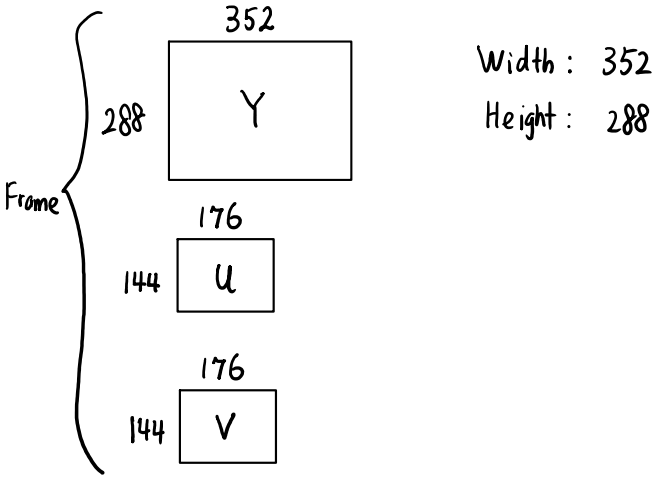
Lecture 3

Video Sequences in YUV format

http://www.ece.uah.edu/~dwpan/course/ee604/images/video/



Y: Luminance, or Luma (grayscale version of the color image)
U, V (Chrominance, or chroma)
U = 0.492 * (B - Y);
V = 0.877 * (R - Y);

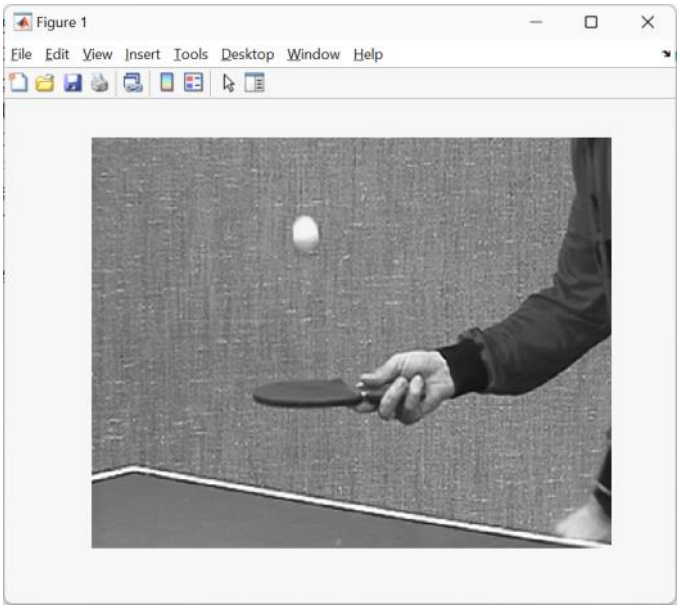
Chroma subsampling (less resolution for chroma information than for luma information, by taking advantage of the human visual system's lower acuity for color differences than for luminance.

