# Lab 5: Image Compression and Segmentation

SYDE 575

Thomas Fortin, 20712151
Cathleen Leone, 20711309

December 7th,  2021

Department of Systems Design Engineering
University of Waterloo

# Introduction

In this lab we are going to study chroma subsampling, colour segmentation, DCT block coding, quantization, and neural networks for image recognition.

# Chroma Subsampling

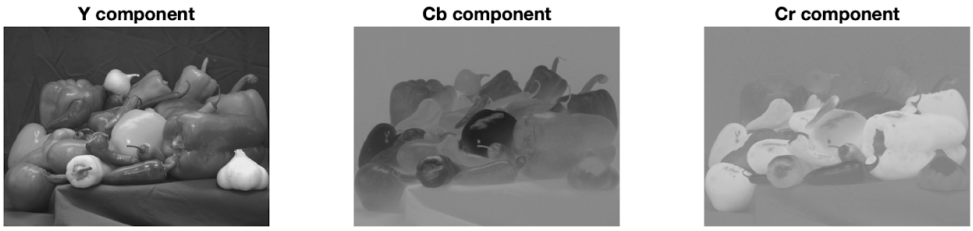



Figure 1: chroma subsampling of peppers.

1. The Cb and Cr images have less defined detail and are less informative to the human eye than the Y component image. There is generally also less contrast in the Cb and Cr images. This is because there are not necessarily large differences in blues and reds throughout the image. The Cb image shows the blue differences, which are mostly dark since there is not a lot of blue in the image, and the Cr component shows the red differences, which are mostly lighter because there is a bit more red in the image.
2. The Y component contains more fine details than the Cb or the Cr component images. This is because the Y component represents the luma component, which shows the differences in brightness throughout the image. This does not rely on the colour information itself, so certain areas of the image that may not show up as well when measured in terms of blue differences or red differences, but still have changes in brightness, are able to be seen in the Y component image.

Figure 2: Original peppers


Figure 3: chroma resampled and reconstructed peppers (left), luma resampled and reconstructed peppers (right)

3. There are no perceptible differences between the chroma subsampling and the original image.

4. The effects of chroma subsampling on image quality are very small and negligible.

5. The visual differences are in the fine detail. Edges and fine textures are smoothed over in the luna subsampling version of the image. Although the overall shape of the fruits is still preserved. The edges are not as perceptible.

6. The luma subsampling seems to have smoothed out the edges and fine detail in the image. This is not desirable and thus luma subsampling has a larger negative effect on image quality than chroma subsampling does.

7. The chroma subsampling method performed better. It is more ideal to compress the chroma information for reducing network bandwidth since more visual acuity will be preserved. This is because subsampling the chroma channels results in a better image that subsampling the lumam channels. So we can compress the chroma channels, transfer the images, and then upsample at the client and still preserve high visual acuity.

# Colour segmentation

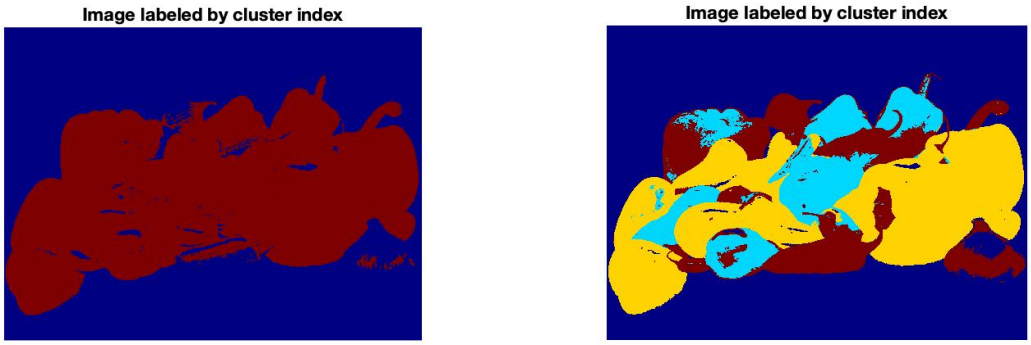



Figure 4: peppers image segmented with k_means clustering using k=2 (left) and k=3 (right)

8. For k=2 there are two different clusters. For k=4 there are 4 different clusters. In each case, one cluster was mostly composed of the purple background while the rest made up the peppers. So in the case of k=2 the other cluster was made up of all the peppers while in the case of k=4 the other three clusters represented 3 different colour groups of peppers.

