Design of Multimedia Applications

Assignment 3: GStreamer ! research

Deadline: 2PM Apr. 26th 2019

1 Introduction

The first assignment was more or less an introduction to manipulating media. The second assignment enables you to make media prototypes which only need more robustness to errors before becoming real products. This assignment will learn you how to "misuse" the GStreamer environment in order to perform research to new algorithms. The practices you learn here should not be used for product development, but only to make your research life easier.

In essence, you know everything about the GStreamer environment at this moment, so there will be no tutorial sections this time in order to get you started. Try to look for code snippets online, but take care that your code is readable by a programmer. If the function you use explains itself, then a comment is not needed, but if sections of code are not straightforward, then add comments to make it easy to understand the code. It should not take more than 15 minutes for a programmer to thoroughly understand what is happening inside the program.

2 Motion Detection

In a security environment, cameras want to detect motion before starting to store information. Different motion detection algorithms exist, all of them with different specifications, i.e. one algorithm will be tuned for the detection of people, the other to detect cars. Some algorithms will be able to run on a battery operated hardware device, while the other needs a server farm at amazon. In this assignment, we are going to do research on motion detection algorithms and we are going to use GStreamer to make our life as easy as possible.

2.1 Background subtraction

Background subtraction is a common and widely used technique for generating a foreground mask (namely, a binary image containing the pixels belonging to moving objects in the scene) by using static cameras.

As the name suggests, background subtraction calculates the foreground mask performing a subtraction between the current frame and a background model, containing the static part of the scene or, more in general, everything that can be considered as background given the characteristics of the observed scene.

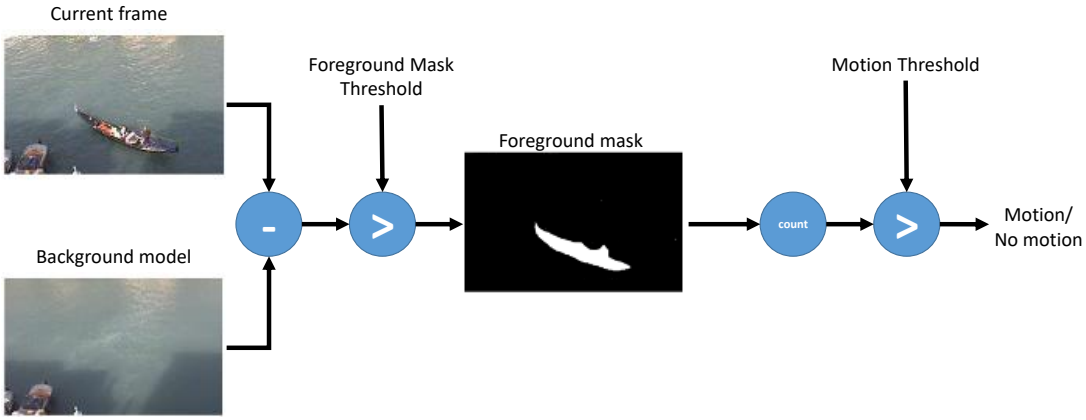


Figure 1: Background subtraction as motion detector including a "ForegroundMaskThreshold" to convert the difference between current frame and background model to foreground mask followed by a "MotionThreshold" to end up with a binary motion/no-motion decision.

Background modeling consists of two main steps, namely background initialization and background update. In the first step, an initial model of the background is computed, while in the second step that model is updated in order to adapt to possible changes in the scene.

In order to go from background subtraction to motion detection, one first has to binarize the foreground mask using a "ForegroundMaskThreshold". Afterwards, on the foreground mask image, one can apply a "MotionThreshold" in order to convert it to 1 binary value for the image. For example, whenever there are more than 1% (a "MotionThreshold" of 1%) of the pixels different from the background, motion can be considered present in the scene.

2.2 Evaluation

The detection performance of an algorithm can be measured using accuracy, precision, and recall. Wikipedia `https://en.wikipedia.org/wiki/Precision_and_recall` summarizes it nicely in Figure 2.

