Augmented reality: Lesson 1 Introduction, Colour encoding, Colour space transformations, Basic filtering

Francesco Di Giacomo

Augmented reality course
Francesco Di Giacomo
giacf@hr.nl

Tentative course schedule

- **Lesson 1**: Introduction, Colour encoding, Colour space transformations, Basic filtering.
- **Lesson 2**: Practicum on filters.
- **Lesson 3**: Interpolation
- **Lesson 4**: Practicum on interpolation
- **Lesson 5**: Vectors and operations on Vectors. Data clustering, K-Means algorithm, Application to Image Segmentation.
- **Lesson 6**: Omnidirectional cameras + practicum on K-Means.
- **Lesson 7**: Practicum on omnicams + K-Means.

Examination

Assignments based on course topics + oral check. At the oral check you will be asked questions about the assignments and to complete simple exercises covering course topics. See the Modulewijzer for further information about the grading.

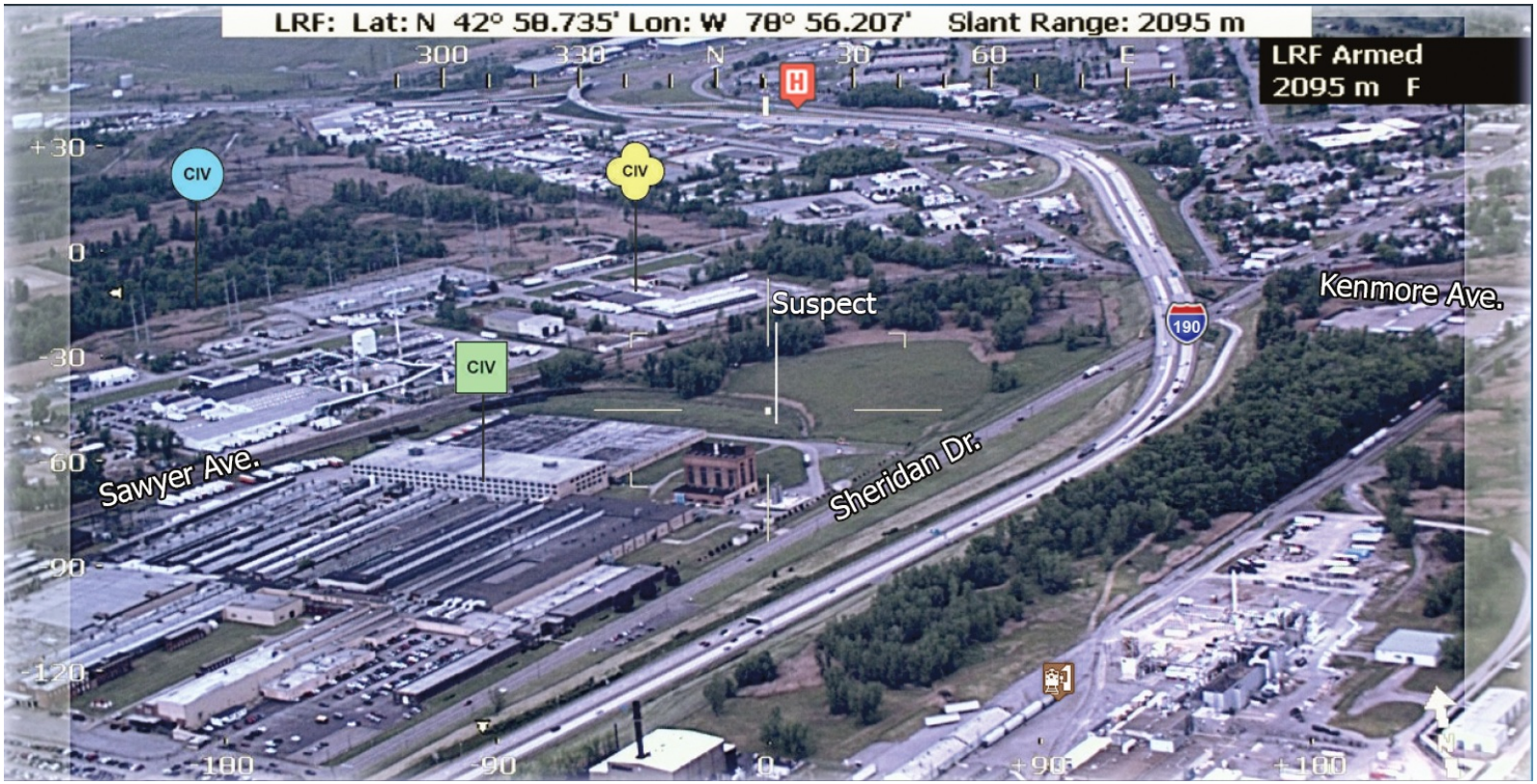
Assignments

The assignments are demanding! Do not start doing them the day before the deadline!

