CM3106: Multimedia
Tutorial/Lab Class 5 (Week 6)
MATLAB Graphics, Images and Video Formats

Prof David Marshall

dave.marshall@cs.cardiff.ac.uk

School of Computer Science & Informatics
Cardiff University, UK

All Lab Materials available at:
http://www.cs.cf.ac.uk/Dave/Multimedia/PDF/tutorial.html

Dithering

- Dithering is often used when converting greyscale images to bit-mapped ones e.g. for printing
- The main strategy is to replace a pixel value (from 0 to 255) by a larger pattern (e.g. 4×4) such that the number of printed dots approximates the greyscale level of the original image
- If a pixel is replaced by a 4×4 array of dots, the intensities it can approximate from 0 (no dots) to 16 (full dots).
- Given a 4×4 dither matrix e.g.

$$\begin{pmatrix} 0 & 8 & 2 & 10 \\ 12 & 4 & 14 & 6 \\ 3 & 11 & 1 & 9 \\ 15 & 7 & 13 & 5 \end{pmatrix}$$

we can re-map pixel values from 0–255 to a new range 0–16 by dividing the value by (256/17) (and rounding down).

A Basic Dithering Template

Q1 Hint: Need to replace with a proper **Dithering Matrix**, dither_q1_hint.m

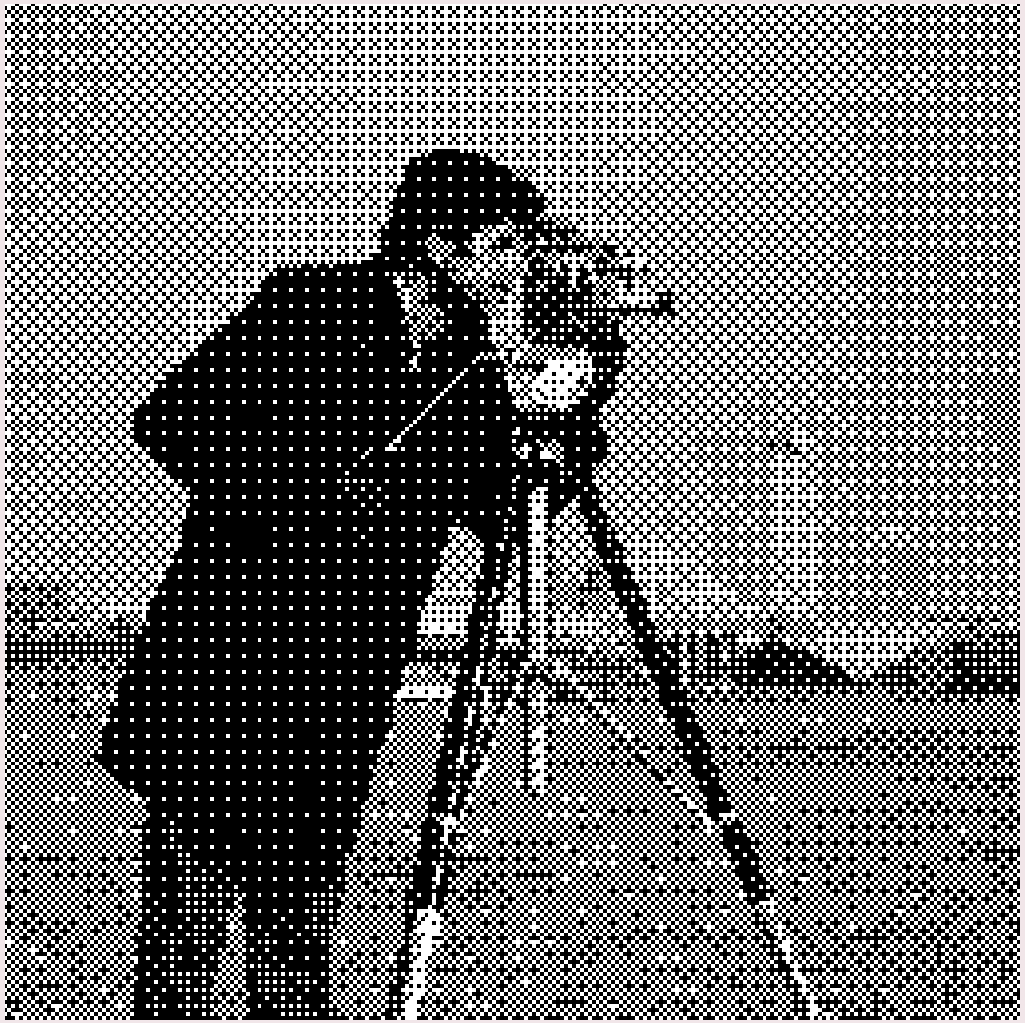
```
im = imread('cameraman.tif'); %Load the image


di = 4*ones(4,4);  % REPLACE WITH PROPER Dithering MATRIX

[m n] = size(im); % Obtain image size

mat = repmat(di, m/4, n/4); % Repeat the matrix to same size

im = im / 17; % Map intensity to 0-16
dithered = im > mat; % Set 1 when entry has im > mat

imshow(dithered); % Show the dithered image
```

