Erasmus Mundus Joint Master Degree (EMJMD) in
Image Processing and Computer Vision (IPCV)

Applied Video Sequence Analysis

Lab 2 “Object detection and classification”

José M. Martínez - Paula Moral

josem.martinez@uam.es - paula.moral@uam.es

OBJECTIVES

- **To develop routines for blob extraction and classification** using the output of the MOG2 background subtraction module provided by OpenCV.
- **To develop routines to detect stationary foreground blobs** in video sequences

- **To write a report** to summarize achievements and results

DELIVERY RULES

- Assignment available on Moodle to submit your material
- The material must be submitted as a ZIP file with the following format *name1Surname1_ name2Surname2_lab2.zip*
- The submitted ZIP file will contain
  - Report in PDF format (max 6 pages)
  - *Makefile* to compile and link the program by simply running *make** *(Suggestion: use the makefile provided for this lab)*
  - "src" directory with all source files (.h, .hpp, .c and .cpp) necessary for compiling and executing the corresponding program in Linux
  - "other" any other code in MATLAB/Python you have used/developed for this assignment

*You may want to develop your C/C++ program using Eclipse but only the Makefile must be submitted (please do not submit Eclipse' config files)

TASKS

- Task 0 – Study the sample code
- Task 1 – Blob Extraction
- Task 2 – Blob Classification
  - Optional - Improved classification model
- Task 3 – Stationary Blob extraction and classification

TASKS

- Task 0 – Void object detection and classification using template
  1. Go to “LAB 2” on Moodle
  2. Download the example project “LAB 2 code template.zip”
  3. Download the datasets “AVSA Lab2 datasets”
  4. Uncompress the files
  5. Create a new Project Lab2.0AVSA2021 in Eclipse (see next slide)
  6. Open the Lab2.0AVSA2021.cpp file in the “src” directory
  7. Set the directory of the dataset and results according to your installation
  8. Compile it (“Build Project” in Eclipse)
  9. Create a “Launch Configuration” for the Project
     - Select the Project in the “Project Explorer” tab, go to the toolbar and select in the “Launch Configuration”pane “Create New Launch Configuration”
     - In the “Environment” tab set ***LD_LIBRARY_PATH=/opt/installation/OpenCV-3.4.4/lib***
  10. Run it …

TASKS

- Creating an Eclipse Project from scratch (see Lab0)
  - How to install and setup Eclipse for OpenCV 3.4.4
    - Configure Eclipse CDT for using OpenCV
- Creating an Eclipse Project with existing code
  - After creating the Projects and src folder …
  - … instead of creating a new source file (step 7), copy the files (.cpp, .hpp) into the src folder
    - Select them in a Files explorer and move them to the Project src folder in Eclipe
  - Continue with step 8
- …

TASKS

- …
- Creating a copy of an Eclipse Project (for incremental development – e.g., different versions in Lab1)
  - Select source Project in "Project Explorer": then "copy" and "paste"
  - Name the new project
  - Rename "main" file and change *project dependent* code (e.g., project_name)
  - Delete Binaries in Project Folders (e.g., Debug)
  - Build the Project
  - Create Launch Configuration for the new project (see Lab0)
    - Remember to set LD_LIBRARY_PATH
  - Run the Project

TASK 0 - STUDY SAMPLE CODE

- **Lab2.0AVSA2020.cpp**
  - The *Lab2.0AVSA2020.cpp* file integrates all functionality for the analysis of blobs.
  - Basically, it performs background subtraction using the Mixture of Gaussians approach to provide a foreground binary mask *fgmask* where the values 255 127 and 0 correspond to foreground, shadows and background, respectively.
  - Then, AVSA2020Lab2.0.cpp calls the blob analysis functions and displays their results to perform some requested tasks in this assignment.
  - OpenCV’s background subtraction algorithms can be used as presented in the following lines of code:

```
//Creation of the MOG2 module and compute foreground (in fgmask)

Ptr<BackgroundSubtractor> pMOG2 = cv::createBackgroundSubtractorMOG2();

pMOG2->apply(frame, fgmask, learningrate); // update background model
```

