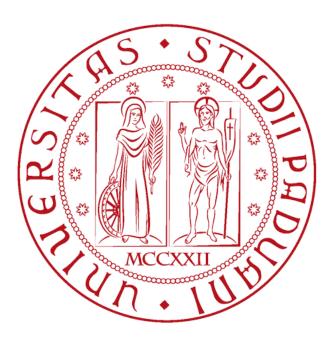
UNIVERSITY OF PADOVA

Computer Vision Intermediate project report



Marek Tatýrek Mateusz Miroslaw Lis Abioye Obaloluwa Peter

Academic Year 2024-2025

Contents

1	Introduction 1.1 Notes	2 2
2	Overview	2
3	Implementation details 3.1 FileLoader 3.2 SceneChangeDetect 3.3 TestDatasetEval 3.4 FilmStatistics	3 3
4	Evaluation	4
5	Dataset description	4
6	Results summary	4
7	Individual contributions	4
8	Conclusion	4

1 Introduction

1.1 Notes

To make code more clear, we use:

- snake_case for variables
- PascalCase for classes
- camelCase for functions

2 Overview

We decided to use class design, where classes are logically separated depending on task. Dataflow between classes is provided by several user structures. Data are processed squentially. Unfortunately used dataflow is not that clear because we havent been able to fully follow originally defined structure. We also used Doxygen documentation so for detailed description you can check it here.

For loading data we have two classes ImageLoader() and VideoLoader(), derived from class InputSource(). From user side the classes have common interface and usage is the same. Important is method hasNextFrame(), used for driving the while loop in the main, returning true in case there is frame or image that we can read. Class is returning timestamp, cv::Mat with current sample.

As was already said, tasks are performed sequentialy, which is convenient, because we are reading many frames and to store them in the buffer, it will be very memory demanding. Also from the same reason we have to take care about not making deep copies.

We have Preprocessing() class, for editing the image before actual detection of haar features.

For detecting features in the images we are using class HaarDetector() using haar cascades for finding desired patterns in the images. In our case faces and eyes.

From the detected features we need to make evaluation and decide which shot type are we having. For this purpouse we have class FeatureEvaluator() that outputs structure classification_result with information if is current sample wide shot, medium or close up.

Because we want to make statictic from the data, we made FilmStatistics() class. It makes sense to use this class only on video data. At the init we provide configuration structure FilmStatisticsEvalConfig, with many settings. We can export time sequences to .csv file or we can use getters to get the time sequences and use them in the code.

For graphical output of the statistic data we have ResultDisplayer() class, which is inputting all data types got from FilmStatistics() getters and returning cv::mat with plots.

3 Implementation details

3.1 FileLoader

In this file we have four classes.

Class InputSource is parent class for ImageLoader and VideoLoader It serves to provide easy interface to work with image and video files. In both cases we aiming to process data in series, that means frame by frame. Both of these classes have same user interface made of:

- hasNextFrame() returns if there is one more frame to process, used for driving while loop.
- nextFrame() returns next frame in the sequence.
- getCurrentTimestamp() returns current timestamp in ms.

For preprocessing we have class Preprocessing(). We only implemented processing methods that we found beneficial. The class contains this methods:

- LoadFrame()
- resizeImage()
- toGrayscale()
- equalizeHistogram()
- GetProcessedImage()

Names on methods are describing function sufficiently.

3.2 SceneChangeDetect

This file contains single function for optimizing processing of Viola & Johnes detector. As an input we take two frames, we compute how much they are different and return boolean value. Because Viola & Johnes is computationally quite expansive, we want to process the frame by it, only when it is really needed.

We have two parameters:

- pixelDiffThreshold Difference between two pixel at same position in input frames.
- threshold Percentage of pixels, that reach pixelDiffThreshold parameter value.

Algorithm computes difference between all pixel at apropriate positions and make boolean array. Then it computes percetage of pixels that reaches the pixelDiffThreshold and compare the value with threshold. If the computed value reaches the treshold it returns true.

3.3 TestDatasetEval

This simple class is designed for computing accuracy of the algorithm on the test dataset. That means at image dataset. Since we have test datased sorted in the folders, task is pretty simple. We have class TestDatasetEval. with constructor we pass ShotType, which we want to test with this class. Then we have two methods:

- check() Is used in while to add currently classified image ShotType to evaluation.
- getEvalResults() After all images are processed we call this method and it returns accuracy.

3.4 FilmStatistics

Class is mainly designed for video. As input it takes class probabilities and it outputs .csv file with statictical data. Constructor takes one argument - structure FilmStatisticsEvalConfig, which is used to set up all the parameters for class.

We have 4 public methods:

- addConfigurationStruct() Serves to change configuration setings.
- addFrameResult() Add frame to final evaluation.
- exportToCSV() Exports computed statistical timelines to csv file for future processing.
- printSummary() Prints basic statistical summary.

It is important to mention what specifically class does. It takes input probability. We can set oversapling or skipping to input data to filter random noise etc. Class computes entropy and entropy variance, it detects cuts in the scenes and clasify most probable ShotType. We can set different window sizes for oversampling window, entropy window and entropy variance window. We can also reduce output data flow if we dont want to have informations from each single shot and we reduce size of output csv file. All of these parameters are settable in FilmStatisticsEvalConfig.

Since we are using sliding windows for signal processing, we introduce delay to the system, which grows as we setting bigger window sizes. Delay is different for each statistical parameter timeseries. Class handle this situation and starts outputing data when all of the parameters are valid, with correct timestamp.

- 4 Evaluation
- 5 Dataset description
- 6 Results summary
- 7 Individual contributions
- 8 Conclusion

References