What is augmented reality?

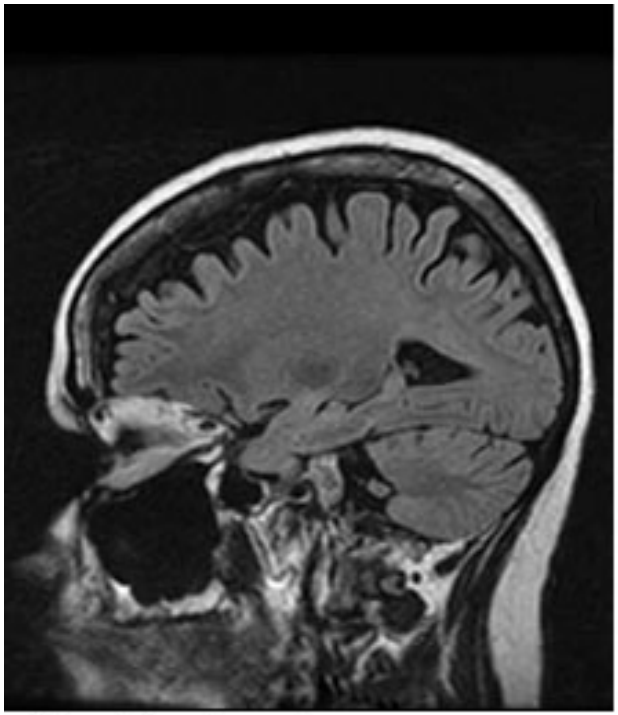
- View of the physical world, whose elements are augmented with the aid of sensors.
- Different from *virtual reality*: the preception of the world is not "replaced" but "enriched".
- The augmentation is in real-time.

Examples: Search and rescue, traffic control

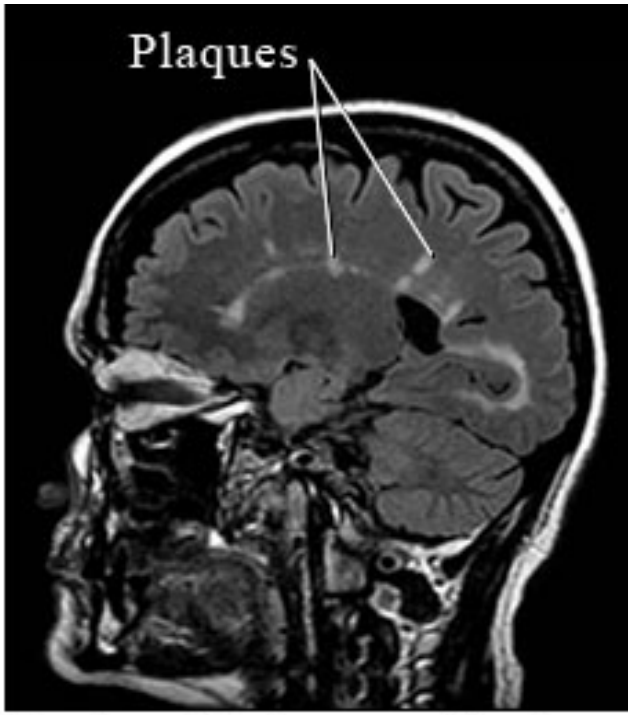


Examples: Jet HUD

Examples: Multiple Sclerosis detection



Healthy brain



Brain with damage (lesions or plaques) caused by MS

Augmented reality and computer vision

- Acquiring, processing, and analysing images.
- A great part of augmented reality involves image processing.