TASK 0 - STUDY SAMPLE CODE

- **blob.hpp**
  - The *blob.hpp* file provides the basic structure to handle blobs *cvBlob*, and the prototypes of the blob analysis routines.
  - To create a new blob, use the in-line function provided in the same file:

```
//create a new blob with ID=1, center at (19,10), width=100 and height=200
cvBlob blob = initBlob(1, 19, 10, 100, 200);
blob.label(PERSON); //set class label of blob (using the CLASS type)
```

  - We also use the standard C++ class std::vector to manage lists of blobs (i.e. add, remove and search blobs in lists). These examples show the required functionality for the lab:

```
std::vector<cvBlob> bloblist; //create an empty blobList
bloblist.clear();//clear blob list
bloblist.push_back(blob); //add the cvBlob ‘blob’ to a list
//get each blob from a list
for(int i = 0; i < bloblist.size(); i++)
   cvBlob blob = bloblist[i]; //get i-th blob
```

For further details of std::vector, refer to http://www.cplusplus.com/reference/vector/vector/

TASK 0 - STUDY SAMPLE CODE

- Prototypes of the blob analysis routines.
  - **Provided routine to paint the bounding boxes** corresponding to the extracted blobs.
    - **Mat paintBlobImage(Mat frame, std::vector<cvBlob> bloblist, bool labelled);**
  - **Functions to be implemented (void functions provided)**
    - **int extractBlobs(Mat fgmask, std::vector<cvBlob> &bloblist, int connectivity);**
    - **int removeSmallBlobs(std:vector<cvBlob> bloblist_in, std::vector<cvBlob> &bloblist_out, int min_width, int min_height);**
    - **int classifyBlobs(std::vector<cvBlob> &bloblist);**
    - **int extractStationatyFG(Mat fgmask, Mat &fgmask_history, Mat &sfgmask);**

TASK 1 – BLOB EXTRACTION

- Implement in C/C++ language a **routine for blob extraction** based on the ***sequential Grass-Fire*** algorithm explained during lectures (description provided in *LAB2_AVSA_material_task1.zip*)
  - **int extractBlobs(Mat fgmask, std::vector<cvBlob> &bloblist, int connectivity);**
- Then, implement another **routine to remove blobs that have less than a certain size**. You should compute the size of each extracted blob and discard it if its size is below a threshold.
  - **int removeSmallBlobs(std:vecto<cvBlob> bloblist_in, std::vector<cvBlob> &bloblist_out, int min_width, int min_height);**
- For **testing**, use **2-3 sequences** selected from previous lab or this one

Hint: For blob extraction using OpenCV, it is recommended to use the available functionality for connected component analysis through the function *floodFill* and the structure *cv::Rect*. Check the online documentation for related functionality at https://docs.opencv.org/3.4.4/d7/d1b/group__imgproc__misc.html

TASK 2 – BLOB CLASSIFICATION

- Implement in C/C++ language a **blob classification routine** based on the ***aspect ratio* feature** and the ***simple statistical classifier*** explained in class *(Gaussian,* with feature models based on mean and variance).
  - **int classifyBlobs(std::vector<cvBlob> &blobllist>);**
- The considered classes are defined in *Blob.hpp* using a *typedef* command named as *CLASS (PERSON, CAR …)*.
- For **testing**, use **2-3 sequences** selected from previous lab or this one

TASK 3 – STATIONARY FG EXTRACTION

- Implement in C/C++ language a routine for extracting stationary foreground pixels.
  - **int extractStationatyFG(Mat fgmask, Mat &fgmask_history, Mat &sfgmask);**

- This routine should be based on the paper "Stationary foreground detection for video-surveillance based on foreground and motion history images" and implement the algorithm described in section 3.1.
- In order to detect stationary foreground, also apply equations (9) and (12) without considering MHI or SHI variables
- The paper is available in Moodle as additional material.
- Test sequences are available in Moodle. Sample results must be reported for the majority of the sequences.

