# Computational Photography

Siavash Bigdeli
University of Bern
Fall 2014

# Project 1

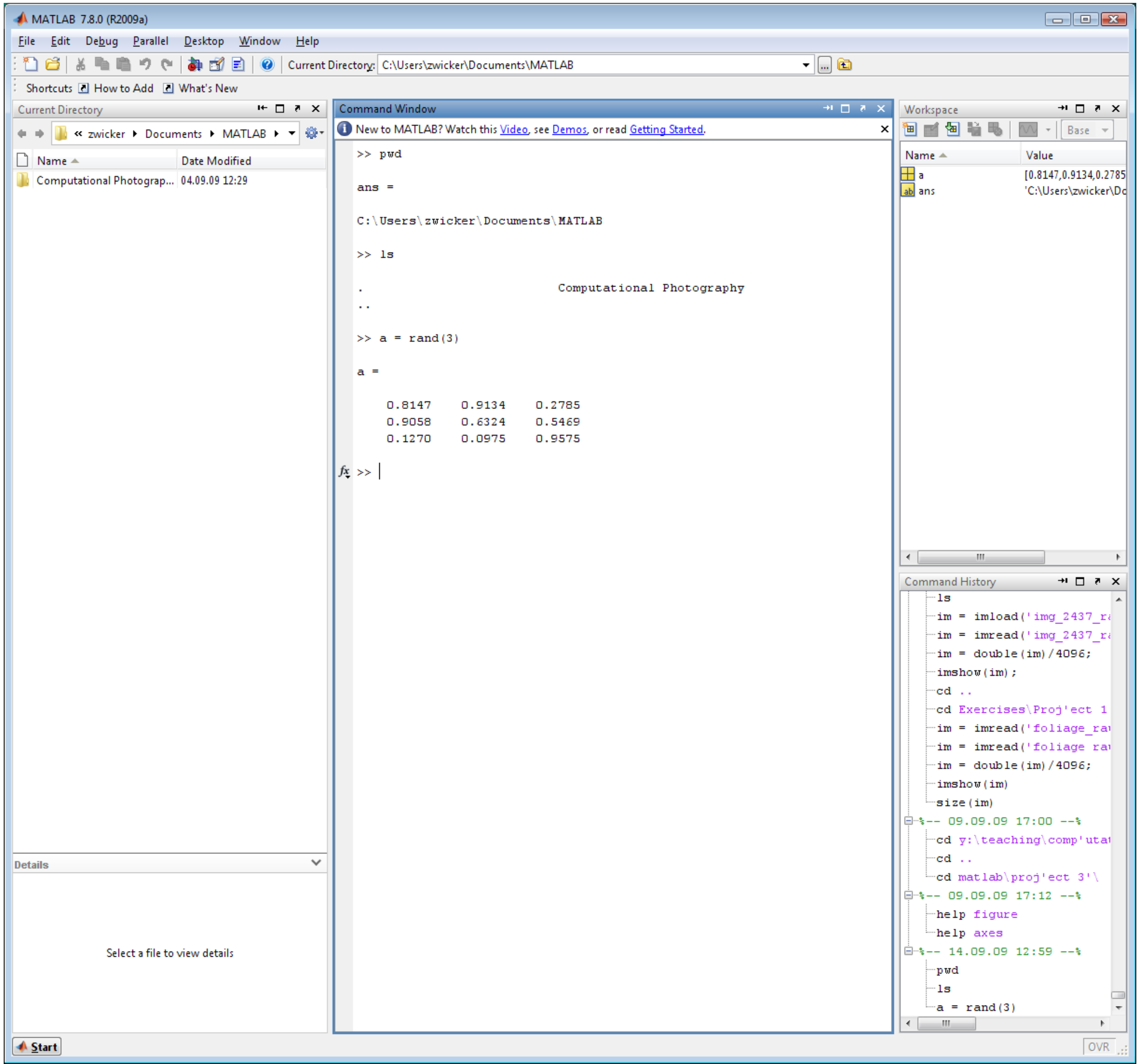
- Topics
  - Getting familiar with Matlab
  - Color images
- Assignmentes
  - „Spanish castle“ illusion
  - Bayer demosaicing
  - White balancing
  - Contrast scaling
- All material on Ilias
- Deadline: 2. October 2014, 10:00

## Matlab

- Easy-to-use rapid prototyping of numerical algorithms
  - Built-in functions for matrix computations
  - Useful functions for image processing
  - Flexible programming language
- Work through the tutorial

  http://www.mathworks.ch/academia/student_center/tutorials/launchpad.html
- Use the web forum for questions
  https://ilias.unibe.ch/goto_ilias3_unibe_frm_702918.html
- Installed in ExWi and CGG pool
  - CGG pool: compphoto14/cpuebung14

# Matlab


MATLAB 7.8.0 (R2009a)
File Edit Debug Parallel Desktop Window Help
Current Directory: C:\Users\zwicker\Documents\MATLAB
Shortcuts How to Add What's New
Current Directory
« zwicker ▸ Documents ▸ MATLAB ▸
Name
Date Modified
Computational Photograp... 04.09.09 12:29
Details
Select a file to view details
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
Workspace
Base
Name
Value
a [0.8147,0.9134,0.2785
ans 'C:\Users\zwicker\Do
Command History
Start
OVR

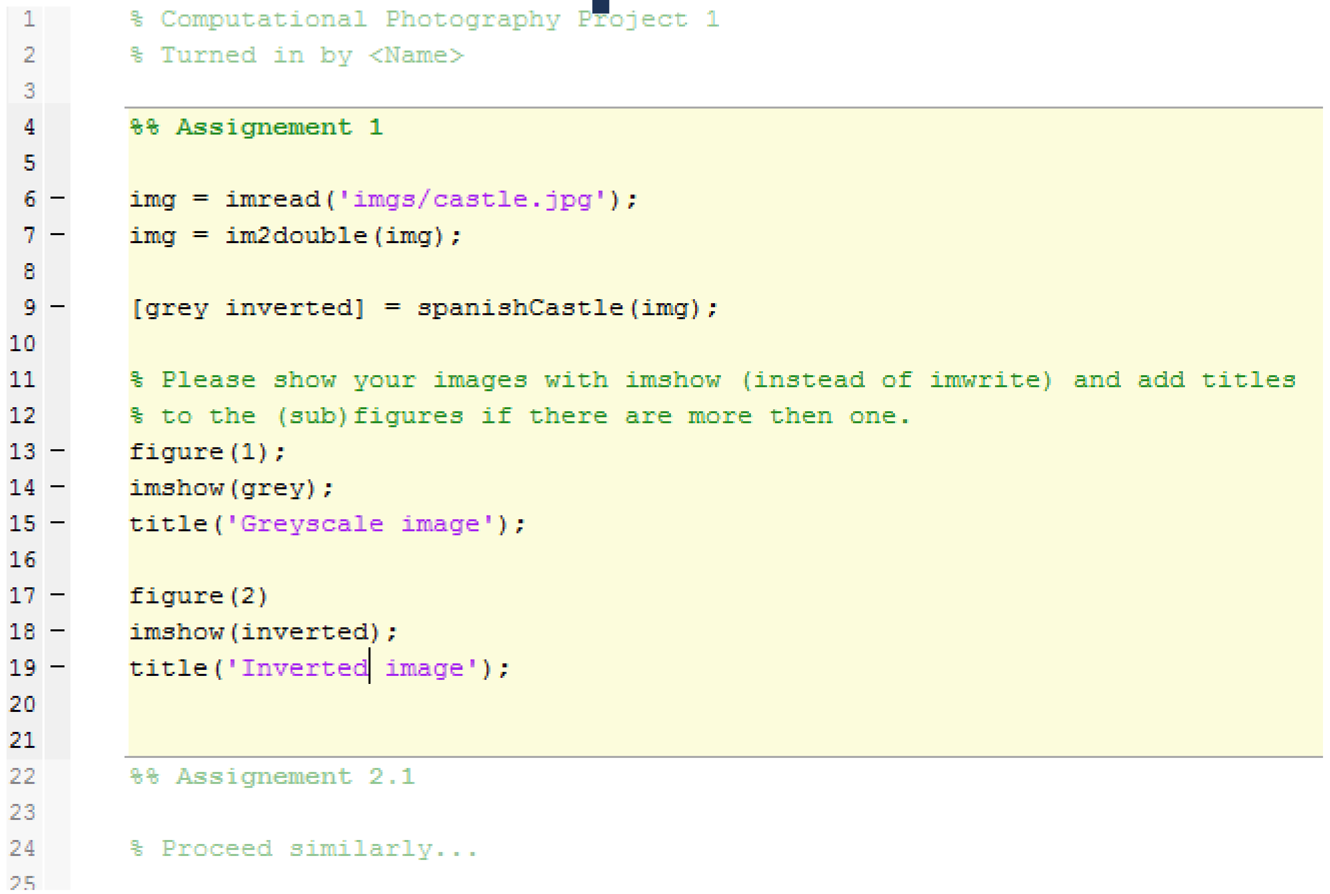
```
>> pwd

ans =

C:\Users\zwicker\Documents\MATLAB

>> ls

.                            Computational Photography
..

>> a = rand(3)

a =

    0.8147    0.9134    0.2785
    0.9058    0.6324    0.5469
    0.1270    0.0975    0.9575

>> |
```

```
ls
im = imload('img_2437_ra
im = imread('img_2437_ra
im = double(im)/4096;
imshow(im);
cd ..
cd Exercises\Proj'ect 1
im = imread('foliage_raw
im = imread('foliage raw
im = double(im)/4096;
imshow(im)
size(im)
%-- 09.09.09 17:00 --%
cd y:\teaching\comp'utat
cd ..
cd matlab\proj'ect 3'\
%-- 09.09.09 17:12 --%
help figure
help axes
%-- 14.09.09 12:59 --%
pwd
ls
a = rand(3)
```