4:2:0 format, for each image frame in CIF:
Chrominance: 352*288
U, V: 352*144

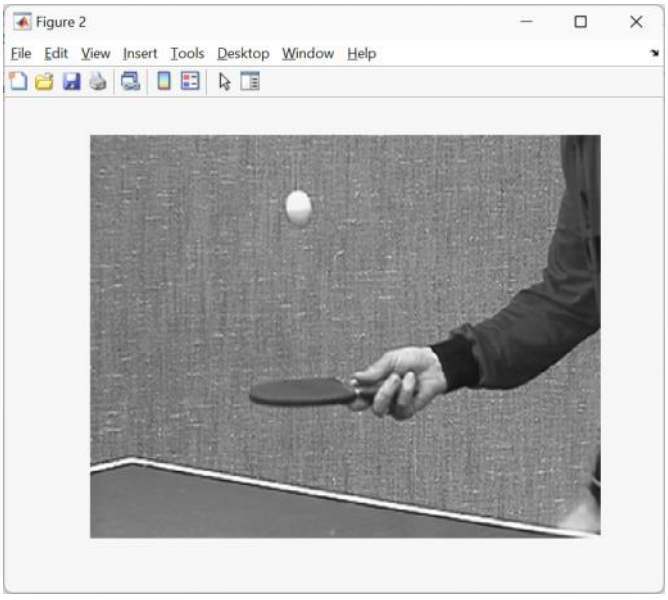
```
FS= 352*288;    % Size of the luminance components for one frame
FSC= 352*144;   % Size of the succeedding chrominance components for one frame
fid= fopen(i);
fseek(fid, (n-1)*(FS + FSC), 'bof');    % Point to a particular frame
cur = fread(fid,FS,'uchar');
y = reshape(cur,352,288)';
imagesc(y);
colormap(gray);
axis off;
fclose(fid);
```

```
function y = vcif2(i,n)
% function vcif(i,n)
% View the n-th frame (luminance) of the input CIF format sequence i
% CIF format has 352 pixels (horizontally) and 288 pixels(vertically)

% Usage:
% y1 = vcif2('tennis_cif.yuv',1);   % Read the first frame (luminance)

FS= 352*288;    % Size of the luminance components for one frame
FSC= 352*144;   % Size of the succeeding chrominance components for one
frame
fid= fopen(i);
fseek(fid, (n-1)*(FS + FSC), 'bof');    % Point to a particular frame
cur = fread(fid, FS, 'uchar');
y = reshape(cur, 352, 288)';
imagesc(y);
colormap(gray);
axis off;
fclose(fid);
```

1st frame



2nd frame




```
>> y1 = vcif2('tennis_cif.yuv',1);
```

```
>> y2 = vcif2('tennis_cif.yuv',2);
```

```
>> diff = abs(y1 - y2);
>> figure; imagesc(diff); colorbar
```

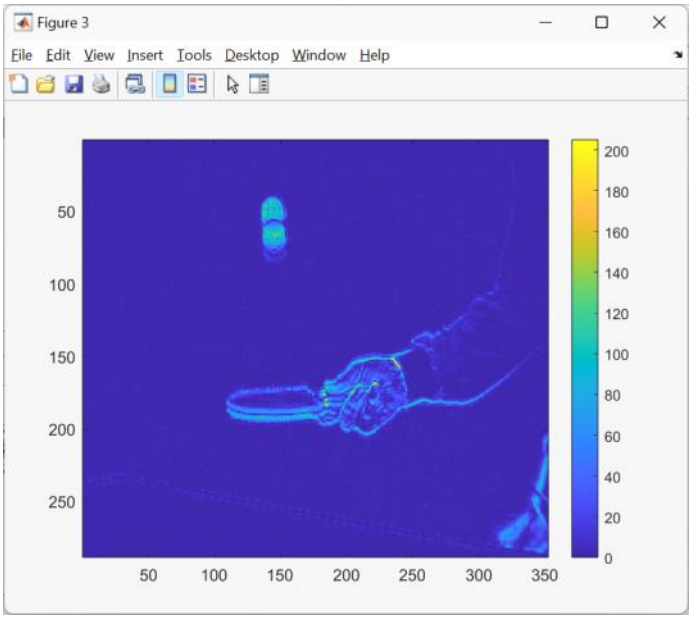
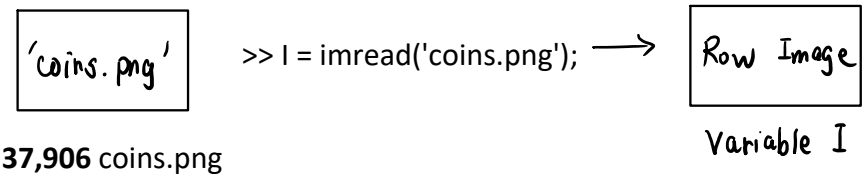


Image Read-in and **Write-out** with Matlab

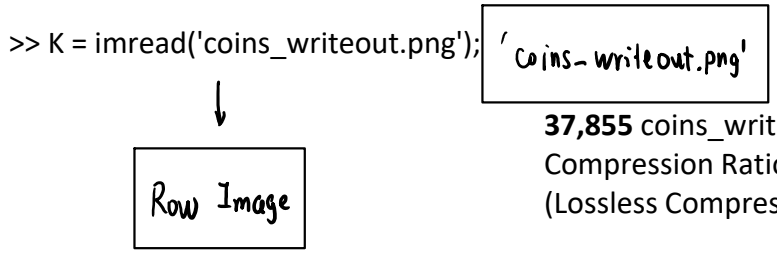
```
>> imfinfo('coins.png')
ans =

  struct with fields:

          Filename: 'C:\Program Files\MATLAB\R2023a\toolbox\images\imdata\coins.png'
       FileModDate: '13-Oct-2002 07:47:01'
          FileSize: 37906
            Format: 'png'
     FormatVersion: []
             Width: 300
            Height: 246
          BitDepth: 8
         ColorType: 'grayscale'
                ⋮
```