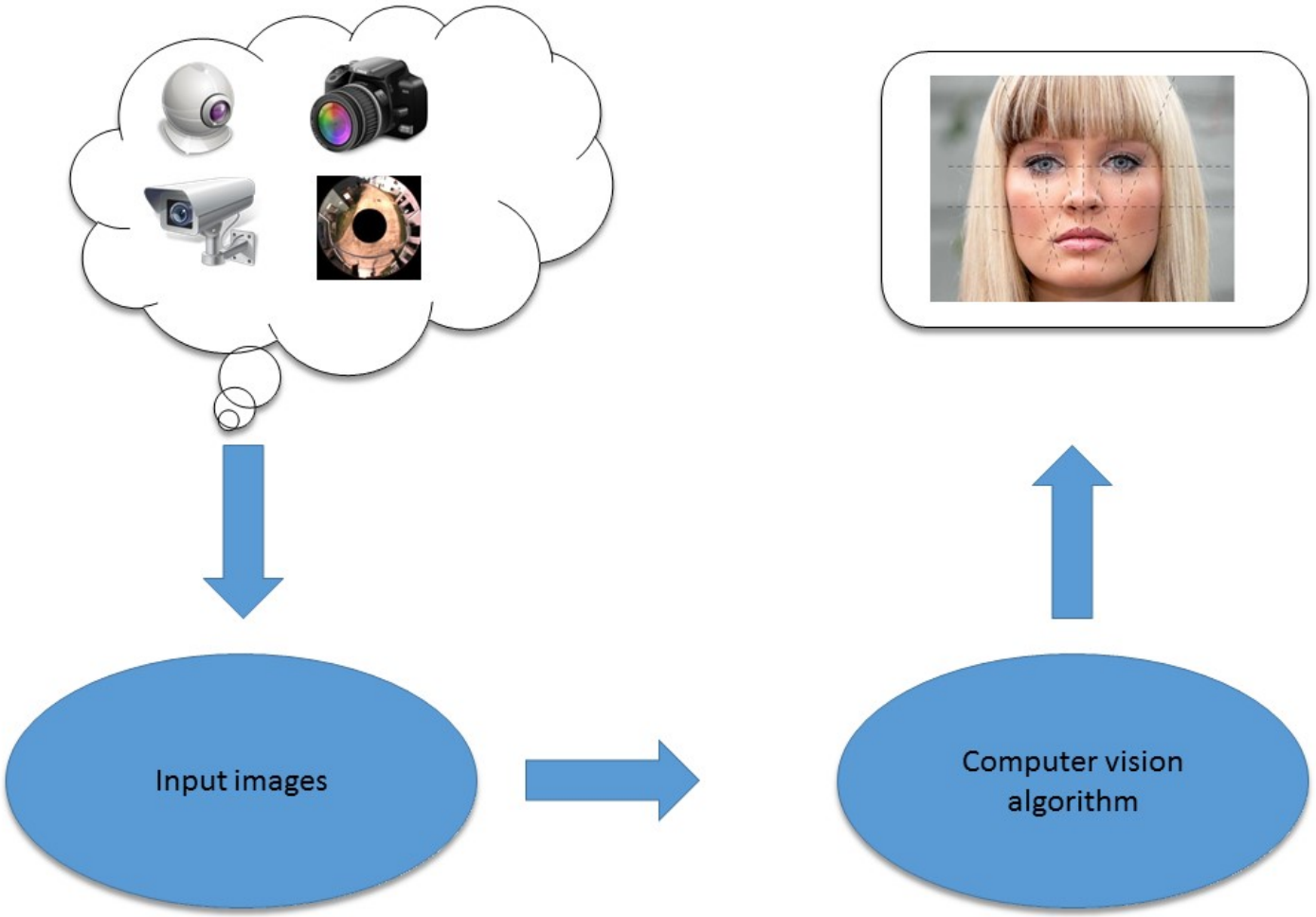
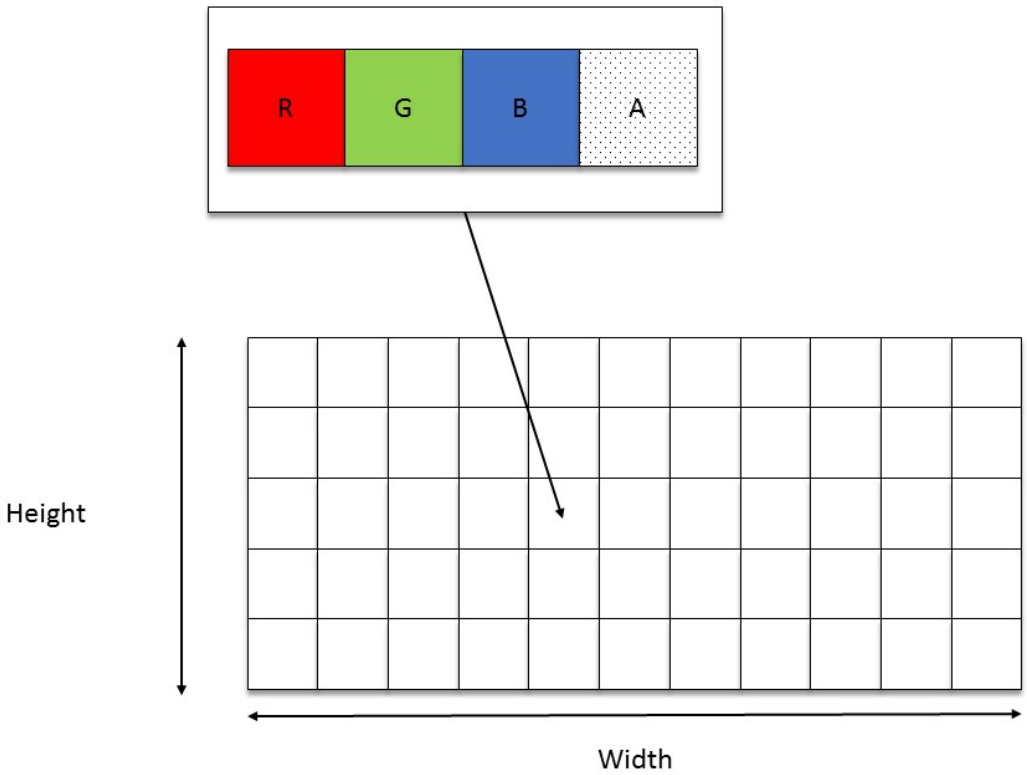


