# Report for CV Project “Gray-Code Kernels for Optical Flow Estimation”

In this activity the main goal was to extract gray-code kernels from video frames and perform a matching procedure between adjacent projections in order to estimate the optical flow.

## Gray-Code Kernels

Gray-Code Kernels are a family of filters that can be used for an efficient filtering scheme. Successive convolution of an image requires only two operations per pixel, regardless of size or dimension of the filter. If we have two kernels b1 and b2 and we want to filter an image with these two kernels, first, we calculate with a convolution the result of application of b1 to the image, then the filtering with b2 could be obtained using only 2 summations per pixel, getting partial result from the filtering with the kernel b1, so cutting extensively the spatial and temporal costs. With this type of kernel the operations per pixel will have constant complexity.

These kernels are created with a binary tree and their efficiency depends on the order with which they are applied to an image. The root of the binary tree is the “seed” which in this implementation is the number +1. Another parameter to take into consideration for the building of the binary tree is the height “k” of the tree; also, k determines the length of the filter, so the filter will be 2k long. In this case has been used a power of 2 so 4 bytes kernel long.

## 3D implementation

Here we have used a 3D implementation of the GCK which is useful to track spatio-temporal variations. In particular, the filters are divided in 3 types:

* Spatial filters: no changes along the z axis
* Temporal filters: changes only on the z axis
* Spatio-temporal filters: changes along all three axes

## Workflow

The main part was to extract the projections from the video, then an optical flow procedure is applied to a series of projections of the same type; in particular I’ve chosen the middle one of the entire 3D block along the Z axis, since it is the most representative one of the variation of the movement.

Two matching procedure have been applied: a Lucas-Kanade optical flow also called “sparse” and a dense optical flow with the Farneback method.

Lucas-Kanade method computes optical flow for a sparse feature set (corners detected using Shi-Tomasi algorithm) while the Farneback method computes the optical flow for all the points in the frame. It is based on Gunner Farneback’s algorithm which is explained in “Two-Frame Motion Estimation Based on Polynomial Expansion” by Gunner Farneback in 2003.

The procedure have been tested on some videos belonging to the Weizmann dataset and on the KITTI dataset.

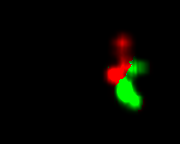
Below are the matching procedures applied to the original video “lena\_walk1” belonging to the Weizmann dataset.

A picture containing text

Description automatically generated

Below are the matching procedures applied to the projections of the video.

A picture containing flying

Description automatically generated

Noticing the points tracked by the Shi-Tomasi algorithm in the first frame belonging to the LK algorithm and in the second picture the motion variations in the Farneback algorithm, green more variation and red less variation.