Computer Engineering Master Degree

Computer Vision

Report about Lab6

Raffaele Di Nardo Di Maio 1204879

June 3, 2020

1 Goal of the experience

In this laboratory experience, the problems of object recognition and tracking are analyzed. Starting from an input video and some example objects (one example for each element in the video), the keypoints of the objects are visualized and the main features are matched inside the first video frame, using SIFT detection.
Then, the flow of previously computed inlier points is estimated by using the Lukas Kanade algorithm.

2 Code organization

The code has been organized in four main files (documented through doxygen):

- **Lab6.cpp** and its header file **Lab6.h**.
  They contain the `main()` function in which we manage the input of the program and we create the instance of the class `ObjectRecognition`, used to compute the detection.
- **ObjectRecognition.cpp** and its header file **ObjectRecognition.h**.
  They contain the declaration and the implementation of the classes we created to perform all the algorithms useful to extract the keypoints and the features.

3 Command line parameters

On command line the user needs to insert, in order to execute the program: the complete path of the input video with extension *.mov*, including its name, and the path of the folder containing *png* images of the objects.

4 Experimental results

On our first attempt, we try to use *ORB* but then we decided to use *SIFT* instead because it allowed to check the keypoints in a simpler and more accurate way.
The corners of an object image are projected on the first frame using the perspective matrix H, evaluated looking at the keypoints of object images.
For all the remaining frames, the features are tracked thanks to the *Lukas-Kanade* algorithm. During the program execution, the user must insert:

- *Ratio*
  it will be used by SIFT detection, to establish if a point is an inlier or not. By decreasing the value of the ratio, the number of extracted keypoints is lower. Hence the Lukas-Kanade phase becomes less efficient, changing very much the shape of the boundaries around the objects in the last frames.
- *Modality of execution*
  Because of the expensive computation of flows in terms of execution time, we follow a couple of approaches to speed up the final detected video:

1. **Storing the video on the disk (output.avi file).**
   This method requires a lot of execution time but then, we can access to the video that has the same frame rate of the original one and the same resolution.
2. **Resizing all the input frames.**
   We resize each frame to half of its original size and we compute the flows by using the default window size of 21x21 for Lukas-Kanade. This guarantees the user can watch the video in real-time, at a frame rate a little bit higher than the original one.

The ratio values for SIFT, that generates good results are $6, 7$.
We tried also to increase the speed of the video for the real time approach, without resizing the frames. Hence we reduce number of inliers detected in SIFT, by decreasing the ratio, and the size of Lukas-Kanade window. Then this method was removed from the code, because in the last frames of the video there were very strong changes in the shape of boundaries around the objects. Another reason of this removal was that the player time was a little bit slower than the original one, using an OpenCV window.

Some examples of our detection are in Figure 1, 2 and 3, in which the boundaries and the inlier points of the objects are highlighted by red, green, blue and yellow boundaries. These results are taken using first modality of execution.

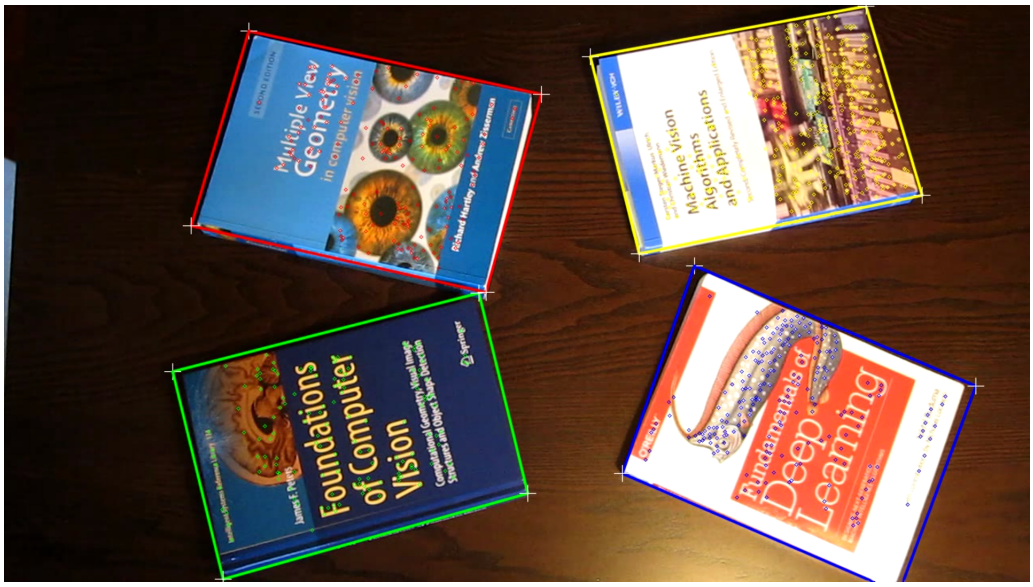

Figure 1: An initial frame($ratio = 6$).