9. The initial points can determine which local minimum the final clusters represent. If the initial conditions are chosen well, the final solution will be the global minimum. But if we choose bad initial conditions we may fall into a local minimum which is not the global minimum while completing the optimization algorithm. This means that we cannot be sure that we've found the best clusters unless we try all initial conditions and compare the resulting clusters' rmse values.

Figure 5: 4 clusters resulting from k means clustering with k=4

10. The above clusters seem to have been quite successful. The purple, red and green have been separated well from each other. This makes sense since they are the largest colour groups in the image. The top right segmentation contains some yellow and some light green. The bottom right segmentation contains dark green and white. The algorithm was not able to keep white in its own segmentation. It likely ended up like this because white is not a very dominant colour in the image. So the other colours were more important. I imagine that if we set k=6 we might get separate yellow and white segmentations.

# Image transform

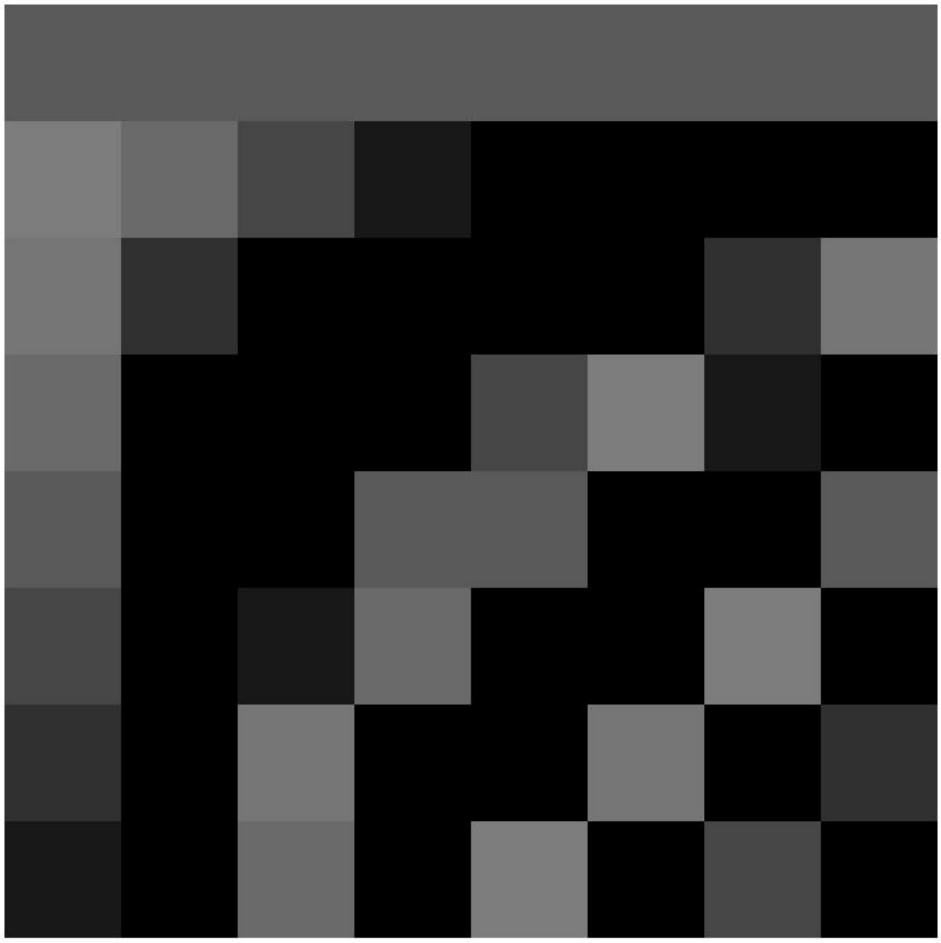


Figure 6: DCT transform matrix

11. The rows of the DCT transform matrix are made up of cosine functions at different frequencies. The top row is actually just a constant (0 frequency) with frequencies increasing as we go down.