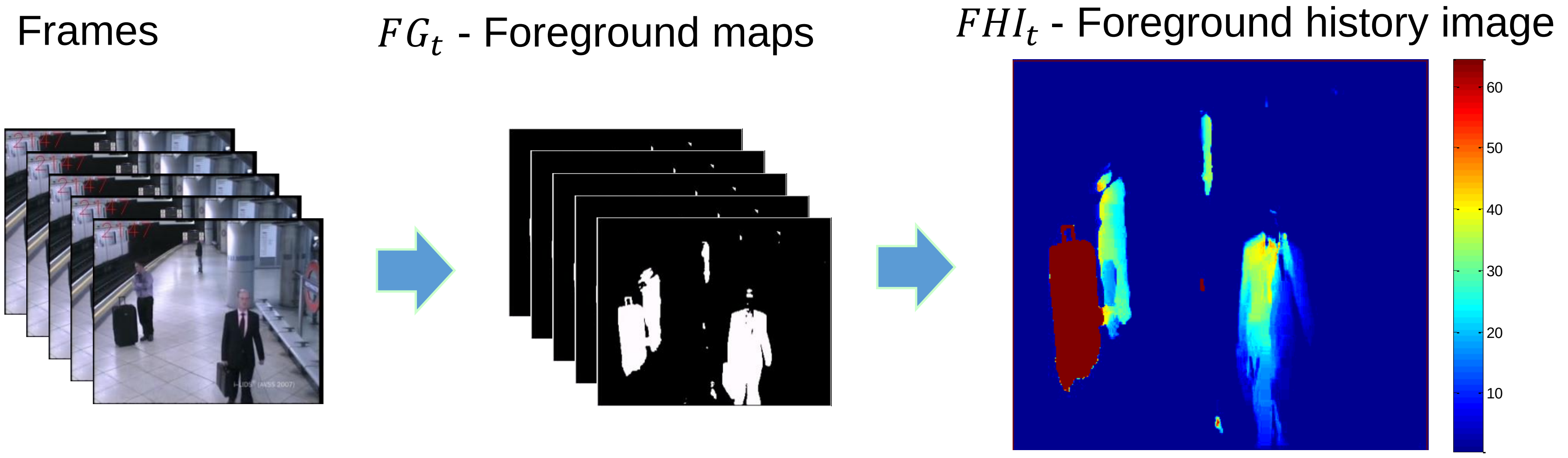
TIP: Use of the parameter *learningRate* to control the update rate of the background model. Please adjust this parameter to avoid including stationary objects in the background model, in order to "give time" to the foreground objects to "create" history

TASK 3 – STATIONARY FG EXTRACTION

- Static detection: Foreground analysis
  - Accumulation over time of *background subtraction* results

$$FHI_t(\mathbf{x}) = FHI_{t-1}(\mathbf{x}) + w^f_{pos} \cdot FG_t(\mathbf{x}),$$

$$FHI_t(\mathbf{x}) = FHI_{t-1}(\mathbf{x}) - w^f_{neg} \cdot (\sim FG_t(\mathbf{x})) \,,$$



EVALUATION

- This lab assignment will be **graded with 10 points**
  - **Mandatory tasks**
    - Report (**up to 2.5 points**) – Please follow the report format (see *Moodle*)
    - Task 1 - Blob Extraction **(up to 2.5 points)**
    - Task 2 - Blob Classification **(up to 2.5 points)**
    - Task 3 - Stationary Blob extraction and classification **(up to 2.5 points)**