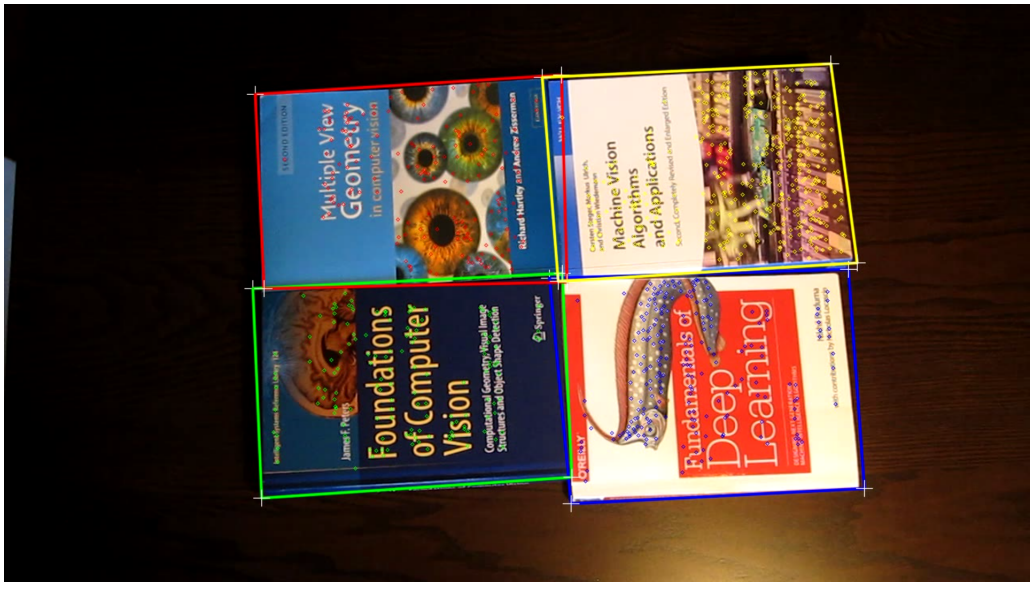

Figure 2: An intermediate frame ($ratio = 6$).

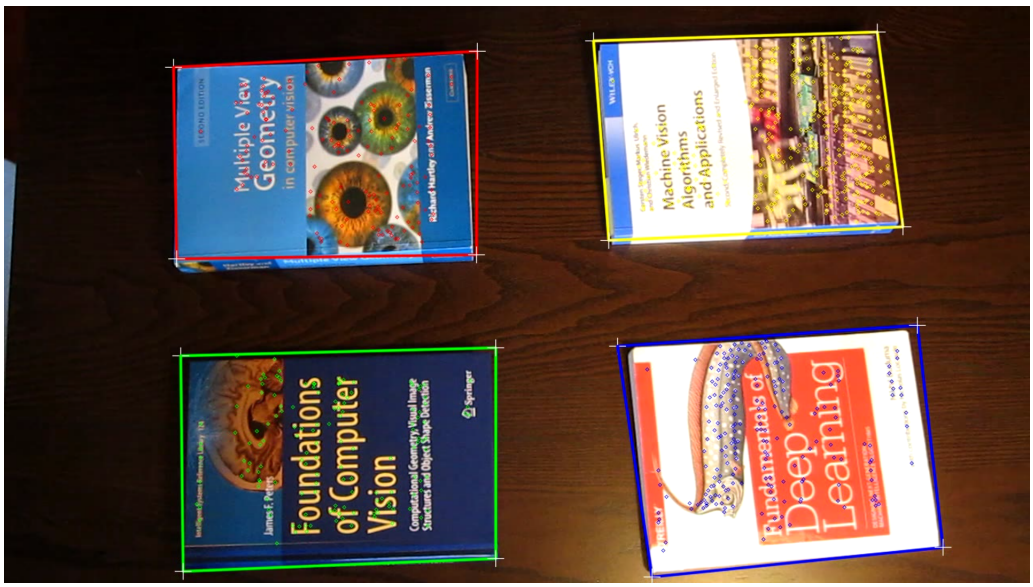

Figure 3: A final frame ($ratio = 6$).