# Turn-in guidelines

- Upload solutions to ILIAS
- Turn-in out-of-the-box runnable code
  - Including sample images
  - Don‘t forget external saved functions
  - Don‘t forget external saved matrices
  - We‘ll press crtl+Enter, then it has to run!
- Make one „start file“
- Please write `%comments`
- Please use `imshow` instead of `imwrite`

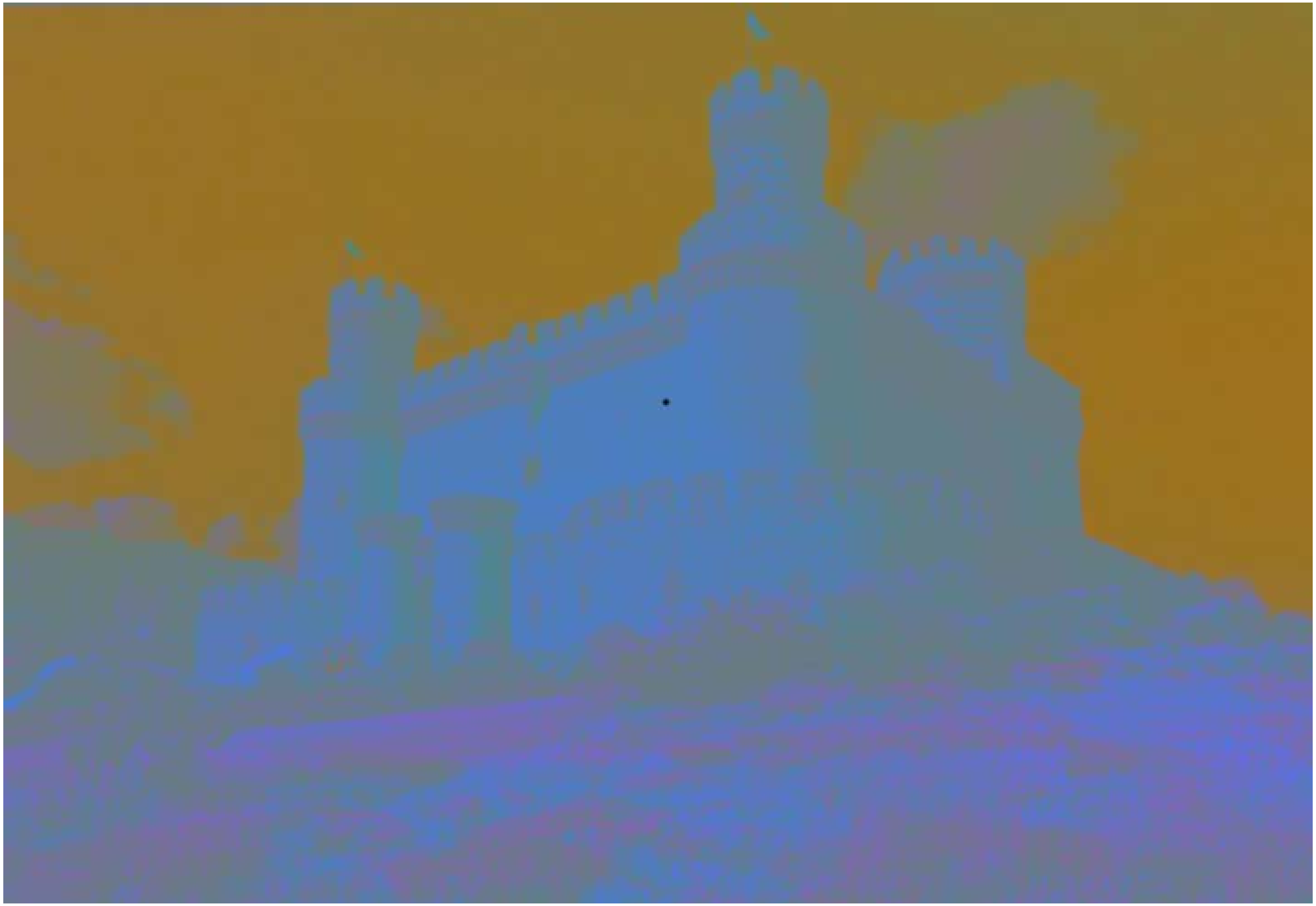
# Start file example



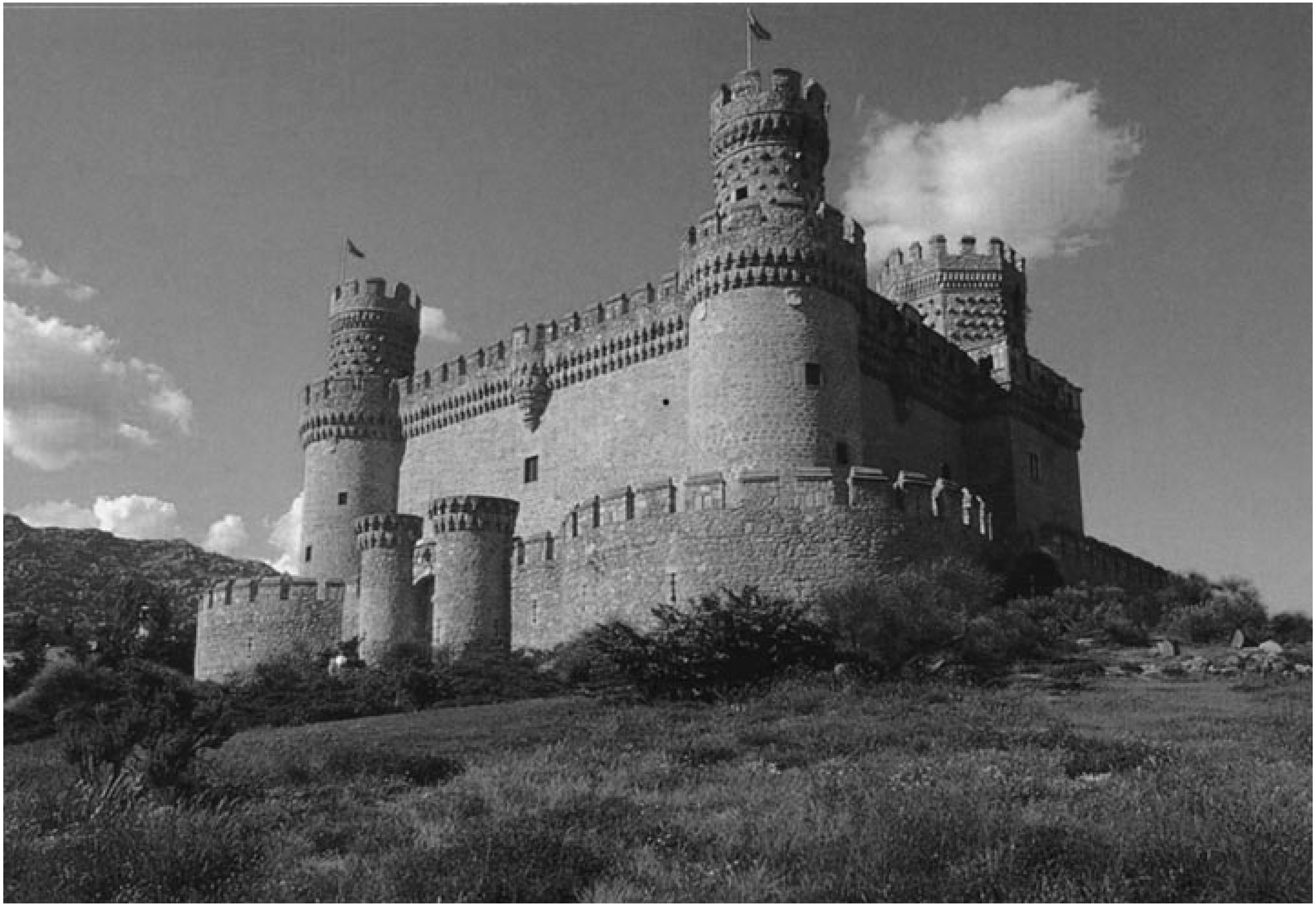
```
% Computational Photography Project 1
% Turned in by <Name>

%% Assignement 1

img = imread('imgs/castle.jpg');
img = im2double(img);

[grey inverted] = spanishCastle(img);

% Please show your images with imshow (instead of imwrite) and add titles
% to the (sub)figures if there are more then one.
figure(1);
imshow(grey);
title('Greyscale image');

figure(2)
imshow(inverted);
title('Inverted image');


%% Assignement 2.1

% Proceed similarly...
```

