First reshape the a^* and b^* channels:

```
ab = double(im_lab(:,:,2:3)); % NOT im2double
m = size(ab,1);
n = size(ab,2);
ab = reshape(ab,m*n,2);
```

Output the cluster indices and show the resulting classification for a given k , using the following code, where `cluster_idx` are the cluster indices. This should show in one image, each cluster with its own unique colour.

```
% Label each pixel according to k-means
pixel_labels = reshape(cluster_idx, m, n);
h = figure, imshow(pixel_labels, [])
title('Image labeled by cluster index');
colormap('jet')
```

1. For the various values of k , how did the clustering change? Explain.

2. What is the effect of the initial points on the final clusters? Does this impose any limitations? Why?

Next, output each cluster/segmented region using the original colours of the image using the `pixel_labels` found for $k = 4$. (*Hint*: `repmat` is useful here. When showing the segmented regions for a particular pixel label, other pixels should be set to zero.)

3. Include an image of each cluster and comment on the segmentation performance.

4 Image Transform

Let us now study the discrete cosine transform (DCT) and the characteristics of an image in the transform domain. The DCT decomposes an image into a series of sinusoids with different amplitudes and frequencies. In block transform coding algorithms, the image is divided into smaller sub-images and each sub-image is transformed using an image transform separately. Perhaps the most popular image transform for block transform coding is the DCT. One efficient method for computing the DCT of a sub-image is to use the DCT transform matrix. The DCT transform matrix can be constructed using `dctmtx` function. For this study, we will use the 8×8 DCT transform matrix T and use grayscale Lena image `f` as the test image. Plot the 8×8 DCT transform matrix.

1. What does each row of the DCT transform matrix represent? Look at the pattern for each row. If you still don't see it, try plotting each of the rows as a 1-D function.

Now apply the DCT transformation matrix on each 8×8 sub-image. This can be performed as follows:

```
F_ttrans = floor(blockproc(f-128, [8 8], @(x) T*x.data*T'));
```

Plot the DCT of the 8×8 sub-image with top-left corner at $(row, col) = (297, 81)$ and the DCT of the sub-image with top-left corner at $(row, col) = (1, 1)$.

1. Describe the energy distribution of the DCT of the sub-images. What does each pixel represent? Explain why DCT would be useful for image compression in the context of the DCT energy distribution.
2. Compare the DCT of the two sub-images. How are they different? Why? Explain in the context of the image characteristics at those locations and the DCT energy distribution.

Now let's try discarding all but 6 of the DCT coefficients in each sub-image and then reconstructing the image. This can be done by first applying a threshold to the sub-images,

```

mask = [1  1  1  0  0  0  0  0;
        1  1  0  0  0  0  0  0;
        1  0  0  0  0  0  0  0;
        0  0  0  0  0  0  0  0;
        0  0  0  0  0  0  0  0;
        0  0  0  0  0  0  0  0;
        0  0  0  0  0  0  0  0;
        0  0  0  0  0  0  0  0];
F_thresh = blockproc(F_trans, [8 8], @(x) mask.*x.data);

```

and then performing an inverse DCT on the sub-images

```

f_thresh = floor(blockproc(F_thresh, [8 8], @(x) T'*x.data*T)) + 128;

```

Plot the reconstructed image and the corresponding PSNR.

1. Describe how the reconstructed image looks compared to the original image. Why does it look this way?
2. What artifact is most prominent in the image? Why does this artifact appear?
3. What conclusions can you draw about the DCT in terms of image compression? Does it work well? If yes, why does it work well?

5 Quantization

One of the most important steps in lossy image compression stage is the quantization step. It is highly desired that the transform coefficients of a sub-image is quantized in such a way that the amount of data needed to represent the image is greatly reduced without causing undesirable artifacts. Let us now study the effects of different levels of quantization on image quality. For this study, the grayscale Lena image will be used as the test image. First, we will construct the quantization matrix used in the JPEG standard:

```

Z = [16  11  10  16  24  40  51  61;
     12  12  14  19  26  58  60  55;
     14  13  16  24  40  57  69  56;
     14  17  22  29  51  87  80  62;
     18  22  37  56  68  109 103  77;
     24  35  55  64  81  104 113  92;
     49  64  78  87  103 121 120 101;
     72  92  95  98  112 100 103  99];

```

Now perform the 8×8 DCT transform on the Lena sub-images (remember to subtract 128). To perform quantization on the sub-images, divide the sub-images by Z , and then round the resulting quantized DCT. To reconstruct the image, multiply the quantized DCT sub-images by Z and then perform the inverse DCT transform on the sub-images (remember to add 128). Plot the reconstructed image and the corresponding PSNR. Now perform the above quantization process again on the image, but this time using $3Z$, $5Z$, and $10Z$. Plot the reconstructed images and the corresponding PSNR.

1. What happens to the DCT coefficients when quantization is performed? What effect does it have on image quality?
2. Compare the reconstructed image produced using $3Z$ with the original image. Why does the reconstructed image look this way?
3. Compare the reconstructed images produced by the different levels of quantization, as well as the PSNR for each reconstructed image. What happens as the level of quantization increases?
4. Which artefact becomes more prominent as the level of quantization increases? Why?
5. What conclusions can you draw about the quantization process? Explain in the context of the trade-off between compression performance and image quality.

6 Report

Include in your report:

- A brief introduction.
- Pertinent graphs and images (properly labelled).
- Include code (can be included in appendix).
- Include responses to all questions.
- A brief summary of your results with conclusions.