Motion analysis

Authors: Ierardi Ambra, Scorrano Andrea, Trovatello Alessandro, *Università degli Studi di Genova*

There will be only *one version* of the code and the report submitted, since the group worked together over every part of the assignment.

KEYWORDS: Optical Flow, Change Detection

1 INTRODUCTION

The goal of this assignment was to perform the **motion analysis** in a sequence of frames, representing a video. The first step was to implement the *Lucas-Kanade algorithm*: it is a well-known optical flow algorithm, which solves a system of linear equations to estimate the flow parameters of the sequence of images. The second part was focused on **change detection**, which performs the detection of changes occurring in a scene, elaborating video sequences captured from single or multiple view-points by fixed or moving imaging devices.

2 REPORT TASKS

2.1 Lucas-Kanade algorithm

The Lucas-Kanade algorithm is a widely exploited technique used to estimate **optical flow** in computer vision, capturing the apparent movement of objects in an image resulting from the relative motion between the camera and the scene: this is why it is only useful in cases where the camera is not moving with the exact same motion of the subject of interest. The steps to implement this algorithm are:

- **Spatial gradient calculation**, where the spatial gradients of the image, or partial derivatives, are computed, namely I_x and I_y, concerning the horizontal and vertical directions, respectively.
- **Temporal gradient calculation**, where the computation of the temporal gradient I_t is performed by taking the difference between corresponding pixels in two consecutive frames.
- **Window selection** where a local neighborhood, a window, is chosen around the target pixel for the optical flow estimation. The window's size is a critical parameter influencing the algorithm's performance, since it determines the spatial extent over which the algorithm assumes the flow to be approximately constant, and this affects the reliability of the output.
- **Optical flow calculation**, solving the following system of linear equations:

$$\mathbf{u} = \mathbf{A}^{\dagger} \cdot \mathbf{b} \tag{1}$$

where $\mathbf{A}^{\dagger}$ is the pseudo-inverse of the matrix A:

$$\mathbf{A}^{\dagger} = (A^T \cdot A)^{-1} \cdot A^T \tag{2}$$

which is the matrix containing the spatial gradients for each element of the neighbour, **b** is the vector containing the temporal derivatives of the same points with a minus sign, and **u** is the resulting **vector field**, with on top, the horizontal velocity components, and on bottom the vertical ones. This operation can be done only assuming the image brightness to be constant.
This procedure has to be done until all the image has been covered, moving the window.

As last point of this section, with the function **TwoFramesLK** a map of the vector field was created, between each pair of adjacent frames. Then, a **magnitude map** was created, considering the horizontal and vertical parts of the **u** vector, in the following way:

$$map = \sqrt{u^2 + v^2} \tag{3}$$

where u is the horizontal component and v is the vertical one. This tool provides a mapping of the vector field of the optical flow in the whole image, frame by frame.

2.2 Change detection

This algorithm has been implemented as a difference operation, between the current image I_t and another one, taken as background. The result of this algorithm is a **binary map**, describing which pixels have changed. Two versions of this procedure have been carried out:

- The first one provides the difference between the actual frame I_t and a **reference frame** I_{ref} which is a picture of the empty scene. The resulting binary map has the following criterion of selection:

$$M_t(x,y) = \begin{cases} 1 & \text{if} \quad |I_t(x,y) - I_{ref}(x,y)| > \tau \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

  where τ is a proper threshold;

- The second version is the **running average** algorithm, where the background is updated every time. This is useful in cases where a change in the scene may be due to permanent variations in the setting, or light changes. This is why a single constant background may be not enough to detect accurate changes in a scene. The map is created according to this principle:

$$M_t(x,y) = \begin{cases} 1 & \text{if} \quad |I_t(x,y) - B_t(x,y)| > \tau \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

  where $B_t(x,y)$ is

$$B_t(x,y) = \begin{cases} B_{t-1}(x,y) & \text{if} \quad D_t(x,y) > \tau' \\ (1-\alpha)B_{t-1}(x,y) + \alpha \cdot I_t(x,y) & \text{otherwise} \end{cases} \tag{6}$$

  where τ' is a further threshold, α is a coefficient which determines how much the previous and the actual frames are important, and $D_t(x,y) = |I_t(x,y) - I_{t-1}(x,y)|$ is used to decide if the pixel is moving or not, by comparing images at the current instant and the former one.