A Basic Dithering Template(cont.)

dither_q1_hint.m (Not really dithered) Output

A Basic Dithering Template(cont.)

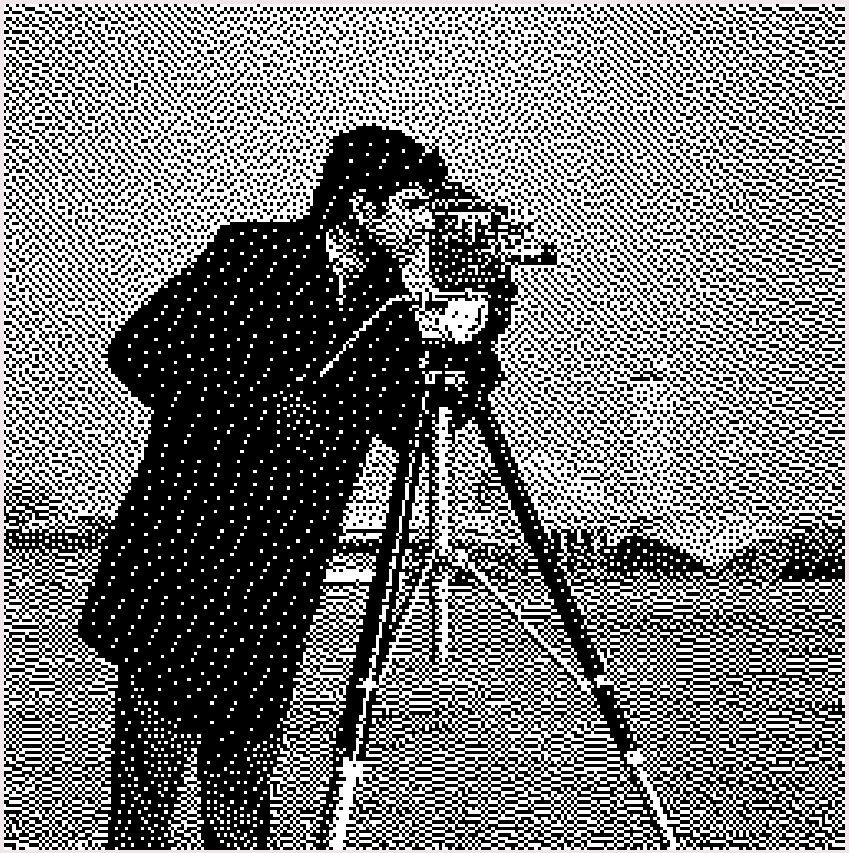
Q1 Task: Produce output like this!



MATLAB's dither() Function

MATLAB dither() example, dither_demo.m

```
I = imread('cameraman.tif');
BW = dither(I);
imshow(BW);
```



MATLAB's image processing toolbox colour space functions:

- **Colormap manipulation**:
  `colormap` — Set or get colour lookup table
  `rgbplot` —Plot RGB colourmap components
  `cmpermute` — Rearrange colours in colormap.
- **Colour space conversions**:
  `hsv2rgb`/`rgb2hsv` — Convert HSV values/RGB colour space
  `lab2double`/`lab2uint16`/`lab2uint8` — Convert Lab colour values to double etc.
  `ntsc2rgb`/`rgb2ntsc` — Convert NTSC (YIQ)/RGB colour values
  `ycbcr2rgb`/ `rgb2ycbcr` — Convert YCbCr/RGB colour

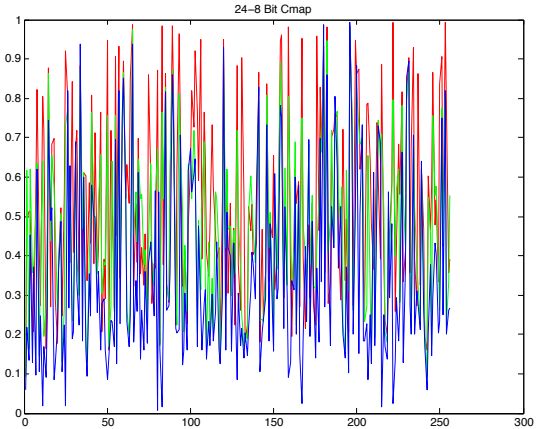
MATLAB Colour Demo Code

rgb_eg.m:RGB 24 to 8-bit Conversion (256 Colours)

Use `rgb2ind()` — see `doc/help rgb2ind()`.