		Predicted condition			
	Total population	Predicted Condition positive	Predicted Condition negative	Prevalence $= \frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	
True condition	condition positive	**True positive**	**False Negative** (Type II error)	True positive rate (TPR), Sensitivity, Recall, probability of detection $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False negative rate (FNR), Miss rate $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$
	condition negative	**False Positive** (Type I error)	**True negative**	False positive rate (FPR), Fall-out, probability of false alarm $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	True negative rate (TNR), Specificity (SPC) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$
	Accuracy (ACC) $= \frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	Positive predictive value (PPV), Precision $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Test outcome positive}}$	False omission rate (FOR) $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Test outcome negative}}$	Positive likelihood ratio (LR+) $= \frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) $= \frac{\text{LR+}}{\text{LR−}}$
		False discovery rate (FDR) $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Test outcome positive}}$	Negative predictive value (NPV) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Test outcome negative}}$	Negative likelihood ratio (LR−) $= \frac{\text{FNR}}{\text{TNR}}$	

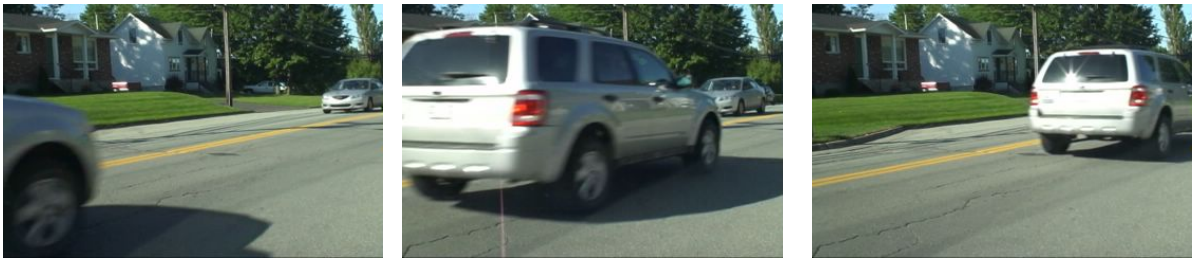
Figure 2: Performance measures.

2.3 Data set

The good thing about researching motion detection is that there are a lot of datasets that can be found online. These datasets include testsequences in different environments and with a different level of complexity to challenge the detection algorithm (for example see Figure 3). Additionally, they probably include a groundtruth dataset such that the accuracy of the detection algorithm can be evaluated (see Figure 4).

The groundtruth images contain 5 labels namely:

- 0 : Static
- 50 : Hard shadow



(a) frame 350 (b) frame 360 (c) frame 370

Figure 3: Input frames from the bungalows sequence.

(a) frame 350 (b) frame 360 (c) frame 370

Figure 4: Groundtruth frames from the bungalows sequence.

- 85 : Outside region of interest
- 170 : Unknown motion (usually around moving objects, due to semi-transparency and motion blur)
- 255 : Motion

This groundtruth only provides detection information for the specified region of interest (ROI) (see Figure 5). As you will see in the dataset, a binary ROI image accompanies the different sequences. The black region in this image has not been annotated with groudtruth information and thus must be ignored.

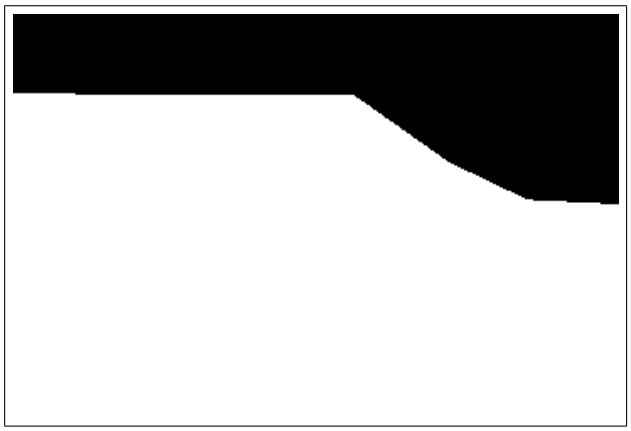


Figure 5: Region of interest of the bungalows sequence.

The downside of such datasets is that the format in which they are provided is not standardized. One dataset will provide you a sequence of images, while the other provides efficiently compressed videos. Luckily, GStreamer is able to cope with all those different formats, so the only thing we will need to do is interface with GStreamer such that the raw video images can be obtained.

