

BGSLibrary: An OpenCV C++ Background Subtraction Library

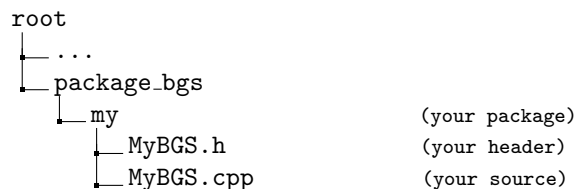
Andrews Sobral

April 24, 2014

1 How to contribute

Everyone is invited to cooperate with the BGSLibrary project by sending any implementation of background subtraction (BS) algorithms.

In this section, some instructions and tips are presented to make easier your contribution. Firstly, you needs to create your own BS package as in the example below:



The next step is to inherit the IBGS interface (Listing 1) in your class header as in Listing 2. The IBGS interface is located in the **package_bgs** folder. All BS algorithms in the BGSLibrary must inherit this interface. This make simple to choose the desired BS algorithm separating an object interface from its implementation.

Listing 1: IBGS.h

```
1 #pragma once
2 #include <cv.h>
3
4 class IBGS
5 {
6 public:
7     virtual void process(const cv::Mat &img_input, cv::Mat &
8         img_foreground, cv::Mat &img_background) = 0;
9     virtual ~IBGS() {}
10 private:
11     virtual void saveConfig() = 0;
12     virtual void loadConfig() = 0;
13 };
```

Listing 2: MyBGS.h

```

1  #pragma once
2
3  #include <cv.h>
4  #include <highgui.h>
5
6  #include "../IBGS.h"
7
8  class MyBGS : public IBGS
9  {
10 private:
11     cv::Mat img_previous;
12
13 public:
14     MyBGS();
15     ~MyBGS();
16
17     void process(const cv::Mat &img_input, cv::Mat &img_output, cv::
        Mat &img_bgmodel);
18
19 private:
20     void saveConfig() {}
21     void loadConfig() {}
22 };

```

After that, you must implement the `process(...)` function as in Listing 3. The Listing 3 shows an example of the Frame Difference algorithm. Remember to copy the foreground mask to **img_output** and the background model to **img_bgmodel** at the end of the process function. Leave empty the **img_bgmodel** if your BS algorithm does not creates a background model.

Here are some basic tips and best practices you can use to help ensure the success of your code:

- Avoid using restricted or proprietary libraries. Give preference to free and open source libraries.
- Also, if possible, avoid using specific headers and functions of your OS. Write your BS algorithm with cross-platform compatibility.
- Please consider the use of namespaces carefully when you are writing your code. Some classes may have the same name in other package. This may cause problems when compiling the library.
- Check if your code have buffer overflows and memory leak occurrences. A memory leak is unnecessary memory consumption by a computer program. If a program with a memory leak runs long enough, it can eventually run out of usable memory. This issue can crash your BS algorithm for long stream videos. There are several memory leak detection tools available at internet. The leaktracer¹ and deleaker² are simple and efficient memory-leak tracer for C++ programs.

¹<http://www.andreasen.org/LeakTracer/>

²<http://deleaker.com/>

Listing 3: MyBGS.cpp

```

1 #include "MyBGS.h"
2
3 MyBGS::MyBGS() {}
4 MyBGS::~MyBGS() {}
5
6 void MyBGS::process(const cv::Mat &img_input, cv::Mat &img_output,
7                     cv::Mat &img_bgmodel)
8 {
9     if(img_input.empty())
10         return;
11
12     if(img_previous.empty())
13         img_input.copyTo(img_previous);
14
15     cv::Mat img_foreground;
16     cv::absdiff(img_previous, img_input, img_foreground);
17
18     if(img_foreground.channels() == 3)
19         cv::cvtColor(img_foreground, img_foreground, CV_BGR2GRAY);
20
21     cv::threshold(img_foreground, img_foreground, 15, 255, cv::
22                  THRESH_BINARY);
23
24     img_foreground.copyTo(img_output);
25     img_previous.copyTo(img_bgmodel);
26
27     img_input.copyTo(img_previous);
28 }

```