Homework. 5: SIFT Serialization Library

Ignacio Vizzo, E-Mail ivizzo@uni-bonn.de

Handout : 21.05.2020
Handin: 05.06.2020 at 23:59:59 (CET)

NOTE: PLEASE DO NOT ADD BINARY FILES OR IMAGES TO YOUR REPOSITORY.
The only images you are allowed to submit is the ones that comes with this homework. If any **binary** file or image file(**.png**) is found on your submission, then your homework will be completely discarded.

General rules

1. You need to provide the build system for this homework. This means, you need to provide as many **CMakeLists.txt** files as you think is needed.
2. Your code will be checked using `clang-format`. If your code in not properly formatted, then the tests will not run.
3. Your code will be static-analyzed by `clang-tidy`, if your code does not comply with the analysis, the tests will not run. If you are using VSCode with the reccomended setup, most of the errors will be visible while you program.
4. **ALL** tasks should be solved within the `homework_5` folder, no need to create `task_x` folders.
5. The design of how to solve this exercise is up to you. The **ONLY** requirement to pass the tests are:
   - You should provide one header file(put it where you preffer) and call it `homework_5.h`.
   - All the functions of this exercise must be accessible form this header file
   - You should guarantee this header file can be included thorugh your build system.

A Before you start

This results of this homework will be used for your final project. The better the code you produce, the better your project will work. Some steps on this homeworks will be marked as **optional**, since they won't be tested by they are highly reccomended to solve to gain time before you start working on your project.

A.1 Bag of Visual Words Introduction

Before you even start coding, make sure to watch the Introduction on Bag of Visual words video: `https://www.youtube.com/watch?v=a4cFONdc6nc`

A.2 SIFTS Keypoints and Features Descriptors

If you are not familiarized with the concept of SIFT keypoints and feature descriptors, then it's probably a good idea to study about the topic a bit more before you start with the homework.

A good starting point could be this video `https://youtu.be/CMolhcwtGAU`

A.3 OpenCV4

For this homework you NEED to use the OpenCV library, there will be a small tutorial on how to install the library and some basic concepts. Please make sure to checkout this tutorial at: `https://www.ipb.uni-bonn.de/teaching/cpp-2020/tutorials/`

A.4 OpenCV4 Examples

In companion to the OpenCV tutorial, there is a small repository with some examples. You could use this repository to test your OpenCV installation and also to play around with the code to get familiarized with the library. There are C++as well as Python3 examples. The examples can be found here: `https://gitlab.igg.uni-bonn.de/teaching/example_opencv`

B SIFT Serialization Library (10 points)

In this homework you will create a small utility **serialization** library. The key idea behind this library is that computing **SIFT** descriptors is computationally time consuming. If we analyze the fact that the **SIFT** descriptors will not change over time(unless the input image is changed) it is a good idea to compute the descriptors on an "offline" fashion. This way, if you are working with a given dataset, you can compute first all the descriptors for this dataset and then work with this data, not needing to compute **SIFT** every time you run a simple test. An example of this idea is shown in Figure 1:

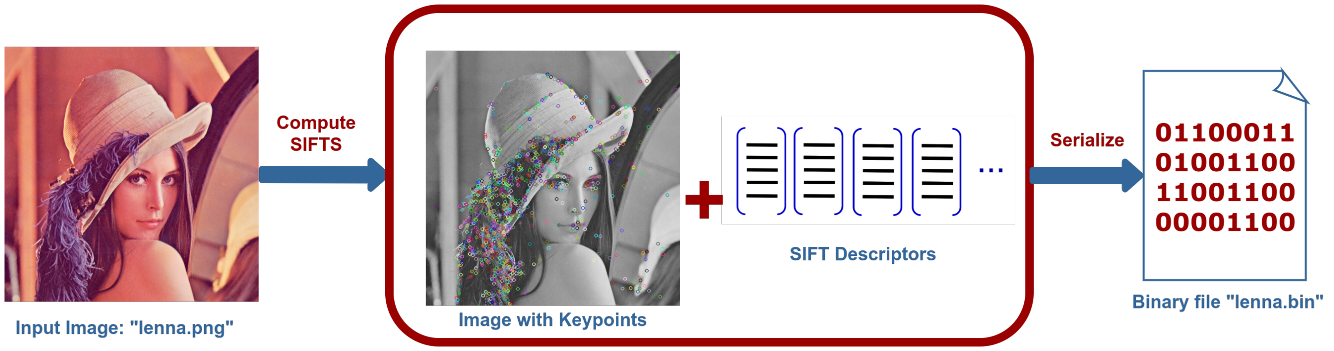

Figure 1: Serialization pipeline example

B.1 task_1: cv::Mat serialization library

The way OpenCV stores the results of the **SIFT** descriptors of a given image is using the data type `cv::Mat`, this is basically, a Matrix. Serializing a SIFT descriptor basically means, providing a serialization for the **cv::Mat** type.