- Create cells using %%
- Each output in a separate (sub)figure
- Give figures titles if there are more than one

## Spanish castle illusion



# Spanish castle illusion



## Spanish castle illusion

- Based on opponent colors
  - Red-green and yellow-blue differences
- Use YUV colors here for convenience
  [http://en.wikipedia.org/wiki/YUV](http://en.wikipedia.org/wiki/YUV)
  - Luminance Y, brightness information
  - Chrominance U and V, color information
  - Important in analog TV and video
- Conversion from „standard“ RGB

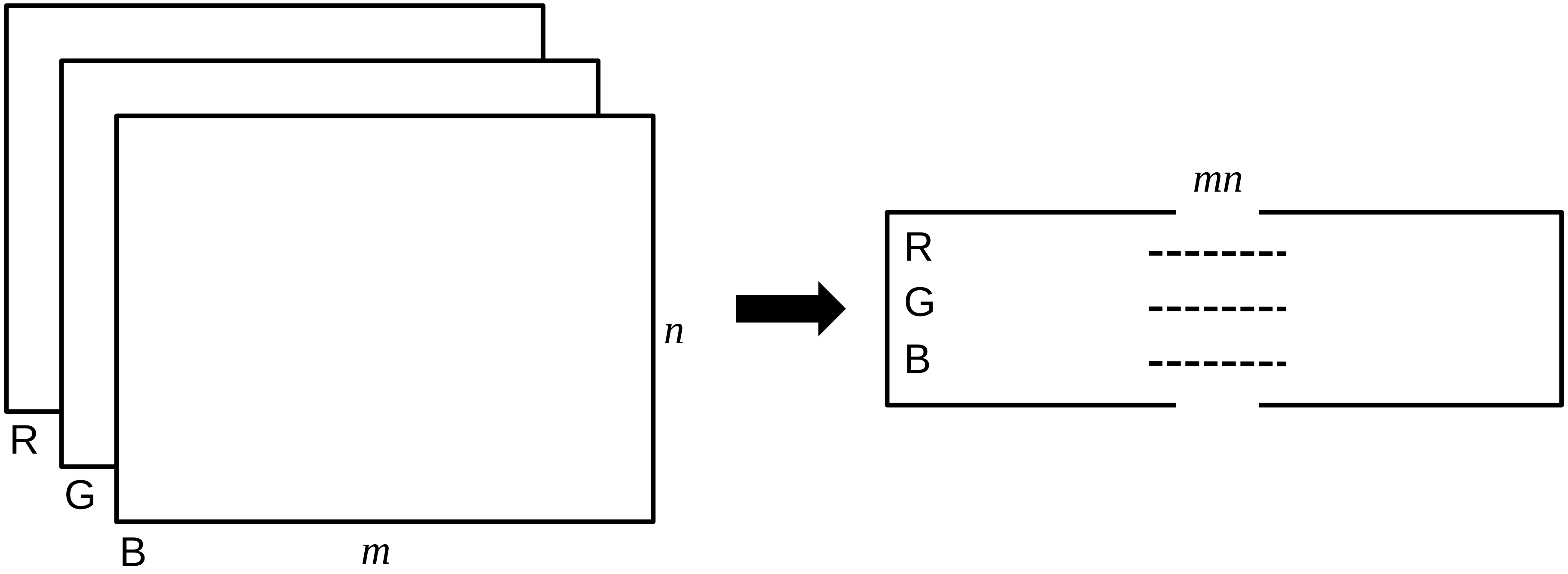
$$\begin{bmatrix} Y' \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.14713 & -0.28886 & 0.436 \\ 0.615 & -0.51499 & -0.10001 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

## Spanish castle illusion

- Use Y channel as grey image
- Create inverted image
  - Set Y channel to 0.6 for **all** pixels
  - Convert this „fake YUV image“ back to an RGB image.

# Color transformation

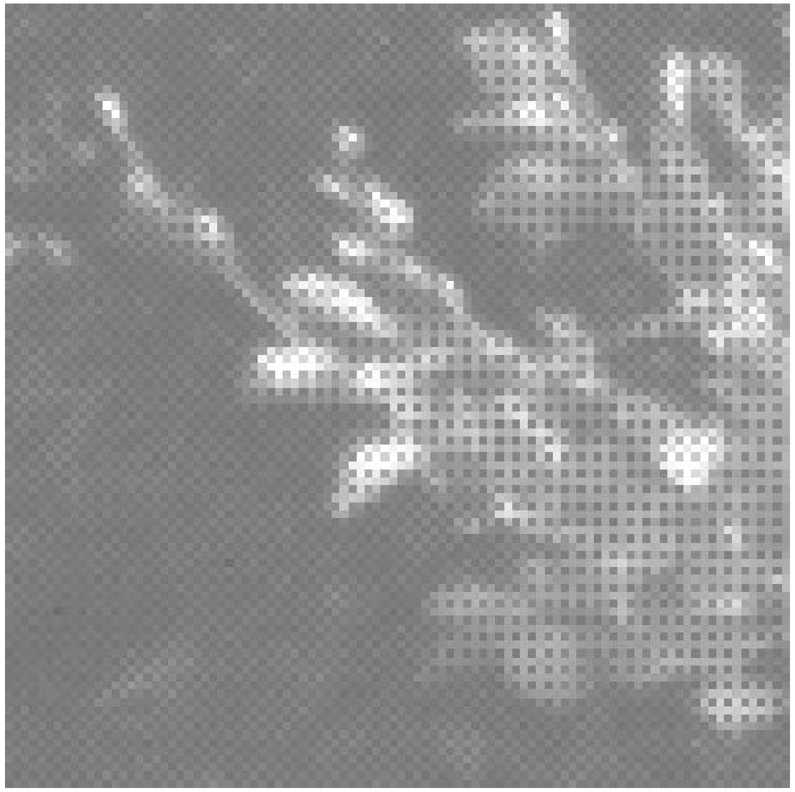
- Without `for` loops in Matlab!
- Reorganize matrix using `reshape`, `shiftdim`
- Transform colors using single matrix-matrix multiplication, then reorganize matrix again