3 Application Source and Sink

In order to grab raw data buffers containing single frames from GStreamer, the elements appsrc (see Figure 7) and appsink (see Figure 6) have been developed. These elements make it possible to route the data buffers from inside the pipeline environment towards the application. As you can imagine, in this way, it is very difficult for the Gstreamer environment to control timing, threading conflicts and so on. Therefore, the use of these elements is strongly discouraged in actual products, but for research purposes, these can be very handy. More information about these elements can be found here:

`https://gstreamer.freedesktop.org/documentation/application-development/advanced/pipeline-manipulation.html#manually-adding-or-removing-data-fromto-a-pipeline`

An example is included on minerva.

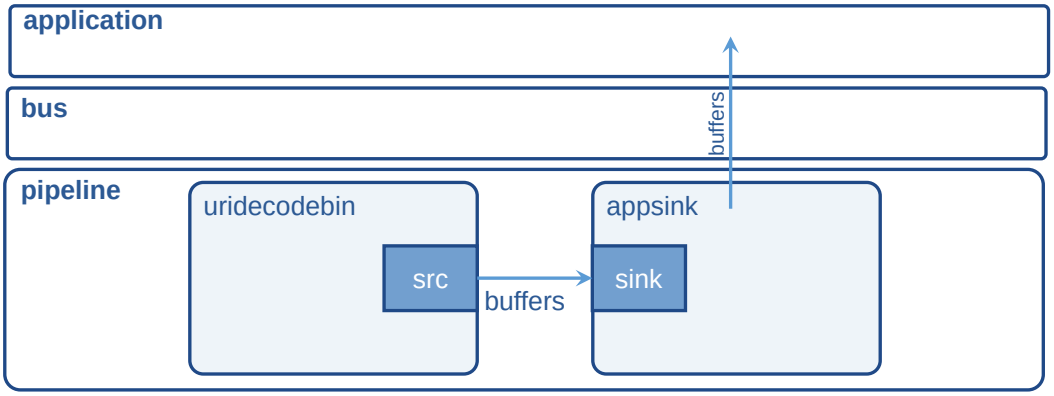


Figure 6: Appsink.

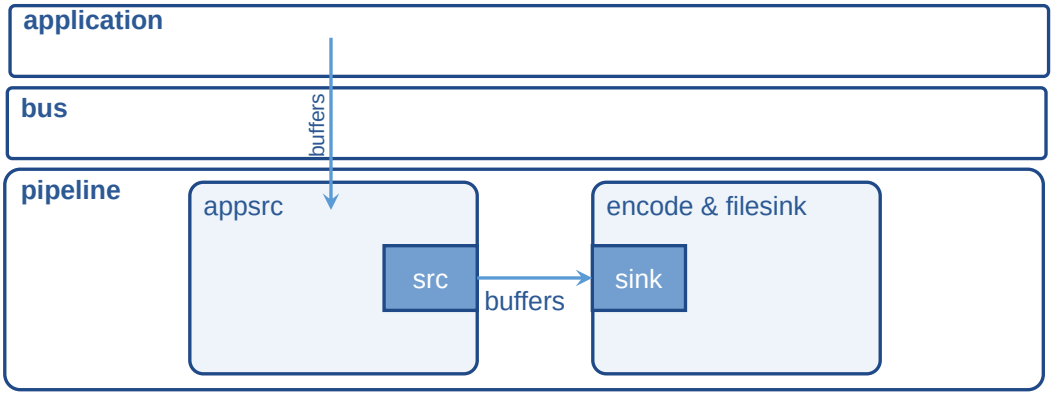


Figure 7: Appsrc.

4 Assignment

In this assignment, we are going to develop a flexible command-line research framework for the improvement of motion detection algorithms. Additionally, we are going to test different motion detection algorithms on their performance.

The program must be called DMA_3_XX.py, the report DMA_3_XX.pdf, and the zip file containing all this information DMA_3_XX.zip.

4.1 Research framework

Program a modular media research framework for motion detection according to the following specifications.