To show the algorithm, it was created a subplot for each time instant, containing on the left the actual frame, on the center the background frame (the previous) and the binary map on the right. The difference between the two methods, magnitude map and binary map, can be seen in the results.

3 RESULTS

3.1 Lucas-Kanade algorithm

The performance of Lucas-Kanade algorithm was tested on various dataset of images, obtaining similar results depending on the dataset considered. To simplify, the result on the **Stennis** dataset will be shown as first. The Stennis dataset shows two people playing table tennis and the result is focused on the first few frames where the focus is on the ball, the racket and the arm as follows:

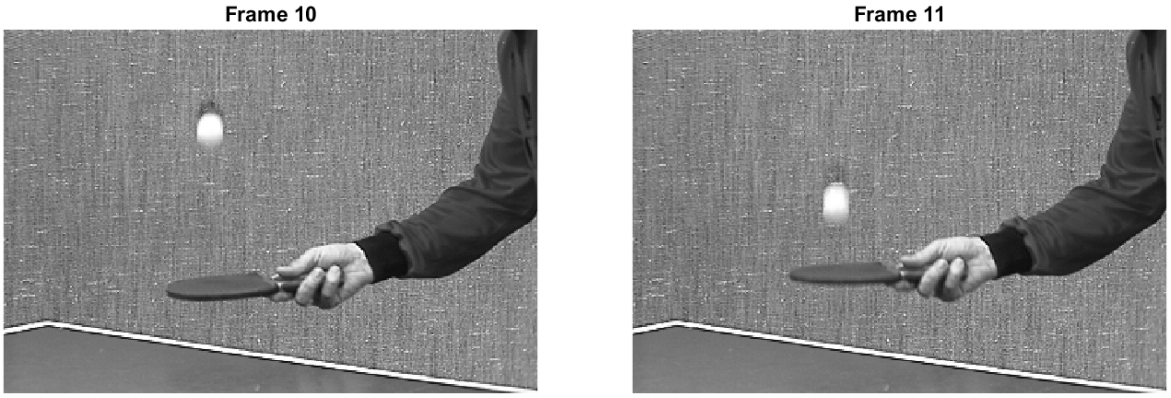

Figure 1: Frame 10 and frame 11 of Stennis Dataset.

Each pair of frames, given as input to the **TwoFramesLK** function, generate, through the built-in Matlab function **quiver**, optical flow vectors which are the gradient of the optical flows. In particular, the gradient, obtained by these two frames, is:

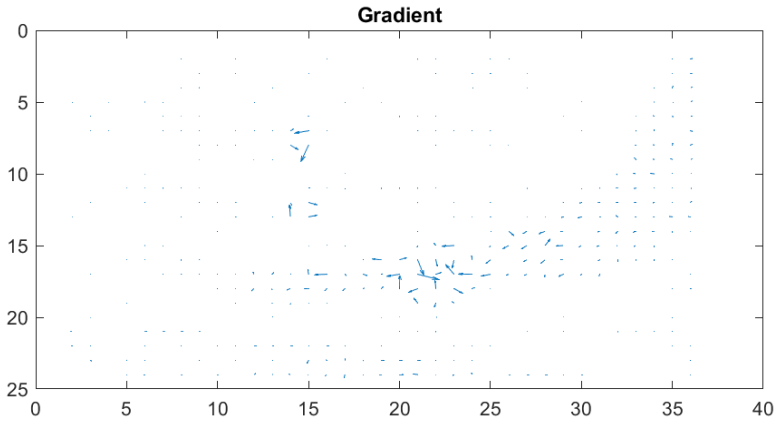

Figure 2: Gradient of the optical flow between frame 10 and frame 11.