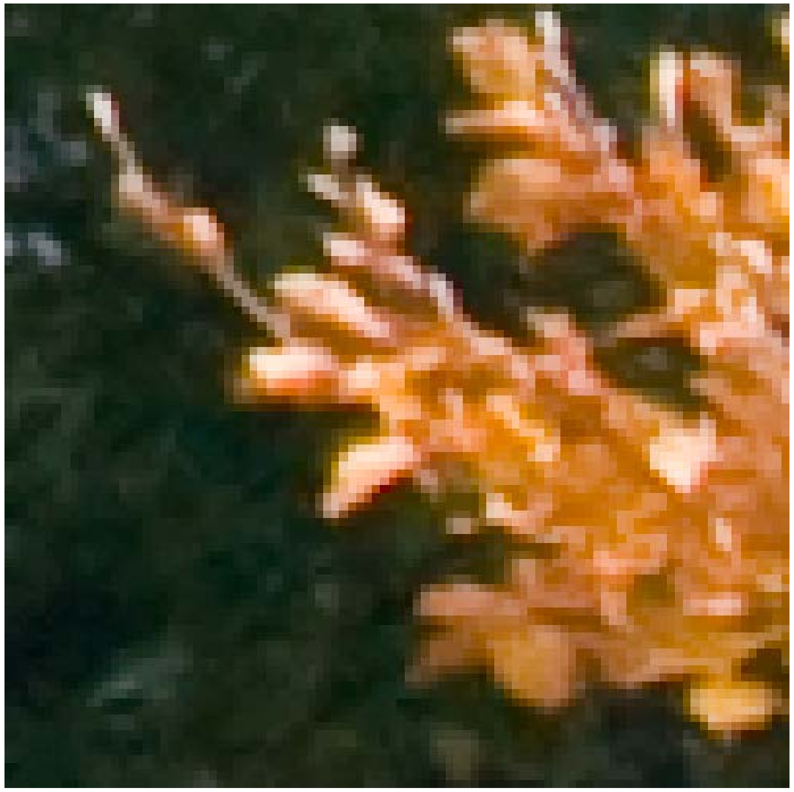


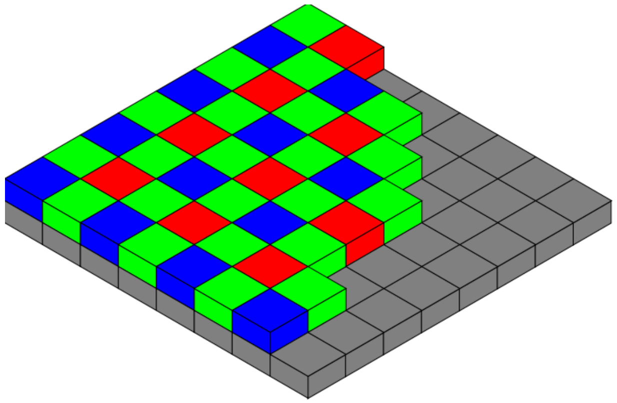
## Proposed schedule

- Week 1
  - Getting familiar with Matlab, Tutorial
  - Spanish castle
- Week 2
  - Demosaicing
  - White balancing
  - Contrast scaling

# Demosaicing







Interpolated
„de-mosaiced“

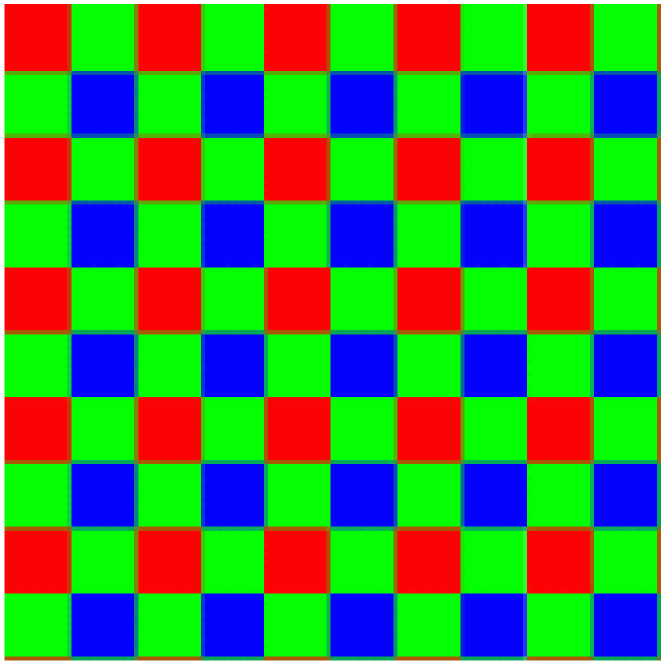
## Demosaicing

- Linear interpolation
- Median filtering

## Linear interpolation

- Mask for sampling grid, i.e., red pixels

```
rMask = zeros(m,n);
rMask(1:2:end,1:2:end) = 1;



rMask =
     1     0     1     0     1
     0     0     0     0     0
     1     0     1     0     1
     0     0     0     0     0
     1     0     1     0     1
     ...
```


Grid for example file

## Linear interpolation

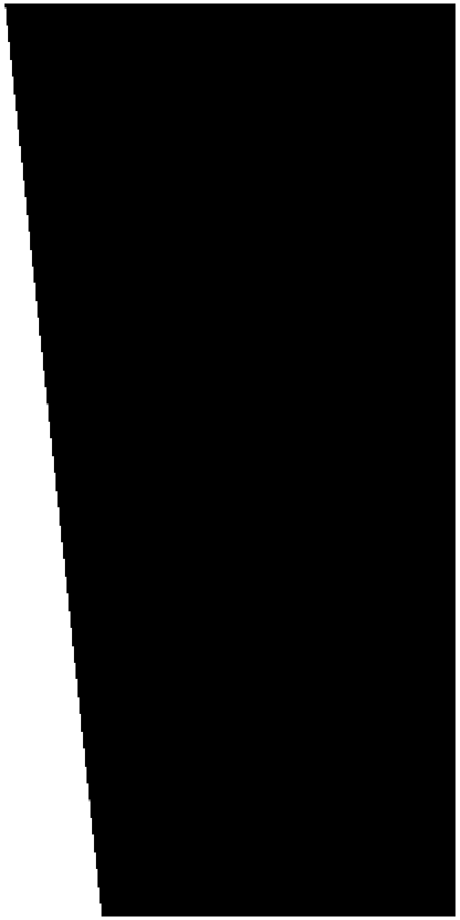
- Perform linear interpolation using convolution
  - Convolution is local weighted averaging
  - `conv2` in Matlab
  - 3x3 convolution kernel consisting of 1‘s
  - Need to divide through sum of weights

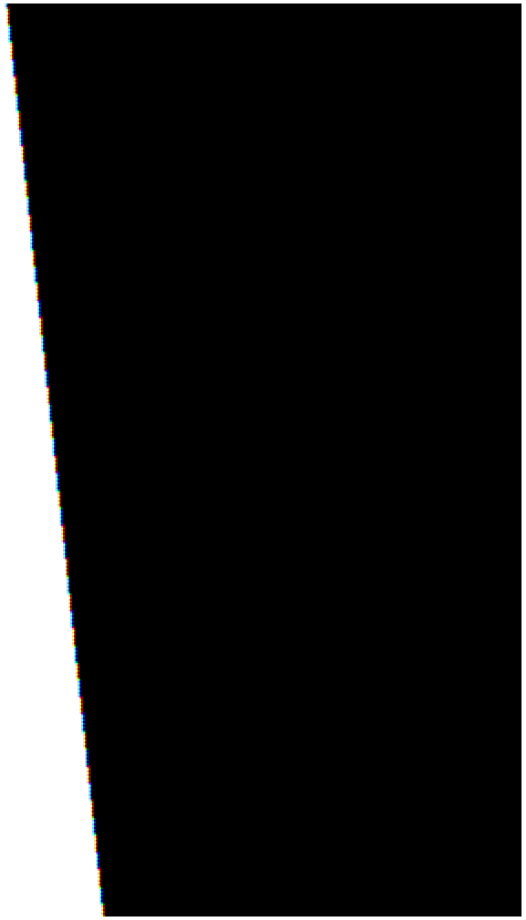
```
K = ones(3);

rInterp = conv2(in.*rMask, K, 'same');
rInterp = rInterp./conv2(rMask, K, 'same');
```