Universidad Autónoma de Madrid

CRITERIA FOR EVALUATION

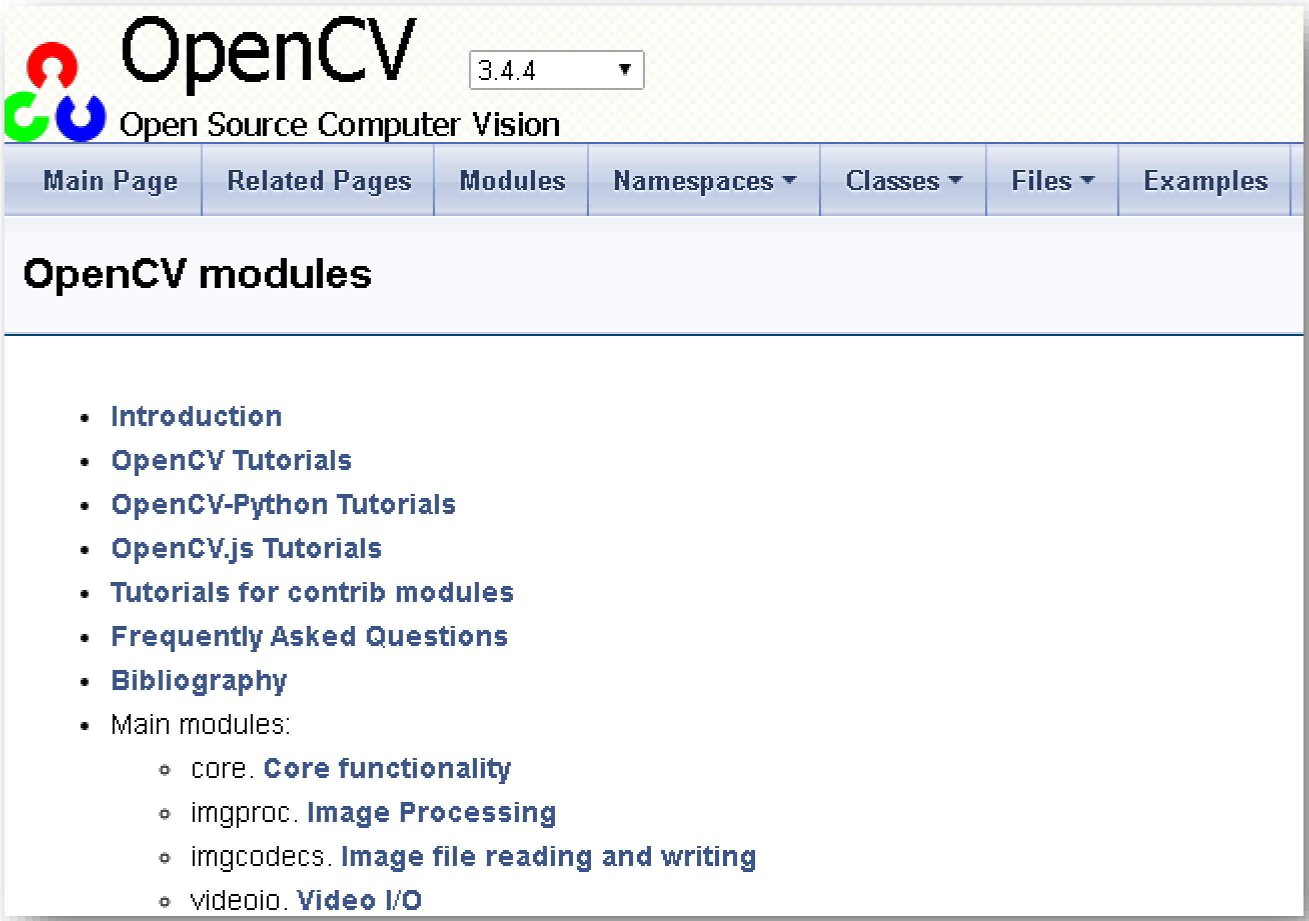
- The following general criteria will be used for grading:
  - The solution addresses the requirements for the assignment.
  - The program compiles, links, and executes.
  - The program runs correctly.
  - The program is easy to read and to understand, i.e., it is well commented, and variables are correctly named.
  - The lab report clearly describes the algorithm and experiments.

- The following specific criteria will be used for grading:
  - Foreground segmentation using colour information.
  - Complexity of the implemented algorithms
  - Processing with OpenCV functions at matrix/vector level instead of pixel-level.

In addition, up to an extra point (1 point) may be awarded depending on the quality of the software delivered in the assignment.

HELP

- Check OpenCV documentation for finding specific functions at https://docs.opencv.org/3.4.4/



OpenCV modules

- Introduction
- OpenCV Tutorials
- OpenCV-Python Tutorials
- OpenCV.js Tutorials
- Tutorials for contrib modules
- Frequently Asked Questions
- Bibliography
- Main modules:
  - core. Core functionality
  - imgproc. Image Processing
  - imgcodecs. Image file reading and writing
  - videoio. Video I/O