As is it shown in Figure 2, the optical flow vectors represents the movement between the images. It is possible to notice, at about 15 on the horizontal axis and 8 on the vertical axis, the vectors describing the descent of the ball. The shape of the arm is also noticeable due to the movement of the arm. The magnitude map generated by frame 10 and frame 11, will be:

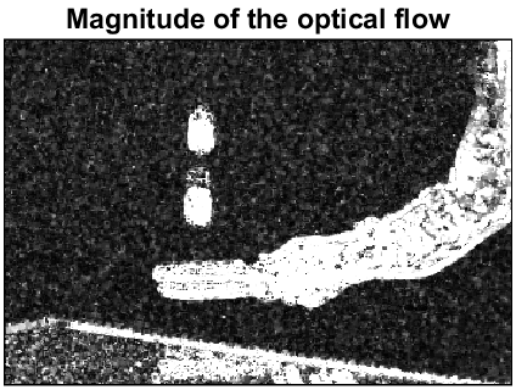

Figure 3: Magnitude map of the optical flow between frame 29 and 30.

The magnitude map represents the intensity of optical flow at each point in the image, so Figure 3 indicates how much the point has moved between frames 10 and frame 11. The white parts are the ones characterized by the major motion.

Concerning the results obtained in the **Video surveillance** dataset, between the frames 29 and 30, which is the following couple of images:

Figure 4: Frame 29 and frame 30 of Video Surveillance Dataset.

the gradient obtained by the sequence of these two images is the following one:

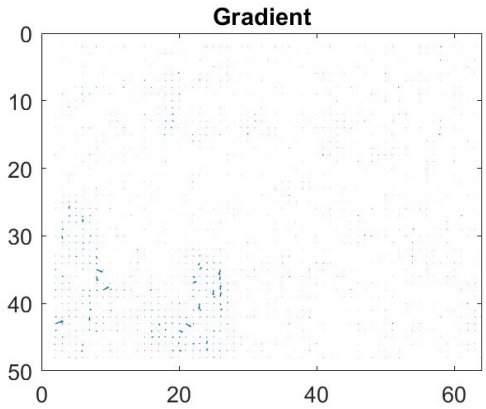

Figure 5: Optical flow gradient of the frames 29 and 30.

It is possible to notice, in the lower left corner, a group of arrows corresponding to the two people moving in the scene. The arrows this time are not so visible due to the fact that the moving part of the image is not so zoomed in.
The magnitude map of these two frames is the following one:

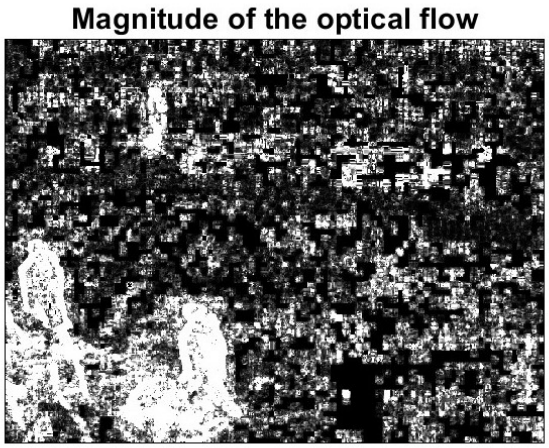

Figure 6: Magnitude map of the frames 29 and 30.

The magnitude map of the velocity in these two frames is quite confused, probably due to low quality of the pictures, since some peaks, the white parts, are shown even where the scene is surely still: see the right part of the scene, which is clearly equal in both the images.

3.2 Change detection

The change detection algorithm exploiting the static background model was implemented in the **change_detection.m** file, where it is possible to notice the binary map as follows:

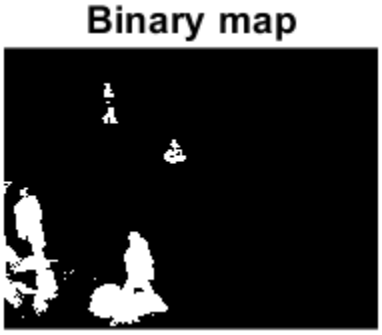

Figure 7: Binary map of video surveillance, static background.