- Note: final result should keep original data, only fill in holes

# Linear interpolation

- Color fringing artifacts


Input (using Bayer pattern)


Linear demosaicing

## Median filtering

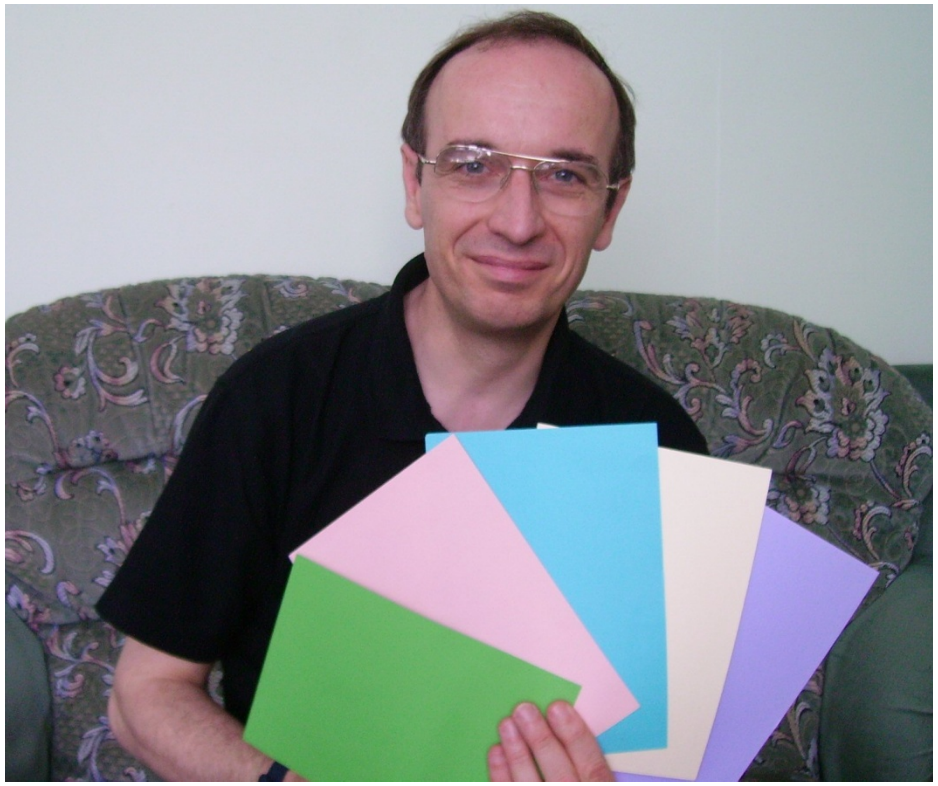
1. Linear interpolation as before
2. Transform to YUV
3. Median filter U and V
   – Use `medfilt2` in Matlab
4. Transform back to RGB
   – Invert the YUV transformation
5. Assemble output

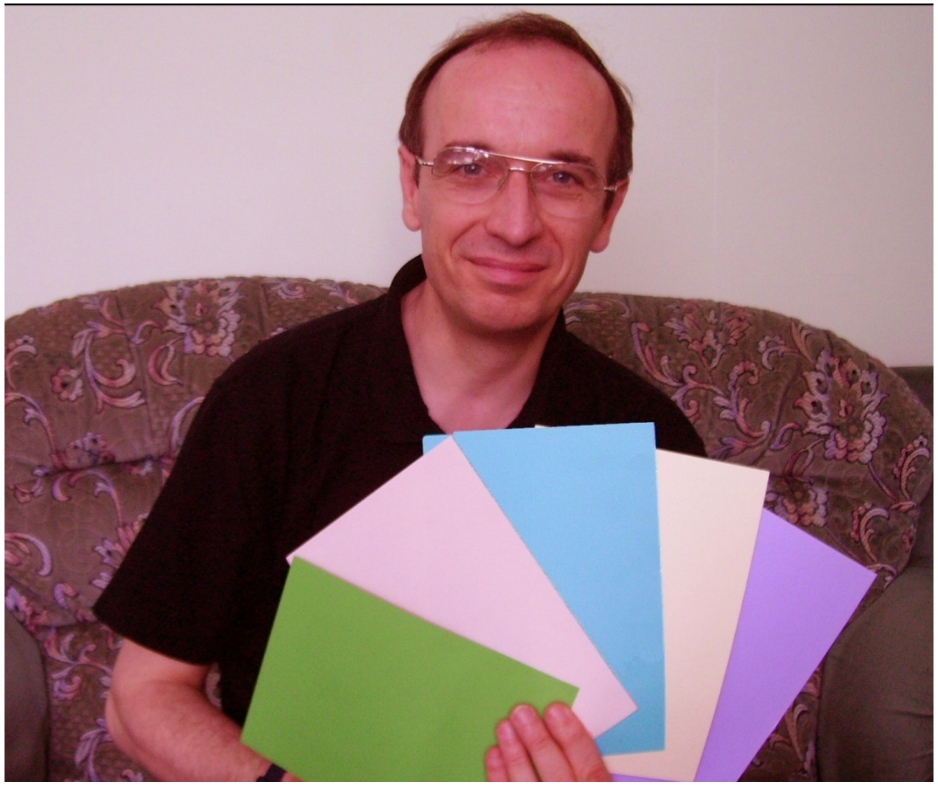
## Color balancing

### Color constancy

- Human color perception ensures that perceived color of objects remains relatively constant under varying illumination conditions
- White object under reddish illumination is captured as a red object by a camera, but perceived as white by human observer
- White balancing removes red tint from captured image to reproduce the white object as it is perceived, which is white

# Color constancy





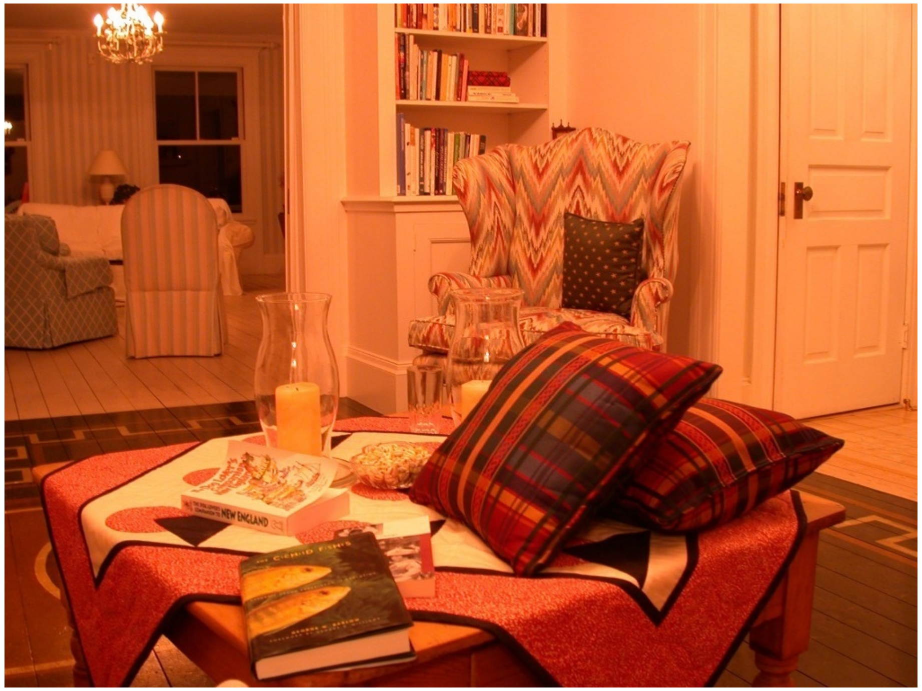
The second card from the left seems to be a stronger shade of pink in the top picture. In fact they are the same color, but the brain changes its assumption about color due to the color cast of the surrounding photo.