Figure 7: Original Lena image

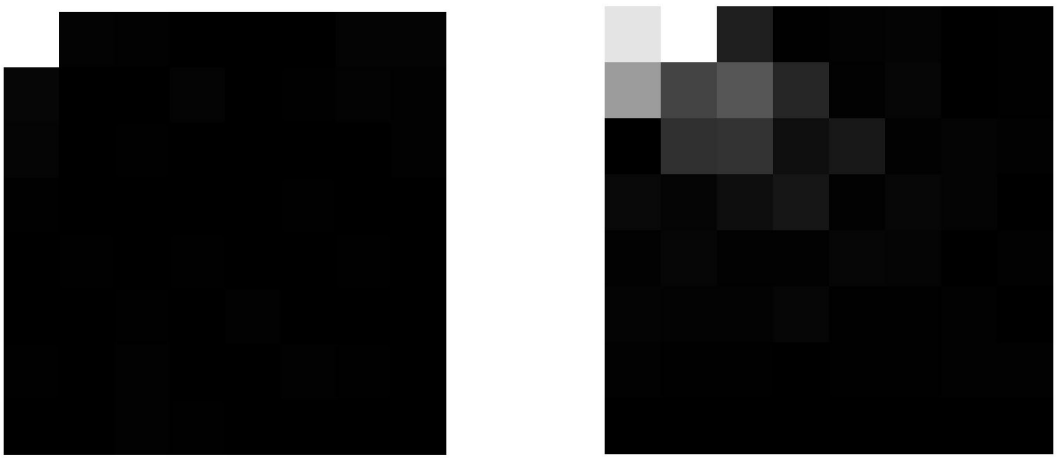


Figure 8: DCT of subimages of lena with top left corner starting at (1,1) (left) and top left corner starting at (81, 297) (right)

12. The energy distribution of the DCT of the subimage with top left corner starting at (1,1) is very low everywhere except for in the top left cell where the energy is high. (Remember that we are rescaling with [] so we just know that the top left cell is the highest value. We don't know if it's 1 or something else). The energy distribution of the DCT of the subimage with top left corner starting at (81, 297) is higher around the top left corner of the DCT. However, the high values are not constrained to just the leftmost and highest cell. The 3x3 block of cells in the top left are visibly showing high values along with a few others dispersed around them. The rest of the cells in the DCT are low.

Each pixel in the DCT represents the energy of the frequency band proportional to its cell number in the subimage. The pixel on the top left represents DC energy and as we move farther down and to the right the pixels represent higher frequency energy. As we move to the right we get higher frequency in the horizontal direction while moving further down gives us higher frequency in the vertical direction.

As we can see in the DCTs above, most of the values we see are low. So if we could not save any of the low values and just ignore them, we could store an image in a compressed format.

13. As was said previously, the energy distribution of the DCT of the subimage with top left corner starting at (1,1) is very low everywhere except for in the top left cell where the energy is high. This makes sense because in the top left corner of the original lena image, there is not much change in intensity throughout space. So it makes sense that we have almost exclusively DC energy.

The energy distribution of the DCT of the subimage with top left corner starting at (81, 297) is higher around the top left corner of the DCT. However, the high values are not constrained to just the leftmost and highest cell. The 3x3 block of cells in the top left are visibly showing high values along with a few others dispersed around them. The rest of the cells in the DCT are low. This makes sense since this area of the image is an edge. So we need some more energy in some non-zero frequency cells to create the edge. However, the resolution of the image is high enough that the edge transition is not a perfect step. That is, it's smooth. So we do not need every cell in the DCT to encode that edge.

PSNR = 30.5573
Figure 9: Reconstructed lena after discarding all but 6 DCT cells

14. Above, the reconstructed image after discarding most of the DCT can be seen. The image looks somewhat fuzzy. As we saw previously, we needed more cells than the ones we kept in order to encode an edge in this image. Since we got rid of some of those cells, the edges have become smoother and thus the image is fuzzier.

15. We see a blocking artifact. This is because the error presented by the DCT quantization is smoothed everywhere except on the borders between subimages. It is smoothed within the subimages because we are using low frequency components. But since subimages are independent, there is a discontinuity in the image where two subimages share a border.