- Returns 8-bit image: `im8bit`, and
- Colourmap: `cmap8bit`
- `rgbplot()` plots a **histogram**/**graph** of the colour map

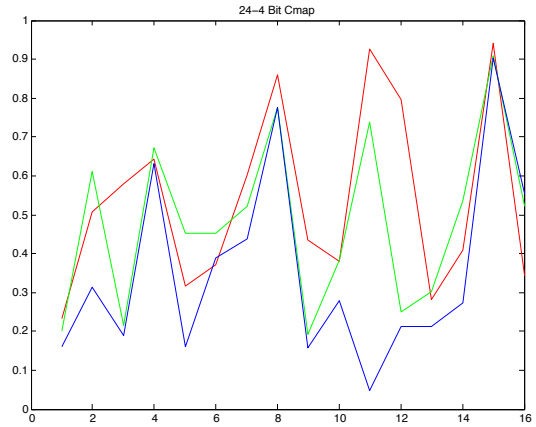
```
imRGB = imread('parrots.jpg');
figure,  imshow(imRGB), title('RGB Image');

% Convert to 8-bit
[im8bit, cmap8bit] = rgb2ind(imRGB,256);

figure,
imshow(im8bit, cmap8bit), title('24-8 Bit Image');

figure,
rgbplot(cmap8bit), title('24-8 Bit Cmap');
```



MATLAB Colour Demo Code (Cont.)

rgb_eg.m: Can create other Colour maps sizes, RGB 24 to 4-bit Conversion (16 Colours)

```
% Convert to 4-bit
[im4bit, cmap4bit] = rgb2ind(imRGB, 16);

figure,
imshow(im4bit, cmap4bit),
title('24-4 Bit Image');

figure,
rgbplot(cmap4bit), title('24-4 Bit Cmap');
```



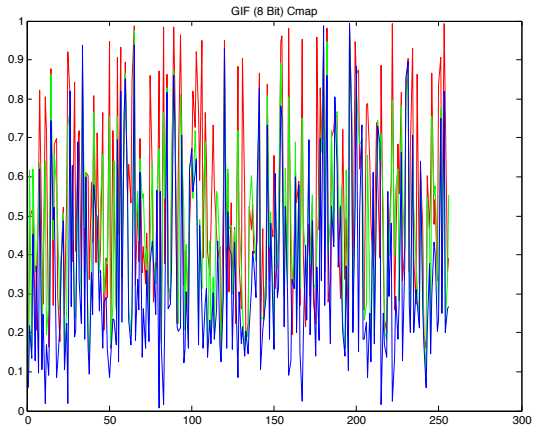
MATLAB Colour Demo Code (Cont.)

rgb_eg.m: Getting GIF image and Colormap

Use `imread()`:

- Returns 8-bit image: `imGIF`, and
- Colourmap: `cmapGIF`

```
%  Gif (8bit) Cmap.
[imGIF, cmapGIF] = imread('parrots.gif');

figure,
imshow(imGIF, cmapGIF), title('24-8 Bit Cmap');

figure,
rgbplot(cmap8bit), title('GIF (8 Bit) Cmap');
```



MATLAB Colour Demo Code (Cont.)

rgb_eg.m: Changing a Colourmap

For the current figure, call `colormap()` with a give colormap as parameter:

- See `doc/help colormap` for full list of default colour maps
- Can supply your own.

```
% Change Colourmap to a
% Predefined MATLAB 'Jet' cmap

figure,
imshow(imGIF, cmapGIF),
title('Jet Cmap'); % show image

colormap('jet'); % change its colormap
```