Image colour encoding
RGB encoding

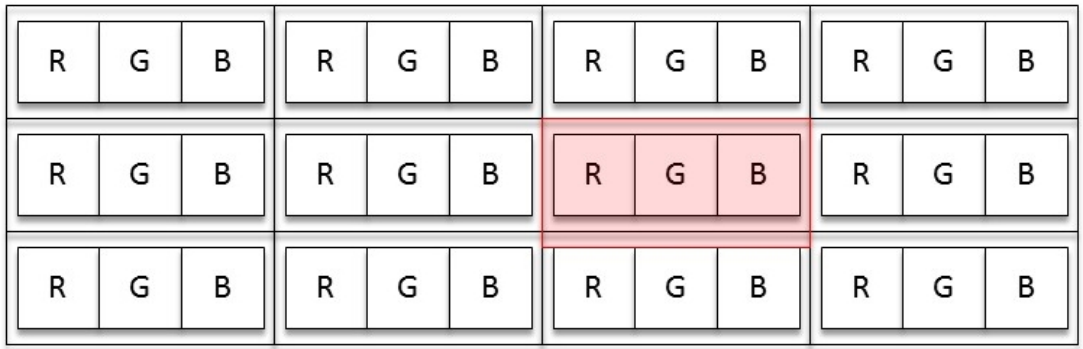
- Most of modern displays and image file formats use the RGB (Red-Green-Blue) encoding.
- 3 bytes for chromatic components + 1 byte (optional) for the alpha channel (transparency).
- An image is a matrix of RGB colours.