One advantage of this fact is that you could write a simple test program to test tour serialization library, read an image(it's also stored as cv::Mat), write its binary counterpart to the disk, read this binary file(de-serialize) and show the image. If the image looks the same, with no strange artifcats, then you might assume the serialization of the cv::Mat is done properly.

To solve this task you need to write a small utility library that writes the content of a cv::Mat(respresenting SIFT descriptors) to a binary file. To achieve this you need to implement two functions defined in the **serialize.hpp** header file.

```
void Serialize(const cv::Mat& m, const std::string& filename);

cv::Mat Deserialize(const std::string& filename);
```

B.1.1 Requirements to pass the tests

- The binary file should have the **same** name(stem) as the input image file but with a ".bin" extension.
- Create a library called **mat_serialization**, the `hw_bot` will look for this library.
- The serialization library should work independently if the given input is an image or a SIFT descriptor.
- The functions should be accessible by only including the **homework_5.h** header file.
- Make sure the library is linked against all the required dependencies. For example, if the library depends on OpenCV, then, you need to make sure that it's being linked with OpenCV.

B.1.2 *Tips*

1. Spend some time trying to understand what is the purpose of this homework before trying to solve the task.
2. The solution for this homework could be no more than 40 lines of code. You should probably spend more time thinking on how to solve it than coding the solution.
3. Make sure you pay attention to the **namespaces**.
4. Make sure you create a small test program to test serialization-deserialization before submitting your homework.

B.2 task_2: Convert an image dataset into SIFTS descriptors binary files

In this task, you will use the **ipb::serialization** library from previous exercise and will go one step further.

You need to write a small utility library that reads **ALL** the **.png** images from a given dataset, compute the **SIFT** descriptors for each image, and store the results to the filesystem. You can see this library as a "conversion" library, where you exchange simple **RGB** images for its **SIFT** descriptors and store the results in binary format.

You are given with a small test dataset(**homework_5.zip**) with 10 images that could be used to test this task. Those 10 images are the only images allowed to be stored in your homework repository.
To achieve this you need to implement two functions defined in the **convert_dataset.hpp** header file:

```
void ConvertDataset(const std::filesystem::path& img_path);

std::vector<cv::Mat> LoadDataset(const std::filesystem::path& bin_path);
```

B.2.1 Requirements to pass the tests

- You should only process **.png** while serializing and **.bin** while de-serializing.
- Remember you need to compute **SIFT** descriptors, and these is what you need to serialize to the disk.
- The name of each new binary file should match its image counterpart but using the ".bin" extension.
- The output binary files should be placed on the **same** parent folder than the orignal images, but in a separate **bin** directory. For example, if the images are in **data/freiburg/images/** then the binary files must be written to **data/freiburg/bin/**
- Create a library called **convert_dataset**, the `hw_bot` will look for this library.
- The functions should be accessible by only including the **homework_5.h** header file.
- Make sure the library is linked against all the required dependencies. For example, if the library depends on OpenCV, then, you need to make sure that it's being linked with OpenCV.

C Optional

This task will not be corrected/reviewed, since it's not part of your submission but it's highly reccomended to do it.

Once you know your utility libraries are working(basically, that the hw bot is happy with your submission) You can take advantage of this library and gain some time for your project.

Download the freiburg dataset, this is the dataset that you must use for your final project, a copy can be found: `https://uni-bonn.sciebo.de/s/c2d0a1ebbe575fdba2a35a8033f1e2ab`

Once you have all the images available, you could use the functions you've implemented and pre-compute all the SIFT features for ALL the images in this dataset, and store the results to disk. This will take some time, since the number of images is relatively high. Once you completed this step, you will have a SIFT-binary version of the images on your local system, ready to use in your final project.

NOTE: PLEASE DO NOT ADD THESE IMAGES/BINARIES TO YOUR CPP-HOMEWORKS REPOSITORY.

You can place these anywhere on your system, I usually have a **dev/datasets/** directory on my home directory.