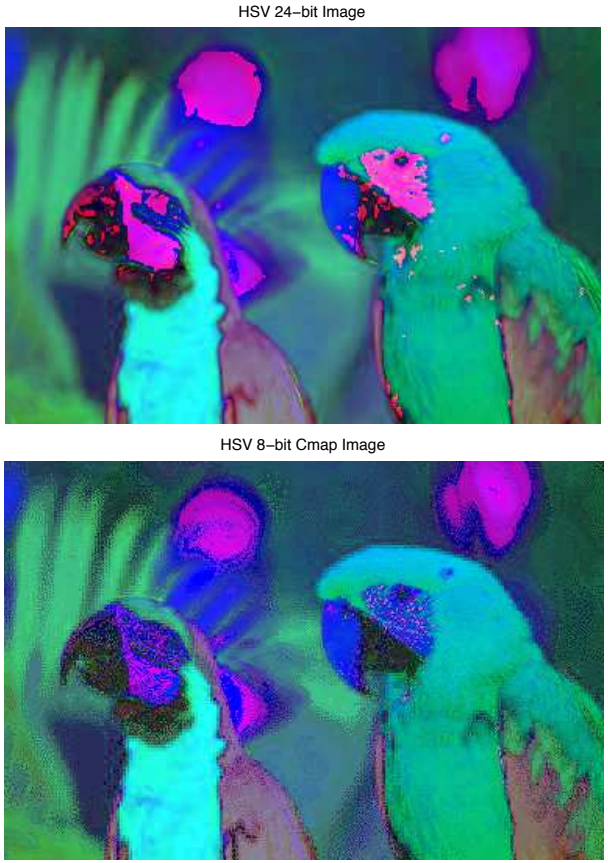
MATLAB Colour Demo Code (Cont.)

rgb_eg.m: Changing to a different Colour Space (HSV) here, others similar (Q2 Hint!)

Use `rgb2hsv()` (or `hsv2rgb()` to get back RGB).

- Image is **input parameter**.

```
% Example colour space conversion

hsv_image24 = rgb2hsv(imRGB); % 24-bit
figure,
imshow(hsv_image24),
title('HSV 24-bit Image');

cmap_hsv8 = rgb2hsv(cmapGIF); % 8-bit
figure,
imshow(imGIF,cmap_hsv8),
title('HSV 8-bit Cmap Image');
```



MATLAB Colour Demo Code (Cont.)

rgb_eg.m: Displaying Colour Channels

- Each **24-bit colour model** is essentially a **3D-Array**
- **Image Coordinates** — dimension 1 and 2 (x-y)
- **Colour Channel** — **3rd dimension** (z)
  - So in MATLAB easy to plots any channel: *e.g.* `imRGB(:,:,1)` for **RED plane**.

```
% Show Colour channels
figure, imshow(imRGB(:,:,1)), title('RGB R plane');
figure, imshow(imRGB(:,:,2)), title('RGB G Plane');
figure, imshow(imRGB(:,:,3)), title('RGB B Plane');


figure, imshow(hsv_image24(:,:,1)), title('HSV H plane');
figure, imshow(hsv_image24(:,:,2)), title('HSV S Plane');
figure, imshow(hsv_image24(:,:,3)), title('HSV V Plane');
```

MATLAB Colour Demo Code (Cont.)

RGB R plane

RGB G Plane

RGB B Plane

HSV H plane

HSV S Plane

HSV V Plane

Chroma Subsampling

Operate on color difference components

The signal is divided into:

Luma (Y): the **intensity** component and

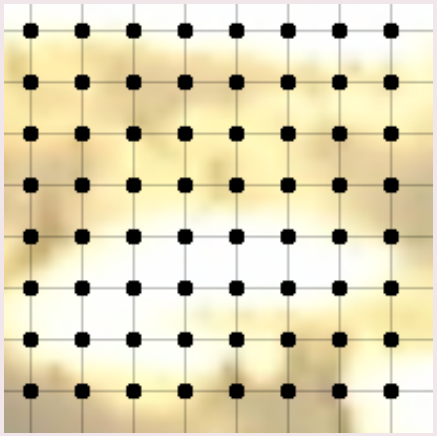
Chroma: **two color difference** components which we **subsample** in some way to reduce its **bandwidth**

- **Analogous** to **Analog** Video Compression (NTSC or PAL).

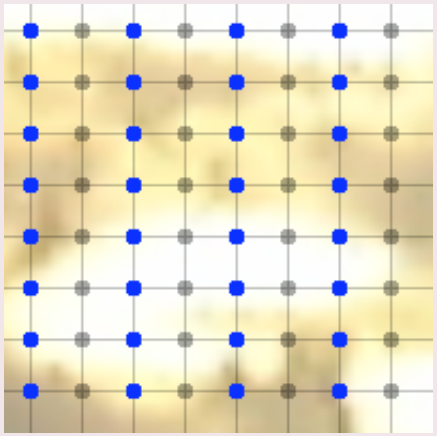
Chroma Subsampling: How to Compute?