1. Input files must be at a fixed location: /home/dma/Downloads/.
   Files can be found in `users.datasciencelab.ugent.be/MM/DMA_3_video.zip`.
2. The input and the groundtruth must be read using GStreamer. Make a base class called Reader to perform this task.
   The ROI is a single image, so this can be read using OpenCV (cv2.imread()).
3. The groundtruth can be binarized: background (values around 0) and foreground (values around 50, 170, and 255). Ignore the regions outside the ROI (value around 85), so also the first few frames from every sequence.
4. Calculate and visualize your background model and your foreground mask. Visualization must happen on the screen (ScreenWriter) or to a file (FileWriter) using GStreamer. Both of these classes can inherit from a base class called Writer. The visualization on the screen does not require a GUI.
5. Consider a 1% "MotionThreshold" to be applied on the foreground mask in order to convert the frame mask to a binary motion detection for each frame. So if there is less than 1% of foreground, then there is no motion in the frame. So, as a groundtruth, every frame has only one binary value, namely motion or no motion.
6. Evaluate using precision and recall. Make curves or tables depending on the message you would like to convey. Do not only provide results, but state your findings in a paragraph.

Finally, the different motion estimators should inherit from the following base class:

```
import sys
import cv2
```

```
class Estimater(object):

    def __init__(self, max_buffer_size):

        self._max_buffer_size = max_buffer_size
        self._current_index = 0
        self._buffer = []

    def append_to_buffer(self, frame):

        if len(self._buffer) < self._max_buffer_size:
            self._buffer.append(frame)
        else:
            self._buffer[self._current_index] = frame
            self._current_index = (self._current_index + 1) % self._max_buffer_si

    # Feed the motion detector with a frame and
    # return a binary foreground mask:
    # background: 0
    # foreground: 255
    def feed(self, frame):
        raise NotImplementedError
```

4.2 Mean Motion Estimator

Write a mean motion estimator (named MeanMotionEstimator), which calculates the background by taking the mean of the X previous frames. This estimator must be written from scratch only using python or numpy functions. Vary the "ForegroundMaskThreshold" as described in section "2.1 Background subtraction" and fix the algorithm to a suitable value. Document your decision with a graph and some rationale.

Vary X between 5 and 50 and formulate a conclusion about the performance of this parameter X. Use precision recall curves to illustrate the performance of the technique. Do not use more than an A4 page and take care all figures are readable without zooming in on the document. Name your axes on your graph.

4.3 Block Based Estimator

In general, the purpose is to write a block based motion estimator (named BlockBasedEstimator), which detects motion using a threshold on the motion vector size of the different blocks (MotionVectorSizeThreshold). This estimator must also be written from scratch only using python and numpy functions.

More specifically, divide the image in blocks of 8x8. Calculate for each block a motion vector relative to the previous frame. If the motion vector is larger than the MotionVectorSizeThreshold, then all the pixels in this block are categorized as foreground. Again, as in the previous tasks, if there is more than 1% of all the pixels marked as foreground, then label the frame as moving.

Vary the MotionVectorSizeThreshold and formulate a conclusion. Vary the search range between 5 (so 25 evaluations per block) and 13 (so 169 evaluations per block) in steps of 2. Do not pass the border of the frame when trying to search for motion outside the frame. Again, formulate a conclusion about the performance. In the report, also formulate an opinion with respect to algorithm complexity. Do not use more than an A4 page and take care all figures are readable without zooming in on the document.

4.4 Mixture of Gaussians

From the OpenCV library, use a Mixture of Gaussians (MOG) motion estimator and call it MOGOpenCVEstimator. If the MOG has different parameters, look how they impact the performance and formulate a conclusion. Compare the performance/complexity of this estimator to the others. Do not use more than one A4 page and take care all figures are readable without zooming in on the document.

4.5 Making a product at a company

When making the following product, which motion detection algorithm/parameters would you select and why:

1. A battery operated wireless camera to monitor wildlife.
2. A city wide surveillance system with wiring to a central control room.

Formulate your answer as an email report to a manager without an engineering degree. He ordered this research and expects a decision and a high level explanation containing proof by numbers. If he is interested, he will read your full report later.