37,906 coins.png

```
  Name        Size              Bytes  Class
  Attributes
  I           246x300           73800  uint8
```

```
>> imwrite(I, 'coins_writeout.png');
```



37,855 coins_writeout.png
Compression Ratio: 2 to 1
(Lossless Compression)

```
>> 73800/37855
ans =
    1.9495
```

```
  K           246x300           73800  uint8
```

```
>> isequal(I, K)

ans =

  logical

   1
```

Functions:
imread ( ) --- Image Decompressor (Decoder)
Imwrite ( ) --- Image Compressor (Coder)

=> Lossless Compression!

What if we write the raw image data into a JPG file
JPG: Joint Picture Expert Group

Raw Image

Variable I **73800 bytes**

↓ Compressor

>> *imwrite*(I, 'coins_writeout_jpg.jpg'); **8,782** coins_writeout_jpg.jpg

↓

>> I_JPEG = imread('coins_writeout_jpg.jpg');

Compression Ratio: 8:4 to 1 compression by using JPEG format at the cost of distortion between the original raw image and the reconstructed image (I_JPEG)

```
Name       Size          Bytes  Class    Attributes
 I_JPEG    246x300       73800  uint8
```

>> 73800/8782

ans =

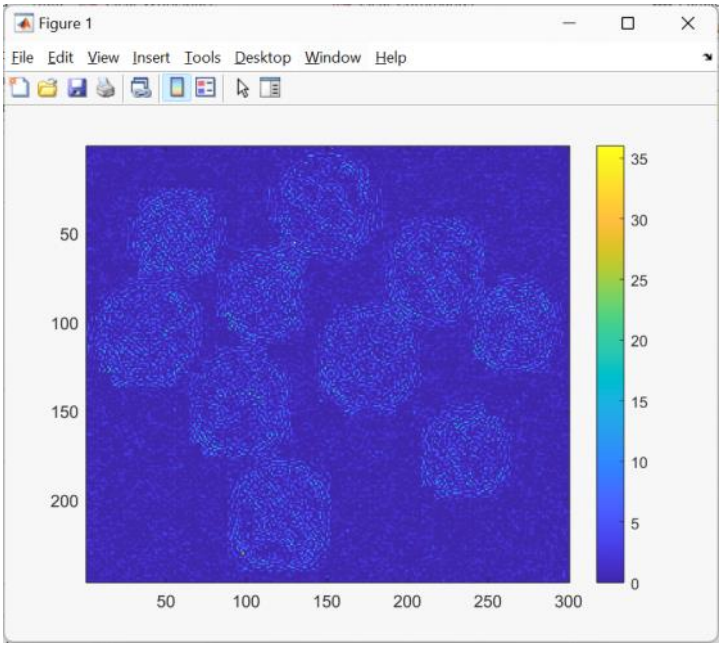
8.4036

>> isequal(I, I_JPEG)

ans =

logical

0 => **Lossy Compression**



>> diff = abs(I - I_JPEG);
>> figure; imagesc(diff); colorbar

Tradeoff between the quality of the reconstructed image and the compression ratio

Quality — Quality of output file
75 (default) | scalar in the range [0, 100]
Quality of the output file, specified as a scalar in the range [0, 100], where 0 is lower quality and higher compression, and 100 is higher quality and lower compression.

```
>> imwrite(I, 'coins_writeout_jpg_q25.jpg', 'quality', 25);
```

4,346 coins_writeout_jpg_q25.jpg

```
>> I_JPEG_q25 = imread('coins_writeout_jpg_q25.jpg');
>> diff_q25 = abs(I - I_JPEG_q25);
>> figure; imagesc(diff_q25); colorbar
```