

VIDEO CODING TECHNOLOGIES

MATLAB-PROYECT



Martina Eckert
Universidad Politécnica de Madrid
Spring 2013



MATLAB PROJECT

- Step by step implementation of video coding functions with the aim to create a reduced video coder.
- **Sessions:**
 1. Getting started: get familiar with Matlab if you don't have much experience and implement some elementary functions.
 2. Implement some basic image processing functions based on the knowledge obtained in topic 2.
 3. Start the video coder project part I – Intra coding
 4. 2nd session of part I – Intra coding
 5. Video coder project part II – Inter coding ????????

If there is not enough time, I'll give instructions to go on alone



MATLAB PROJECT

General advices

1. Implementation of the **demanded functions**:

- Strictly use the indicated header (same name, same number of in- and output values, same order of input values)
- Explain the function with comments in **English**

– **Example:**

```
function image = read_image()  
% Permits to select an image file from a directory and reads the image  
% Input: none  
% Output: image matrix
```

2. Personal **test-programs** (only for you)

- For testing the demanded functions
- Start every test program with:
 - **Clear** (deletes all global variables)
 - **Close** (closes all windows)

3. Send files to professor for evaluation

- Include all **demanded** files in one zip-file (no images, no personal test programs)
- Code has to be executable (if not, I will not search for errors!!)



Session 1

GETTING STARTED



GETTING STARTED

Study a **Matlab primer** if necessary and implement the following functions (off class):

- `get_luma.m`
- `plot_luma_line.m`
- `plot_rgb_line.m`
- `rgb_to_components.m` *
- `components_to_rgb.m` *

* Required functions for the coder-project



GETTING STARTED

Reading an image

- Image format: **uint8**
 - Read as **uint8** → convert to **double** for processing → reconvert to **uint8** for presentation and storage
 - Use **isa(OBJ, 'classname')** for checking format: returns **true** if OBJ is an instance of 'classname'.
 - Example:

```
if ~isa(image, 'uint8')
    error('Error: file is not an uint8 image');
end
```

- Matlab functions:
 - **Imread**, **imshow**, **imwrite**
- Colon operator: **(:)**, e.g.: **A(:, :, k)** is the k^{th} page of three-dimensional array A



GETTING STARTED

Reading an image

- Provided function: `read_image()`
 - Permits to select an image file from a directory and reads the image

```
function image = read_image()  
% Permits to select an image file from a directory and reads the image  
% Input: none  
% Output: image matrix  
[file, path] = uigetfile( {'*.gif;*.jpg;*.bmp;*.png;*.jpg', ...  
    'Image files (gif, jpg, bmp, png, jpg)'}, ...  
    'Open image file');  
image = imread([path file]);
```

- Image size:
 - `Size()` (size of array)
 - for M-by-N matrix X, returns the two-element row vector [M,N] containing the number of rows and columns in the matrix.
 - For N-D arrays, SIZE(X) returns a 1-by-N vector of dimension lengths.
 - Example: `[height, width, component] = size(image_rgb);`



get_luma ()

- Implement the function `get_luma` which calculates Y from R, G, B and returns it as a single component image:
 - Header: `function Y = get_luma(image_rgb)`
 - Convert test image `image_rgb` to `double`
 - Extract the luma component Y (apply formula from Recommendation BT.470, slide 41)
 - Convert Y to `uint8`
- Implement a test_program using the function in the following way:
 - Show luma image on the display
 - Save the luma image to disk
- Test the program with the images **bars.bmb** and **gromit.bmp** and save the output images with appropriate names

plot_luma_line()

- Implement the function `plot_luma_line`: which extracts and presents one single line of a luma image and shows it as a plot:

```
function plot_luma_line(Image_Y, n)
```

- The values have to be scaled to 0-1 (convert previously to `double`)
- Implement a test_program using the function in the following way:
 - Get line number from user with Matlab function `input()`.
 - Test the program with the output resulting from **bars.bmp**
 - Save the resulting plot as '*Enhanced Meta File*' with name **bars_luma_line.emf**
 - Get another line from the image **gromit.bmp** and save the result.

plot_rgb_line()

- Implement the function `plot_rgb_line`: which extracts and presents the same image line from the three components R,G,B:

```
function plot_rgb_line(Image_rgb, n)
```

- Represent the three lines in the same window using `subplot()` with appropriate titles
- The values have to be scaled to 0-1
- Represent the lines in colors
- Extend the vertical range using `axis()` (e.g. between -0.1 and 1.1) to have a better result
- Test the function as before with the image **bars.bmp**
- **Answer the following questions:**
 - Which amplitude and saturation does the bars have?
 - Which is the difference to the figure presented in the standard N-20?
 - How are the bars expressed in standard 471?

TRANSFORM RGB TO YCbCr

1. Implement the function `rgb_to_components` which returns the three components Y,Cb,Cr:

`function [Y, Cb, Cr] = rgb_to_components(Image_rgb)`

- Calculate in `double`, but don't reconvert to `uint8` as the result values are not integers!
2. Implement the function `plot_ycbcr_line`, analogue to the former ones.
 - Use black for Y, red for Cb and blue for Cb
 3. Implement a test program which
 - reads an RGB image and converts it to a component image
 - Reads a line number from the user and represents it
 4. Try the test program with the image **bars.bmp** and compare the results with standard N-10. Reason the differences as before.



INVERSE TRANSFORM: FROM YCbCr TO RGB

1. Implement the function `components_to_rgb` which returns again an RGB-image:

`function image_rgb = components_to_rgb(Y,Cb,Cr)`

- Use Matlab to calculate the inverse matrix M^{-1} , assigning to `M` the values used in `rgb_to_components`.
 - Remember to convert the result to `uint8`
2. Write a test program which:
 - Reads an RGB image and represents it in a window
 - Converts the image to components
 - Converts the components to another RGB image
 - Represents the new image in a different window
 - **Both windows should show identical content**

REPORT

Write a short report about your tests. **The report should contain:**

1. Your Name
2. An appropriate title
3. Very short descriptions about the performed tests (not describing the functions, but the aim and the results obtained. **Answer the questions and show the obtained images and figures.**)

Note: each figure should have a title below

4. Some conclusion (what did you learn, what difficulties did you have)



EVALUATION

Compress the **required functions** and the **report** in a zip-file and send per **e-mail** to the professor (martina.aux@gmail.com). Do not add your personal test programs and no images, only:

- `get_luma.m`
- `plot_luma_line.m`
- `plot_rgb_line.m`
- `rgb_to_components.m`
- `components_to_rgb.m`

Give the zip-file your NAME!!
(example: martina_eckert_1.zip)



**Until latest
Thursday
25th**