Simple Image sub-sampling:

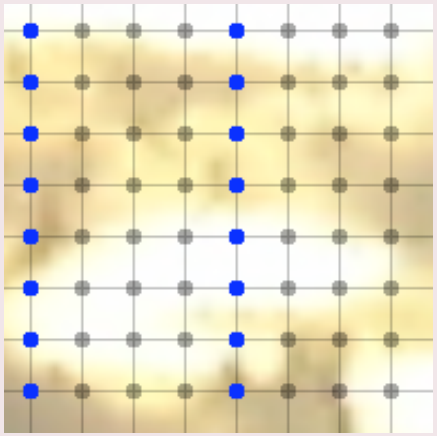
- Simply **different** **frequency sampling** of digitised signal
- Digital Subsampling: For 4:4:4, 4:2:2 and 4:1:1 Perform 2x2 (or 1x2, or 1x4) chroma subsampling
  - Subsample horizontal and, where applicable, vertical directions
  - *i.e.* Choose every second, fourth pixel value.



4:4:4



4:2:2

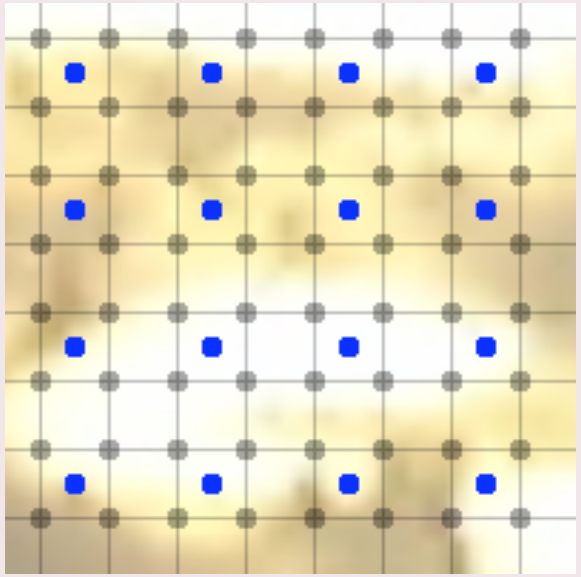


4:1:1

Chroma Subsampling: How to Compute? (Cont.)

4:2:0 Subsampling:

- For **4:2:0**, **Cr** and **Cb** are effectively centred vertically halfway between image rows.:
  - Break the image into **2x2** pixel blocks and
  - Stores the **average** color information for each **2x2** pixel group.



Chroma Subsampling in MATLAB

The MATLAB function `imresize()` readily achieves all our subsampling needs:

```
IMRESIZE Resize image.
   IMRESIZE resizes an image of any type using the specified
   interpolation method. Supported interpolation methods
   include:
        'nearest'  ---  (default) nearest neighbour interpolation
        'bilinear' bilinear interpolation

   B = IMRESIZE(A,M,METHOD) returns an image that is M times the
   size of A. If M is between 0 and 1.0, B is smaller than A. If
   M is greater than 1.0, B is larger than A.
   B = IMRESIZE(A,[MROWS MCOLS],METHOD) returns an image of size
   MROWS-by-MCOLS.
```

After MATLAB colour conversion to **YUV**/**YIQ**, For **U/I** and **V/Q** channels:

- Use `nearest` for **4:2:2** and **4:2:1** and scale the `MCOLS` to half or quarter the size of the image.
- Use `bilinear` (to average) for **4:2:0** and set **scale** to **half**.

Chroma Subsampling Example 1

chromasubsampling_eg1.m: Single iteration 4:2:0 Chroma subsampling example (Questions 4–6 Hints)

```
imRGB = imread('parrots.jpg');
figure,  imshow(imRGB), title('RGB Full Image');

imYIQ = rgb2ntsc(imRGB);

% Subsample the I and Q Channels  4:2:0 Type Subsampling
imYIQsubI = imresize(imYIQ(:,:,2),0.5,'bilinear');
imYIQsubQ = imresize(imYIQ(:,:,3),0.5,'bilinear');

% We have have size image so resample back up
imYIQupsampI = imresize(imYIQsubI,2);
imYIQupsampQ = imresize(imYIQsubQ,2);

reconstruct_imYIQ= imYIQ;  % Copy YIQ keep Y;
reconstruct_imYIQ(:,:,2) = imYIQupsampI;
reconstruct_imYIQ(:,:,3) = imYIQupsampQ;

% Remake RGB and show
reconstruct_imRGB = uint8(256*ntsc2rgb(reconstruct_imYIQ));
figure, imshow(reconstruct_imRGB); title('Reconstructed RGB Full Image');

% show  R,G,B plane errors (Amplified!)
figure, imshow(256*abs(imRGB(:,:,1) - reconstruct_imRGB(:,:,1))); title('Reconstructed R Error');
figure, imshow(256*abs(imRGB(:,:,2) - reconstruct_imRGB(:,:,2))); title('Reconstructed G Error');
figure, imshow(256*abs(imRGB(:,:,3) - reconstruct_imRGB(:,:,3))); title('Reconstructed B Error');
```