16. It does work well. It works well because most images are high resolution enough to not need to present perfect edges for the human visual system to perceive them as perfect edges. So, these images do not require much high frequency information. And even if there is some high

frequency information in an image, we probably wouldn't notice if it was gone. So DCT quantization takes advantage of this by just not storing that information.

## Quantization

PSNR=35.7338
Figure 10: Quantized lena with 1z

PSNR=32.2912
Figure 11: quantized lena with 3z

PSNR=30.3832
Figure 12: quantized lena with 5z

PSNR=27.3112
Figure 13: quantized lena with 10z

17. When we perform quantization, we're basically rounding the DCT coefficients to the nearest multiple of their corresponding value in the Z matrix. So for the top left DCT value, we're rounding to the closest multiple of 16 (0, 16, 32, 48 …). We are basically reducing the resolution of the energy spectrum. This means that there are fewing intensities to select from to represent the DC component of a subimage as well as the amplitude of each non-zero frequency component. This is most noticeable in terms of limiting the intensity resolution of the DC component. As can be seen in the reconstructed image with 10z, blocking is very evident and large groups of blocks all have exactly the same DC component because there are so few values to choose from when doing such radical quantization.

18. In the 3z reconstructed image, we can just barely start to make out some blocking in areas with low non-zero frequency components. Areas with lots of detail still seem as good as in the 1z reconstructed image. This is because the blocking effect has not yet become strong enough to overcome the higher signal content of the more detailed areas. We could say that the signal to noise ratio in those areas is still high.

19. As the level of quantization increases, we lose more and more detail and the blocking artifacts become more intense. As expected, the PSNR also decreases.

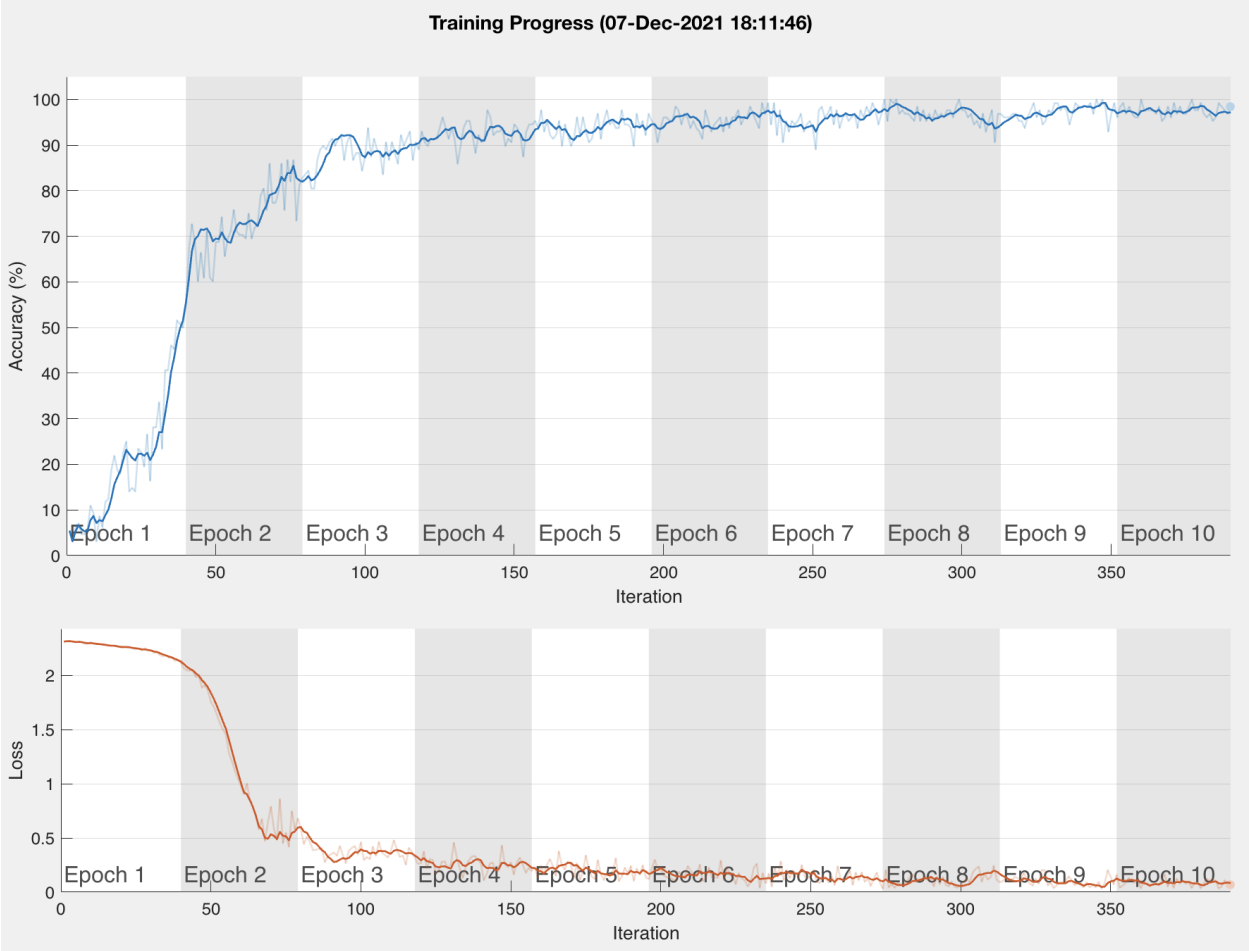
20. The blocking artifact becomes more prominent as the level of quantization increases. This is because the quantization is applied to a block coding and each block is thus independent of its neighbour. When quantization increases we lose more and more detail until we're relying more and more on just the DC component of subimages. Since this can only be perceived to be changing between blocks, higher quantization causes more perceptible blocking artifacts.