http://en.wikipedia.org/wiki/Color_constancy

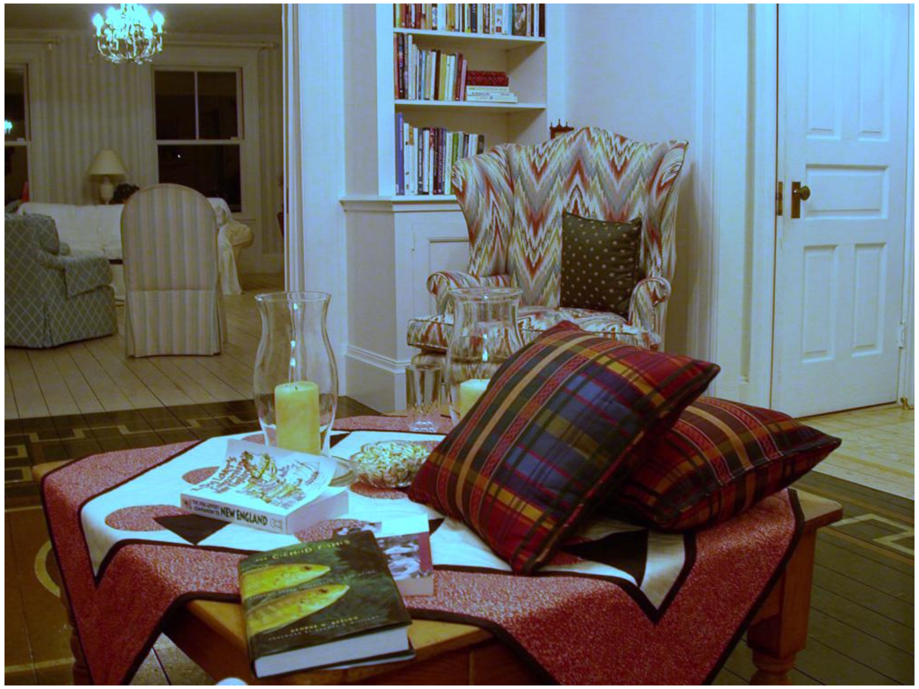
## Color balancing

- Known illuminantion
  - Divide out color of light source
  - Tungsten (light bulb), fluorescent, daylight,…
- Manual white balance
  - Pick pixel that is known to be shade of grey
  - Scale color channels to make it grey
  - Take extra picture of grey object if necessary
- Auto white balance
  - Different heuristics
  - Grey world assumption: average color is grey

# Color balancing



Input



Grey world white balance

# Grey world assumption

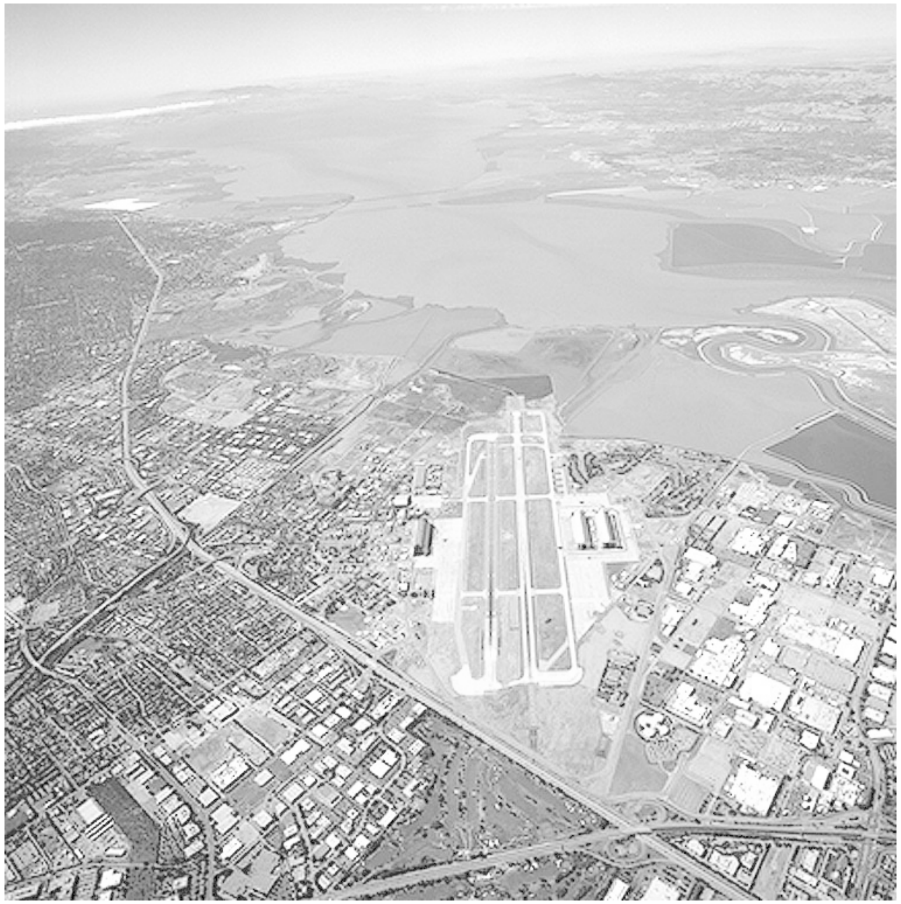
- Compute average of matrix using `mean`
  - Apply twice, i.e., `mean(mean(M))`
  - Similar functions `min`, `max`
- Scaling factors

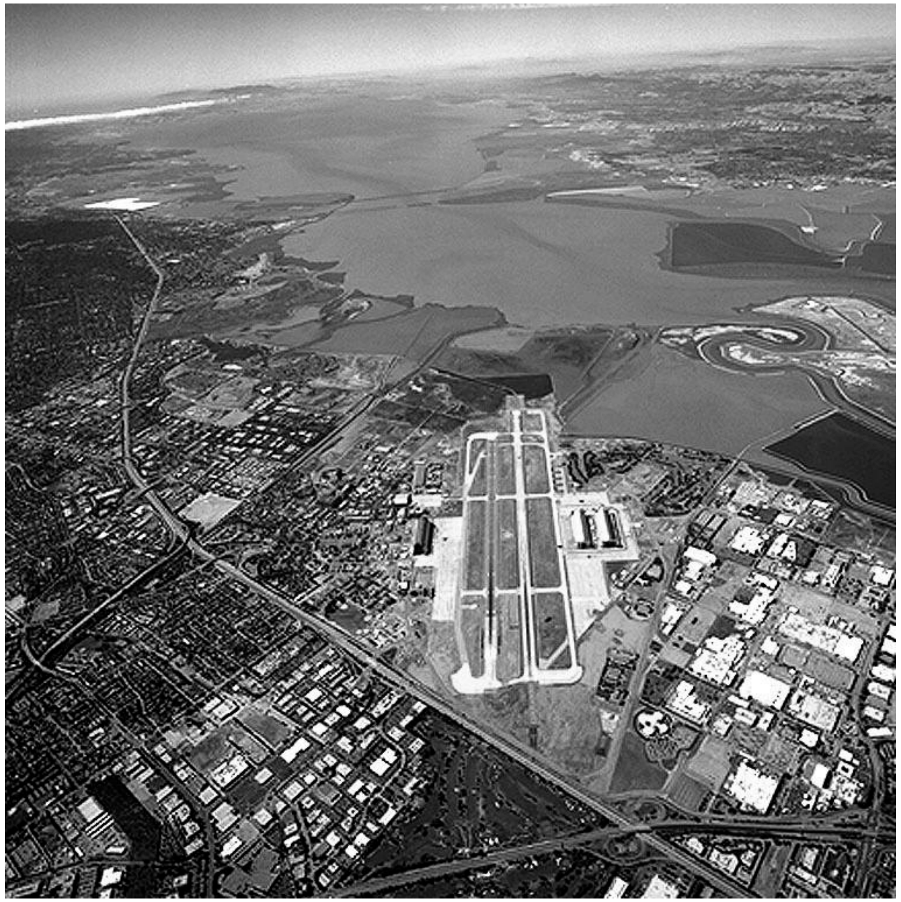
```
greyValue = (avgR + avgG + avgB)/3;
scaleR = grayValue./avgR;
scaleG = grayValue./avgG;
scaleB = grayValue./avgB;
```

- Normalizing scaling for green to one yields

```
greyValue = (avgR + avgG + avgB)/3;
scaleR = scaleR./scaleG;
scaleG = scaleG./scaleG; % equal to 1
scaleB = scaleB./scaleG;
```