In this case, the people are totally white because they represent a change with respect to the model, which is an empty scene of the train station.
In the algorithm updating the background, the parameters α and τ' were changed many times, in order to see the different effects on the detection of differences in the scene. As first try, it was used α equal to 0.9 and τ' equal to 15: this way the information used to build the background is basically only from the new frame.

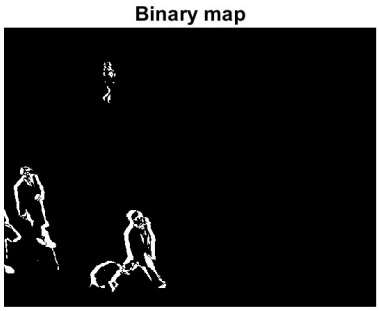

Figure 8: Binary map between frame 29 and 30, first case.

This is why the inner part of the figures is black: no change is detected, since the silhouettes are totally dark, so no change is detected in those parts if moving.
Using α equal to 0.25, the binary map is the following one:

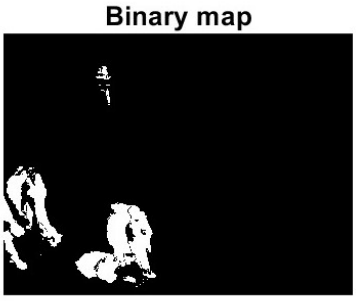

Figure 9: Binary map between frame 29 and 30, second case.

With this value of the α coefficient, the main weight of the image is in the previous frames, so the background with which is made the comparison is much more different with respect to the current image: this is visible in the white parts, which are much wider with respect to the previous case.

The same values were applied to the table tennis sequence. For the first case, the binary map is the following one:

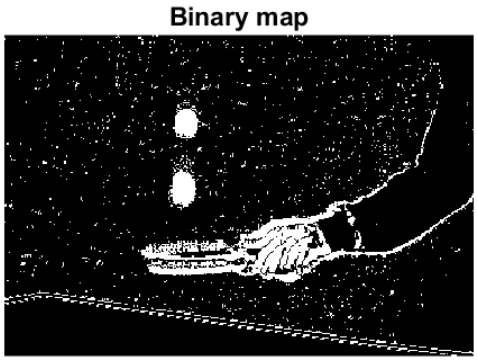

Figure 10: Binary map between frame 10 and frame 11, first case.

The second pair of parameters gave this binary map:

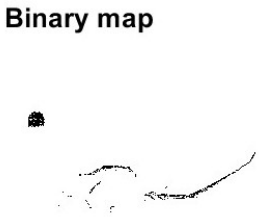

Figure 11: Binary map between frame 10 and frame 11, second case.

In the same pair of frames, two very different binary maps are obtained. This happens because of the α value: in the first case, the background is updated very slowly, since the parameter is close to 0, and being at the initial part of the sequence of frames, the background is not yet fixed, again because of the slowness changing the background. While in the second case the background is updated quickly, since the α value is close to 1. If the background is slowly updated, almost every pixel of the current image will be acknowledged as different with respect to the model, so the resulting map will be full of high values, meaning white in the binary map. While, when the background is quickly updated, more pixels of the current image will be recognized as the same of the previous image, so no change will be detected, and so low values will fill the binary image, meaning that the picture will be mainly dark.
In both datasets, the first set of parameters provide a darker image, with respect to the other case, and this is as expected, due to the previous considerations.

4 SUMMARY

The first step of the assignment was to implement the Lucas-Kanade algorithm, which provides a local estimation of the optical flow and it is suitable for small motion analysis. By plotting the gradient of the optical flow, it was possible to see how the objects in the scene move in that specific frame. The density of arrows is more concentrated in the area of the ”action” of the scene, where the object moves the most.
The second step has the goal of implementing the change detection algorithm, working first with a static background model and then with a background model updated each time. In dynamic environments where the background undergoes significant changes, such as due to moving objects, lighting variations, or gradual scene changes, the static model may become less effective. This can lead to false positives in change detection. Using the running average algorithm, it was possible to solve this problem, and playing with the parameters of the formula (6), it was possible to understand if the pixel is moving or not, adjusting the parameter α, in order to give more weight to the previous background or current image.