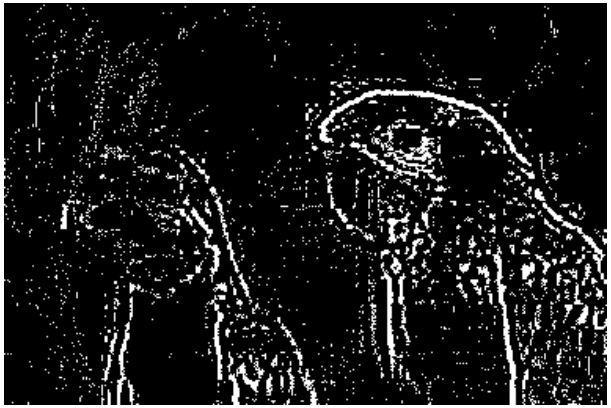
See also: chromasubsampling_eg2.m (0.125 ratio for effect).

chromasubsampling_eg1.m Output:

RGB Full Image

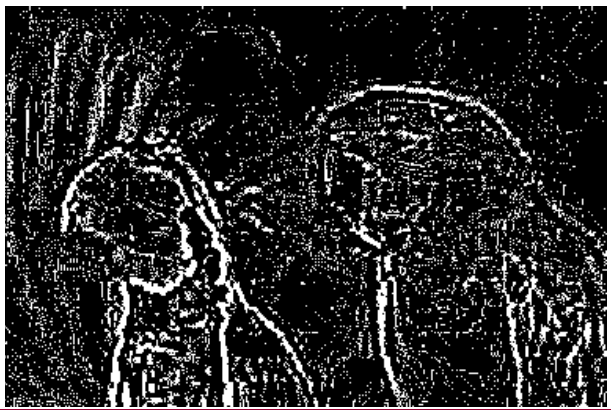
Reconstructed RGB Full Image

Reconstructed R Error



Reconstructed G Error

Reconstructed B Error



Chroma Subsampling Example 2

chromasubsampling_eg4.m (also chromasubsampling_eg3.m): Multiple iteration (1,000 times) 4:2:0 Chroma subsampling example

```
imRGB = imread('parrots.jpg');
figure,  imshow(imRGB), title('RGB Full Image');

imYIQ = rgb2ntsc(imRGB);

for i = 1:1000  % Simulate multiple copying 1000 times!
% Subsample the I and Q Channels  4:2:0  Subsampling
imYIQsubI = imresize(imYIQ(:,:,2),0.5,'bilinear');
imYIQsubQ = imresize(imYIQ(:,:,3),0.5, 'bilinear');

% We have have size image so resample back up
imYIQupsampI = imresize(imYIQsubI,2);
imYIQupsampQ = imresize(imYIQsubQ,2);
imYIQ(:,:,2) = imYIQupsampI;
imYIQ(:,:,3) = imYIQupsampQ;
end

% Remake RGB and show
reconstruct_imRGB = uint8(256*ntsc2rgb(imYIQ));
figure, imshow(reconstruct_imRGB); title('Reconstructed (1000 Iterations) RGB Full Image');

% show  R,G,B plane errors
figure, imshow(256*abs(imRGB(:,:,1) - reconstruct_imRGB(:,:,1)));
title('Reconstructed (1000 Iterations) R Error');
figure, imshow(256*abs(imRGB(:,:,2) - reconstruct_imRGB(:,:,2)));
title('Reconstructed (1000 Iterations) G Error');
figure, imshow(256*abs(imRGB(:,:,3) - reconstruct_imRGB(:,:,3)));
title('Reconstructed (1000 Iterations) B Error');
```

chromasubsampling_eg4.m Output:

