# PRACTICAL WORK 2
# OPENMP AND CUDA BASED IMAGE PROCESSING

## Introduction

The objective of this practical work is to test different types of colored anaglyphs and image processing methods on stereo images.

## Exercise 1 – Image processing methods

Based on OpenCV, implement with OpenMP and CUDA the following image processing methods on stereo images:

### 1 - Anaglyph color methods

Implement all the anaglyph methods presented here:
https://3dtv.at/Knowhow/AnaglyphComparison_en.aspx

### 2 – Gaussian filtering

Gaussian filtering is a common image processing technique used to blur or smooth an image. It works by convolving the image with a Gaussian kernel, which is a 2D bell-shaped function (defined by kernel size, sigma).

Implement a Gaussian filter capable of processing a stereo image **even on its edges**.

- **Parameters**: kernel size, sigma

### 3 – Denoising

The determinant of a covariance matrix can be used as a measure of the complexity or richness of the data distribution. It is related to the volume of the n-dimensional ellipsoid that describes the spread of the data points in the color space. In the case of RGB colors, the ellipsoid represents the distribution of color values in the image.

If the determinant of the covariance matrix is small, it indicates that the color values are concentrated around a single point or axis, which can indicate a lack of color variation in a neighborhood. If the determinant is large, it indicates that the color values are spread out in multiple directions, which can indicate a wide range of colors.

This exercise aims to create a Gaussian filtering technique that utilizes the determinant of the covariance matrix of a specified neighborhood where the size of the Gaussian filtering kernel applied to a pixel of the source image will be inversely proportional to the value of this determinant.

- **Parameters**: neighborhood size for the covariance matrix, factor ratio applied to determine the gaussian kernel size.

**Remarks:**

- For all these image processing methods you will create a unique .cpp for OpenMP and a .cpp + .cu for CUDA.
- Use command line argument to pass the different parameters.
- Benchmark your implementations.
- For the CUDA implementation find the optimal parameters (grid and block size) for your system.

## Exercise 2 – Shared memory optimization

To apply the Gaussian filtering method with CUDA, modify your implementations to incorporate the concept of shared memory within the blocks.

## Exercises 3 – Execution time comparisons

Compile all the execution time results for your implementations in an Excel file, including the corresponding graphs.