Low level: Array of bytes (OpenCV). Components of pixel (i, j)

- R: `j * 3 + pixelPerRow * i * 3`
- G: `j * 3 + pixelPerRow * i * 3 + 1`
- B : `j * 3 + pixelPerRow * i * 3 + 2`

Index in the array of pixel (1,2) G component = 1 * 4 * 3 + 2 * 3 + 1 = 19



High level: `Class` as in `java.awt.Color` or in C# in `System.Drawing`. Matrix of `Color`.
Java:

```
public class Color
{
  private byte r, g, b, a;

  public byte getR() { return r; }
  public byte setR(byte r) { this.r = r; }

  public byte getG() { return g; }
  public byte setG(byte g) { this.g  = g; }

  public byte getB() { return b;}
  public byte setB(byte b) { this.b = b; }

  public Color (int a, int r, int g, int b) { this.a = a; this.r = r
      ; this.g = g; this.b = b;  }
}
```

C#:

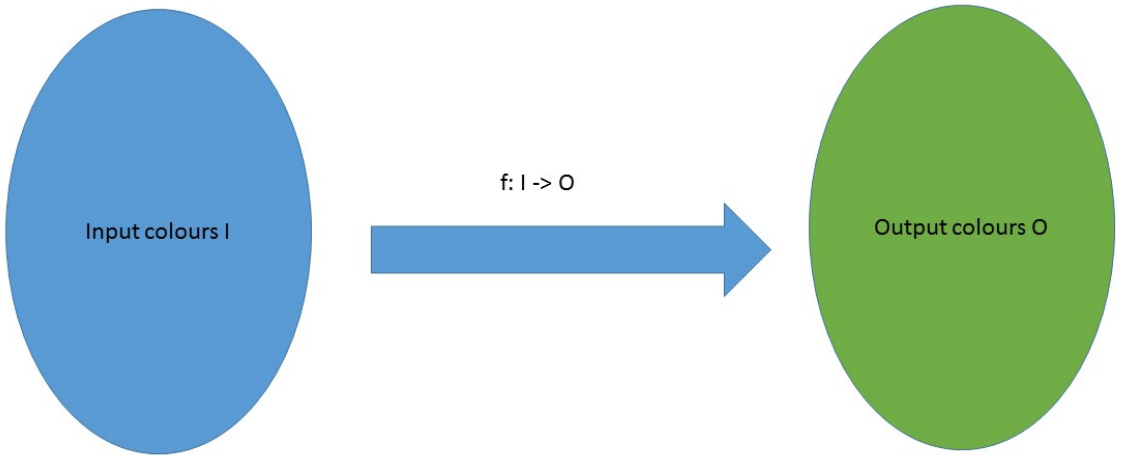
```
public class Color
{
  public byte r,g,b,a;
  public byte R { get {return r;} }
  public byte G { get {return g;} }
  public byte B { get {return b;} }
  public byte A { get {return a;} }

  public Color (int a, int r, int g, int b) { this.a = a; this.r = r
      ; this.g = g; this.b = b; }
}

public static Color Fromargb(int a, int r, int g, int b) { return
    new Color() }
```

Colour space transformations

- Image transformation that only alters the colours and not spatial information (i.e. the position of pixels).
- We map an input set of colours to a modified set of colours.

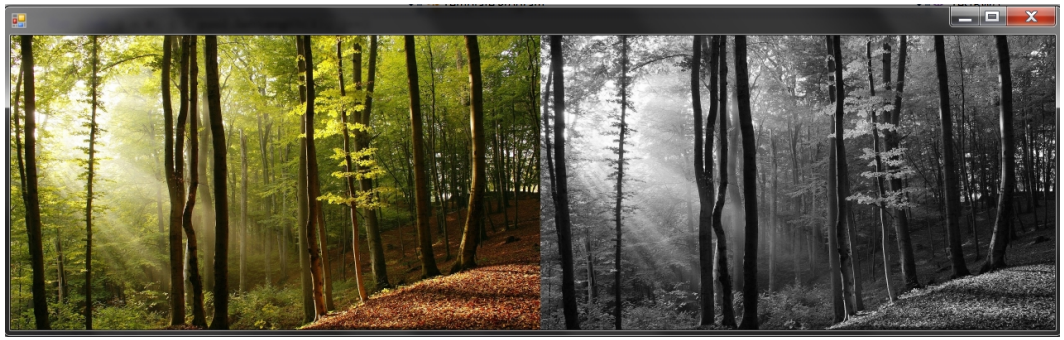


Colour space transformations
Black and white filter in RGB

- BW filter is a colour space transformation, it maps each colour to a black and white scale.
- Ideally BW scale is a scale where 0 is black and 1 white. Gray tones in the middle.
- In RGB gray colours have all the 3 components almost equal.
- Components are in $[0, 255]$: we set all the components to their average.