21. Quantization can compress an image really well. However, it cannot keep all information stored in an image. Luckily, most images do not contain a lot of high frequency information so quantization is able to avoid focusing on those components and still deliver a high quality image. However, there is a limit to how much an image can be compressed before it loses too much detail to be considered acceptable quality.

## Convolutional neural networks

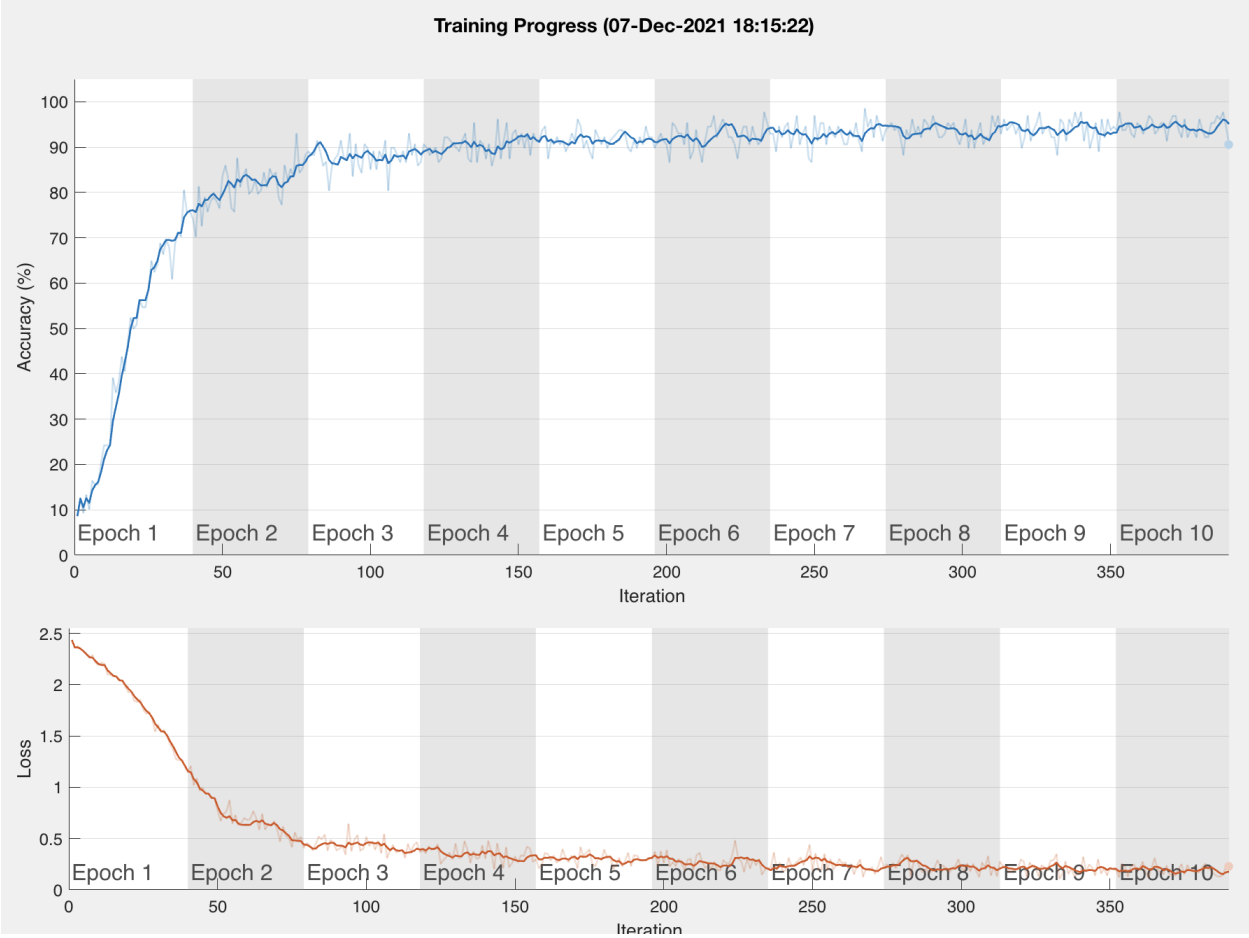
22. Fully connected lakers are more expensive because they have independent weights connecting each input to each output (NxM) while convolutional layers have only a set number of weights (usually 25 or 49) for each output channel (25 to 49 X C). Of course, the dimensions for the weight matrix for either type of layer don't use any of the same variables in my definition. That's because the outputs are very different. It's common to want to have more total outputs from a convolutional layer than a fully connected layer. So it depends on how big you make each layer.