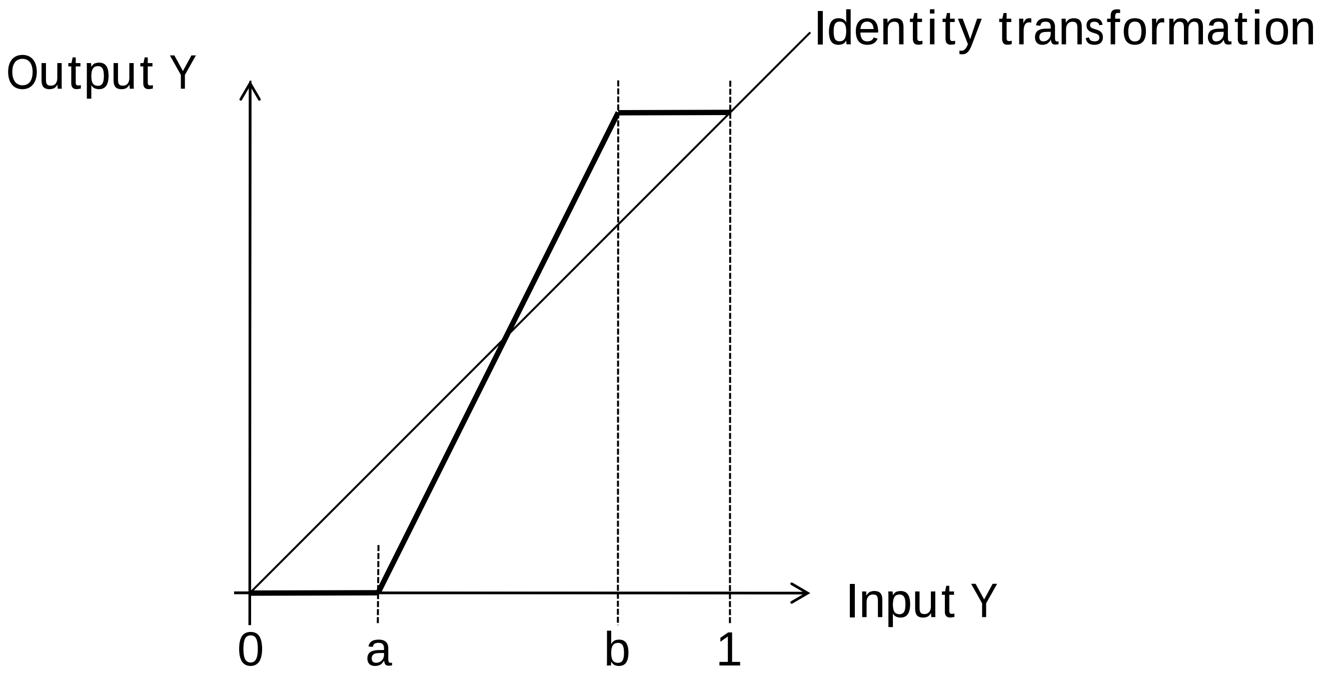
# Global Contrast Adjustment


Input

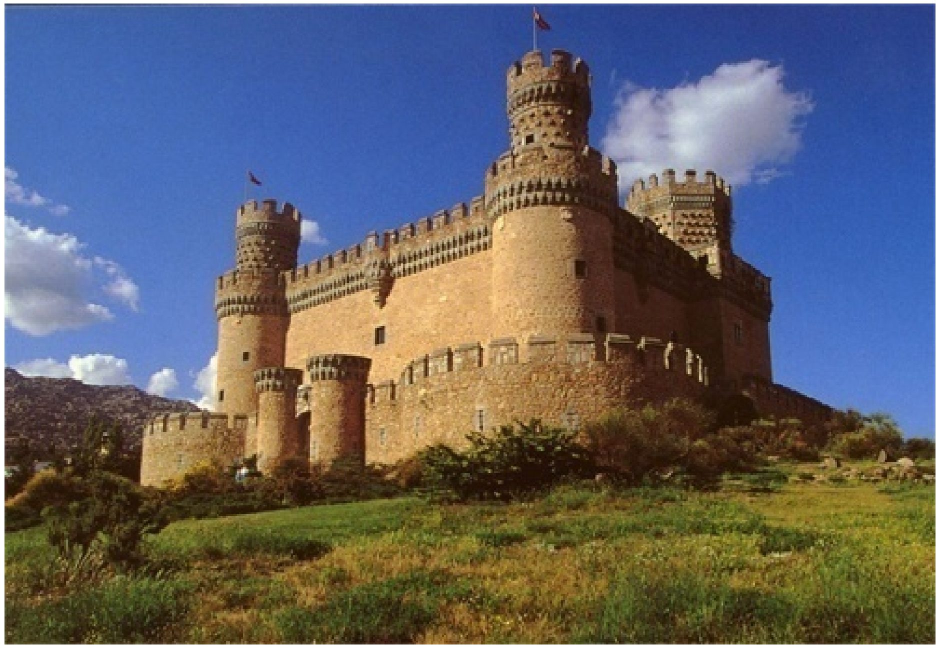

Contrast Adjusted

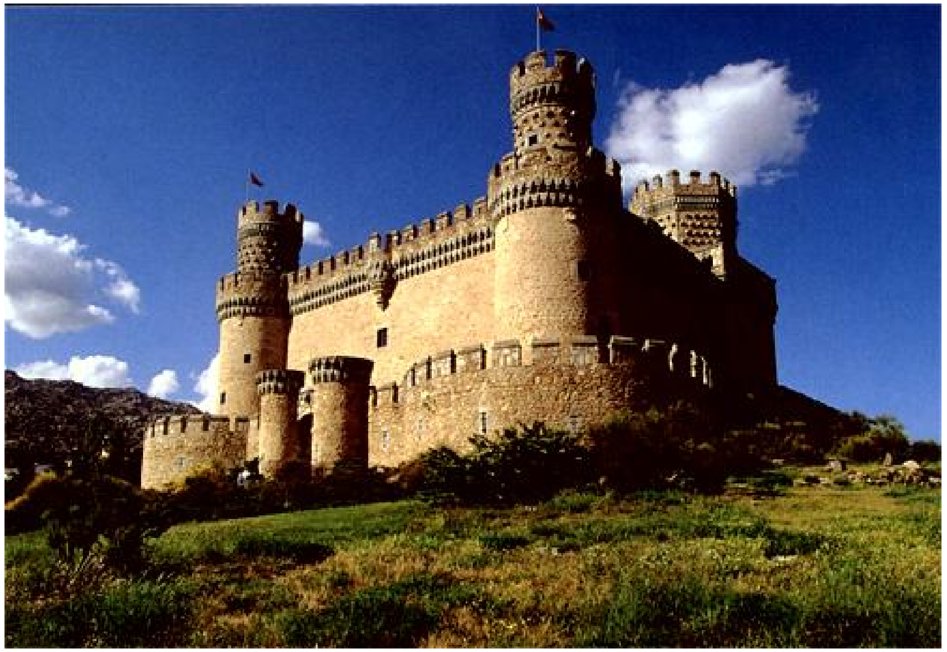
## Linear contrast scaling

- Transform color image to YUV, operate on Y
  - Assume Y in [0,1]
- Find function that maps range [a,b] to [0,1]
  - Usually $0 < a < b < 1$