RGB Full Image

Reconstructed (1000 Iterations) RGB Full Image

Reconstructed (1000 Iterations) R Error

Reconstructed (1000 Iterations) G Error

Reconstructed (1000 Iterations) B Error

Aliasing in Video: Temporal aliasing

Aliasing Explained:

- **'Strobing Effect'**: *e.g.*, rotating wagon wheel spokes apparently reversing direction,
  - See aliasing_wheel.m + spokesR.gif
- The **incorrect** sampling rate "**freezes**" the frames at the wrong moment

Frame 1 Frame2 Frame3 Frame4 Frame5

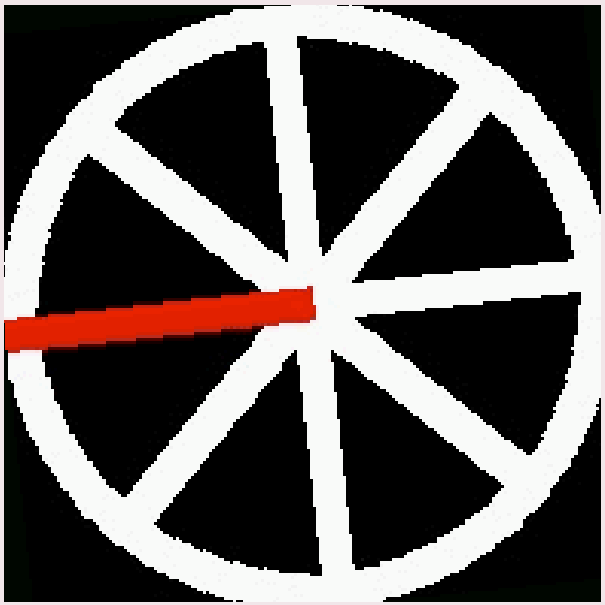
Aliasing in Video: Temporal aliasing (Cont.)

aliasing_wheel.m Code: above sampling frequency

```
sampfreq = 15;
rotfreq = 15;
rotstep= 360/rotfreq;

[im cmap] = imread('spokesR.gif');
[orign origm] = size(im);
offx = floor(orign/2);
offy = floor(orign/2);

% Create Movie of just 1 complete rotation of wheel --- NO SAMPLING ISSUE
% Effectively  NYquist sample frequency is 15*15 = 225 Hz way above rotation frequency

movie_wheel = avifile('aliasing_wheel_rot.avi', 'fps', 2,
     'compression', 'none', 'colormap', cmap);

for i = 0:rotstep:360
IMR = imrotate(im,-1*i);

[n m] = size(IMR);
centrex = floor(n/2);
centrey = floor(m/2);

IMR = IMR(centrex-offx +1:centrex+offx,centrey-offy + 1 :centrey+offy);

movie_wheel = addframe(movie_wheel,IMR);
end;

movie_wheel = close(movie_wheel);
```

Aliasing in Video: Temporal aliasing (Cont.)

Sampling at Above Nyquist Video Sample Rate



Above Nyquist Video

Click on image or links to see video.

Aliasing in Video: Temporal aliasing (Cont.)

aliasing_wheel.m Code: at sampling frequency

```
% Create Movie of rotating of wheel at sampling frequency
movie_wheel = avifile('aliasing_wheel_sampfreq.avi', 'fps', 2,
                      'compression', 'none', 'colormap', cmap);

rotstep = mod(360/(sampfreq/rotfreq),360)

for i = 0:15

rot = i*rotstep;
IMR = imrotate(im,-1*rot);

[n m] = size(IMR);
centrex = floor(n/2);
centrey = floor(m/2);

IMR = IMR(centrex-offx +1:centrex+offx,centrey-offy + 1 :centrey+offy);

movie_wheel = addframe(movie_wheel,IMR);
end;

movie_wheel = close(movie_wheel);
```

Aliasing in Video: Temporal aliasing (Cont.)

Sampling at Nyquist Video Sample Rate

At Nyquist Video

Click on image or links to see video.

Aliasing in Video: Temporal aliasing (Cont.)

aliasing_wheel.m Code: below sampling frequency

```
% Create Movie of rotating of wheel above sampling frequency

rotfreq = 29;

rotstep = mod(360/(sampfreq/rotfreq),360)

movie_wheel = avifile('aliasing_wheel_oversampfreq.avi', 'fps', 2,
                                'compression', 'none', 'colormap', cmap);
for i = 0:15

rot = i*rotstep;
IMR = imrotate(im,-1*rot);

[n m] = size(IMR);
centrex = floor(n/2);
centrey = floor(m/2);

IMR = IMR(centrex-offx +1:centrex+offx,centrey-offy + 1 :centrey+offy);

movie_wheel = addframe(movie_wheel,IMR);
end;

movie_wheel = close(movie_wheel);
```

Aliasing in Video: Temporal aliasing (Cont.)