Convolutional layers are beneficial for images because they can focus on features like edges and textures in a more explicit and focused way than fully connected layers can. We've seen throughout our whole SYDE 575 class that convolution is capable of detecting such features.




Test Accuracy: 0.9592

Figure 14: learning curves for CNN with 10 epochs




Test accuracy: 0.9128

Figure 15: learning curves for MLP with 10 epochs

23. As can be seen above, it seems as though MLP does train slightly quicker than CNN. By this I mean that its accuracy increases faster than that of the CNN to start. However, CNN converges to a higher training and testing score. Slow and steady has won the race.

24. The testing accuracy in each case seems to be about 3-5% lower than the training accuracy.

25. Lowering the learning rate causes the speed of convergence to decrease. This means that the magnitudes of the slopes of the accuracy and loss curves are lower. Increasing the learning rate causes the algorithm to converge faster but also makes it more unstable. There are larger oscillations in the learning curves when the learning rate is increased. If the learning rate is too high ( like 1) it will never be able to converge because it will not be stable enough.

When completing training using each method for 100 epochs the resulting accuracies were:

MLP 100 epochs:
test_accuracy = 0.9362
train_accuracy = 1
CNN 100 epochs:
test_accuracy = 0.9710
train_accuracy = 1

26. In both cases with 100 epochs, the models obtained a 100% training accuracy. In the case of the MLP model, the testing accuracy was only 93.6%. This is a large difference and I would say is overfitting. However, the testing accuracy of the CNN with 100 epochs was 97.1%. This is quite close to 100% and still within the 3-5% difference between testing and training that we saw with 10 epochs. It's still overfit, but not as much as the MLP model. Perhaps this is because the CNN has fewer model parameters and therefore is less capable of memorizing the training dataset. Also, the CNN is built to pick up on features that the human visual system would pick up on while the MLP will just pick up anything. Since the written numbers are meant for humans to read, the CNN is predisposed to pick up the important information instead of noise that the MLP might use to help itself memorize the dataset.