# Linear contrast scaling

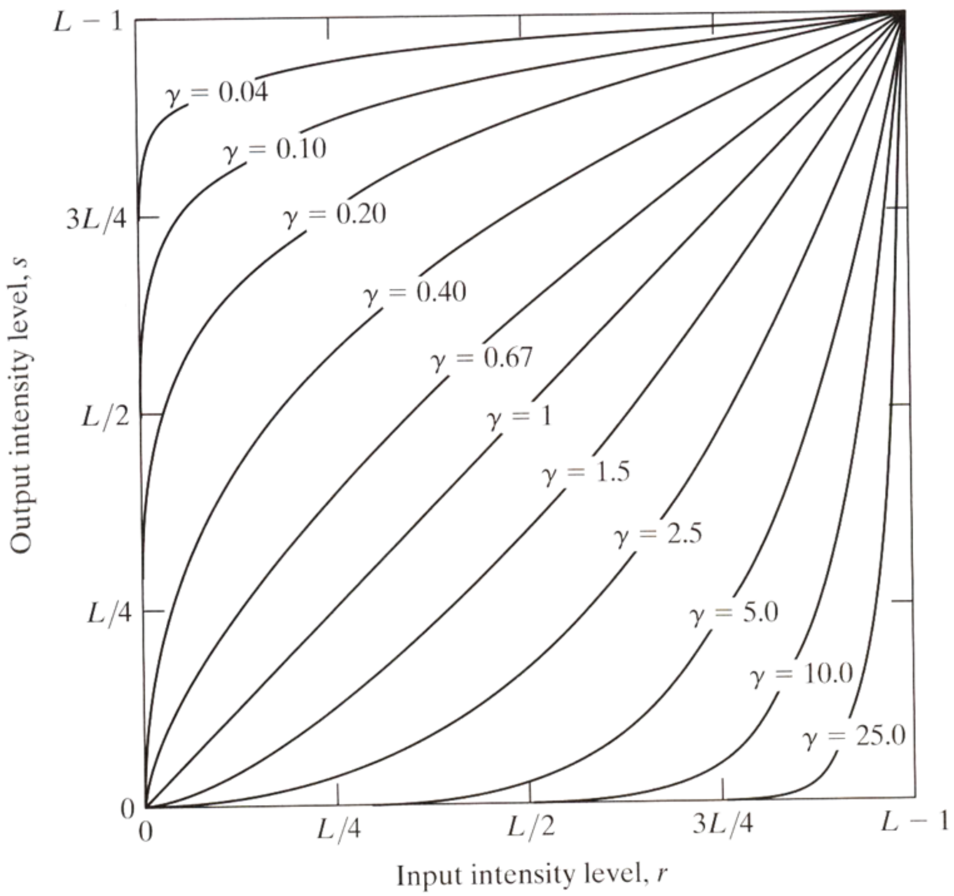

Input


Scaled [0.2,0.8] to [0,1]

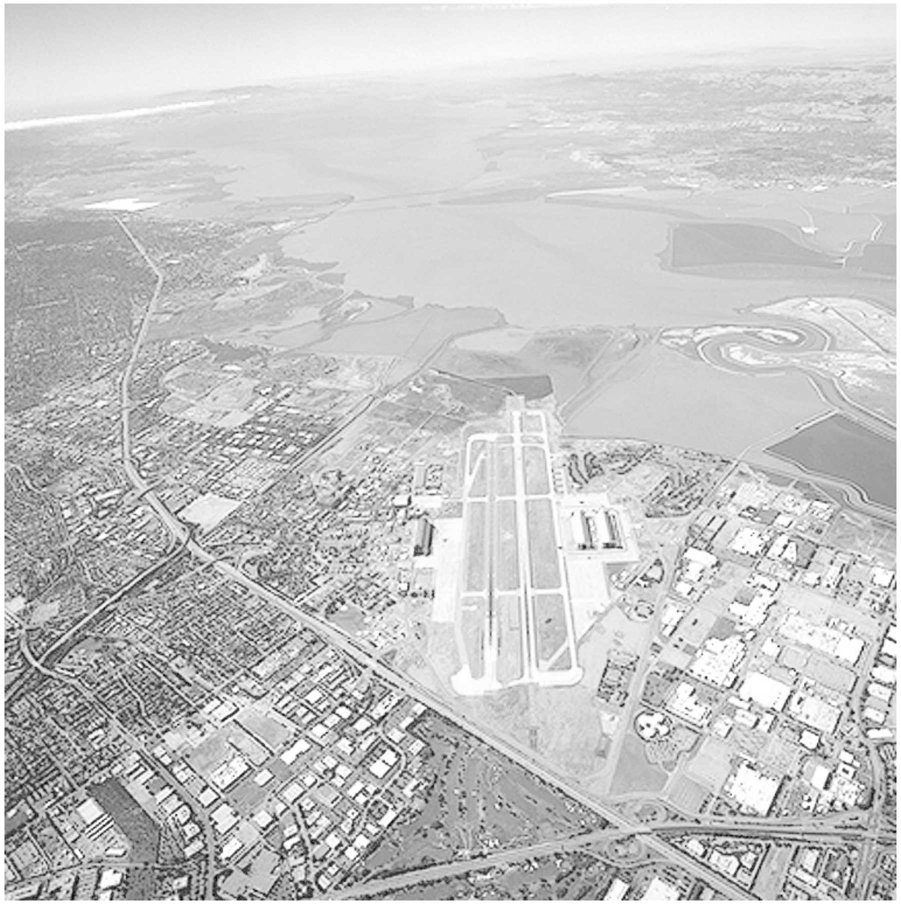
## Gamma transformation

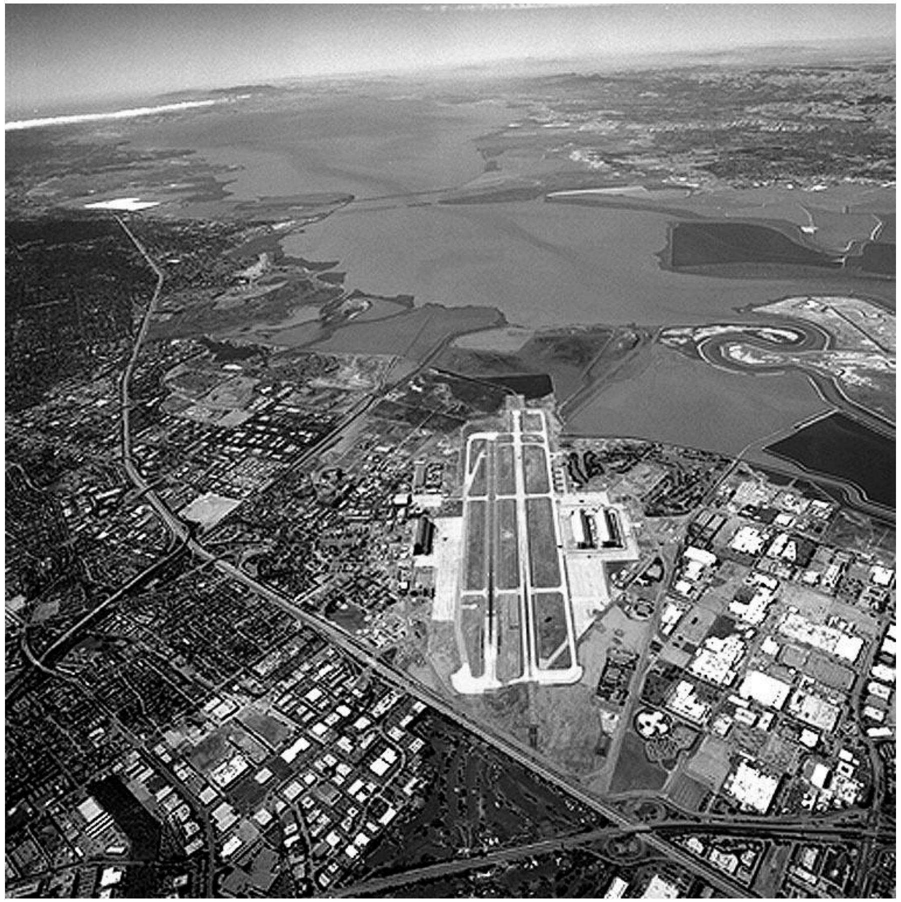
- Family of transformations that allows contrast compression or stretching

$$\text{output} = \text{input}^{\gamma}$$



# Gamma transformation


Input


Transformed with $\gamma$=4

## Gamma correction

- Cameras are (more or less) linear
  - Sensor response proportional to light energy
- Displays are (in general) non-linear
  - Produced light energy not proportional to input electrical energy
- Gamma correction is to obtain best perceptual performance from limited number of bits in each RGB component
  http://en.wikipedia.org/wiki/Gamma_correction
  - Typical display gamma is 2.2
- Usually performed in camera and encoded in jpg image
  - But RAW camera images are linear (not gamma corrected)