Sampling at below Nyquist Video Sample Rate

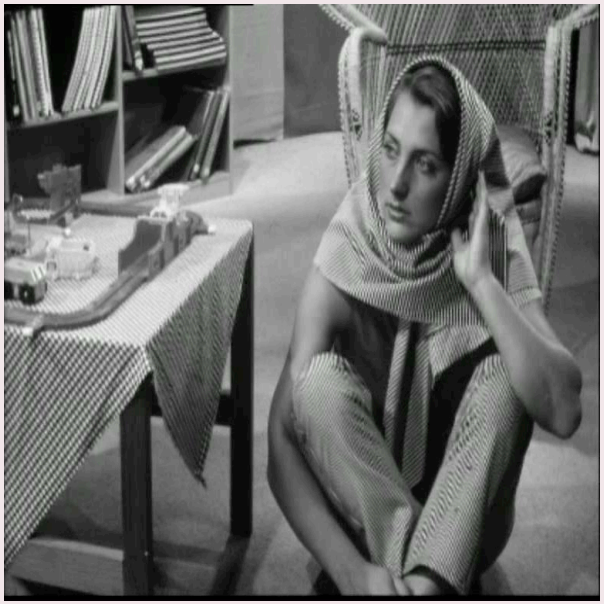
Below Nyquist Video

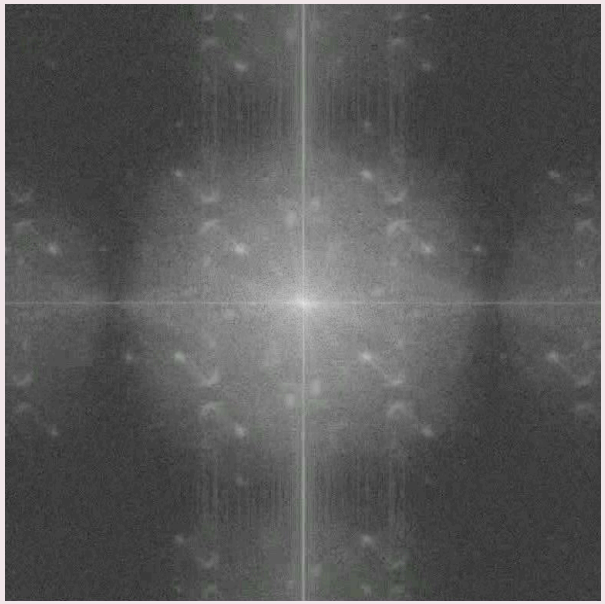
Click on image or links to see video.

Aliasing in Video: Raster scan aliasing

Raster scan aliasing:

e.g., twinkling or strobing effects on sharp horizontal lines, (see raster_aliasing.m + barbara.gif):





Strobing Alias Video

Strobing Alias Frequency Distributions Video

Click on image or links to see video.

Aliasing in Video: Raster scan aliasing (Cont.)

raster_aliasing.m Code:

```
f = imread('barbara.gif');
[ysize,xsize] = size(f);

mov_pics = avifile('aliasing_pics.avi', 'fps', 10, 'compression', 'none');
mov_specs = avifile('aliasing_specs.avi', 'fps', 10, 'compression', 'none');
for xshrink = 0:5:600
desiredxsize = xsize - xshrink;
scale_shrink = desiredxsize / xsize;
T = maketform('affine',[scale_shrink 0 0; 0 scale_shrink 0; 0 0 1]);
f2 = imtransform(f,T);
[currentysize, currentxsize] = size(f2);

scale_boost = xsize / currentxsize;
Tinv = maketform('affine',[scale_boost 0 0; 0 scale_boost 0; 0 0 1]);

f3 = imtransform(f2,Tinv,'size',[ysize xsize]);
Fd = fftshift(log(1+abs(fft2(f3))));

%imshow([f3/max(max(f3));Fd/max(max(Fd))]);
%imshow(f3);
xshrink
fr = im2frame(f3, gray(256));
Fdr = im2frame(uint8(256*Fd/max(max(Fd))), gray(256));
mov_pics = addframe(mov_pics, fr);
mov_specs = addframe(mov_specs, Fdr);
end
mov_pics = close(mov_pics);
mov_specs = close(mov_specs);
```