BW Filter Algorithm:

```
foreach color in image do
  r = color.r
  g = color.g
  b = color.b
  avg = (r + g + b) / 3
  newColor = (avg, avg, avg)
  color = newColor
```



- Invented to keep compatibility with black and white TV's.
- BW displays only use a *luminance* (light intensity) component to define a pixel.
- Encode the signal using two additional chromatic components: U and V.
- The luminance component is a weighted sum of RGB components.
- The chromatic components are obtained by difference from the Y component by subtracting R and B and scaled.

We assume that the RGB components have been normalized to be in the interval [0,1] and not [0,255]

$$W_R = 0.299,\ W_B = 0.114,\ W_G = 1 - W_R - W_B$$
$$U_{max} = 0.436,\ V_{max} = 0.615$$
$$Y = W_R R + W_G G + W_B B$$
$$U = U_{max} \frac{B - Y}{1 - W_B}$$
$$V = V_{max} \frac{R - Y}{1 - W_R}$$

$$Y \in [0, 1], U \in [-U_{max}, U_{max}], V \in [-V_{max}, V_{max}]$$

By inverting those formulas we obtain the inverse transformation

$$R = Y + V\frac{1 - W_R}{V_{max}}$$
$$B = Y + U\frac{1 - W_B}{U_{max}}$$
$$G = \frac{Y - W_R R - W_B B}{W_G}$$

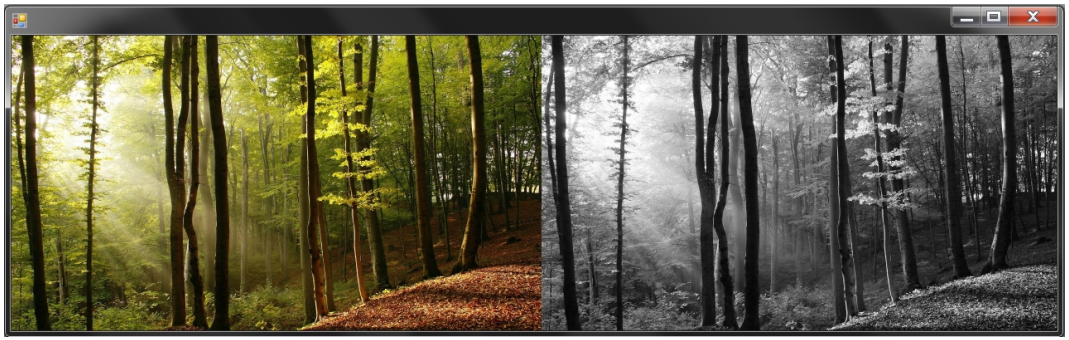
Remember that with this conversion you get RGB in [0,1] and not [0,255]. To get RGB in [0,255] simply multiply every component by 255.

Colour space transformations
BW filter in YUV space

- Transformations which heavily act on the lighting of the image are easier in YUV space.
- YUV is the composition of a luminance component and two chromatic (colour) components.
- By forcing the chromatic components to 0 we are left only with Y which is in the range [0,1].
- The grayscale is simply the luminance component.

BW Filter Algorithm:

```
foreach color in image do
  newColor = (color.Y, 0, 0)
  color = newColor
```



Visual Studio

- Visual Studio 2017 Community: https://www.visualstudio.com/vs/community/.
- Difference between Java and C#: https://msdn.microsoft.com/en-us/library/ms228602%28v=vs.90%29.aspx

C# template

- For each assignment you will be given a VS2015 solution template. You will be prompted to upgrade it to the 2017 format.
- Once upgraded to the new format, you can open the solution and you will find a library `ImageUtilities` to draw images in a window. It is automatically linked in the main project in the template.
- `LoadImage` loads an image from file and returns a matrix of `Color`. The other functions display a matrix of `Color` as an image in a window (read the description in the source).
- You will deliver the template solution containing the implementation of the assignments (if you choose C# as programming language).

Assignment 1:
Colour spaces and B&W filter

- Implement RGB to YUV conversion.
- Implement YUV to RGB conversion.
- Convert a RGB image to YUV and then the converted image in YUV to RGB and display the initial image and the result (they should look like the same). Display both images in a window.
- Implement the Black and White filter both in RGB and YUV. Display the original image and the black and white image in a window.