## Conclusion

In this lab we studied chroma subsampling, colour segmentation, DCT transformation, quantization, and neural networks for image recognition. We discovered that Y, Cb, and Cr components of an image in the YCbCr color space preserve different characteristics of an image, colour segmentation with k_means clustering is capable of grouping objects in an image based on colour, DCT transformations are a type of block coding which can be very useful for compression, quantizations is compression technique employed by the jpeg standard which works very well but can introduce blocking artifacts if used too extremely, and that CNNs are a very useful layer for neural networks when working with images due to their ability to detect features such as edges and textures.

# Appendix

## Chroma Subsampling

```
img = imread('peppers.png');
ycbcr = rgb2ycbcr(img);
y = ycbcr(:,:,1);
cb = ycbcr(:,:,2);
cr = ycbcr(:,:,3);


figure()
imshow(img)

figure()
imshow(y)

figure()
imshow(cb)

figure()
imshow(cr)

new_cb = imresize(imresize(cb, .5, 'bilinear'), 2, 'bilinear');
new_cr = imresize(imresize(cr, .5, 'bilinear'), 2, 'bilinear');

new_ycbcr = zeros(size(ycbcr));
new_ycbcr(:,:,1) = y;
new_ycbcr(:,:,2) = new_cb;
new_ycbcr(:,:,3) = new_cr;
new_ycbcr = uint8(new_ycbcr);

new_img = ycbcr2rgb(new_ycbcr);

figure()
imshow(new_img)

new_y = imresize(imresize(y, .5, 'bilinear'), 2, 'bilinear');

new_ycbcr = zeros(size(ycbcr));
new_ycbcr(:,:,1) = new_y;
new_ycbcr(:,:,2) = cb;
new_ycbcr(:,:,3) = cr;
new_ycbcr = uint8(new_ycbcr);

new_img = ycbcr2rgb(new_ycbcr);

figure()
imshow(new_img)
```

## Colour Segmentation

```
clear all;

f_rgb = imread('peppers.png');
```

```
transform = makecform('srgb2lab');
f_lab = applycform(f_rgb, transform);

% imshow(f_lab)
% title('L*a*b* colour space')


% 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_lab,1) size(f_lab,2)], row, col);

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

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

cluster_idx = kmeans(ab, [], 'Start', mu);

% 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')

for k = 1:K
   img = zeros(size(f_rgb));
   mask = pixel_labels==k
   img(mask(:, :, [1 1 1])) = f_rgb(mask(:, :, [1 1 1]));
   figure()
   imshow(uint8(img));
end
```

## Image Transform

```
T = dctmtx(8);
figure()
imshow(T, 'InitialMagnification', 8000)

img = double(im2gray(imread('lena.tiff')));
figure()
imshow(img, [])

trans_img = floor(blkproc(img-128,[8 8],'P1*x*P2',T,T'));
```

```
figure()
imshow(abs(trans_img(1:8, 1:8)), [], 'InitialMagnification', 8000)

figure()
imshow(abs(trans_img(81:88, 297:304)), [], 'InitialMagnification', 8000)

mask = zeros(8, 8);
mask ( 1 , 1 ) = 1;
mask ( 1 , 2 ) = 1;
mask ( 1 , 3 ) = 1;
mask ( 2 , 1 ) = 1;
mask ( 3 , 1 ) = 1;
mask ( 2 , 2 ) = 1;
thres_trans_img = blkproc(trans_img, [8 8], 'P1.*x', mask);
thres_img = floor(blkproc(thres_trans_img,[8 8],'P1*x*P2',T',T)) + 128;

figure()
imshow(thres_img, [])

psnr(thres_img, img)
```

## Qantization

```
Q = [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];

Q = Q * 5;

T = dctmtx(8);

img_og = double(im2gray(imread('lena.tiff')));

trans_img = blkproc(img_og-128,[8 8],'P1*x*P2',T,T');
trans_img_scaled = round(blkproc(trans_img, [8 8], 'x./P1', Q));
img = blkproc(trans_img_scaled, [8 8], 'x.*P1', Q);
img = floor(blkproc(img, [8 8], 'P1*x*P2', T', T)) + 128;

imshow(uint8(img))
psnr(img/255, img_og/255)
```