"Parallel, real-time implementation of motion detection in a sequence of movie frames"

08.06.2022

Table of contents

1 Description general

1.1 Name program

Motion detection in a video - frame sequence.

1.2 Purpose

The main goal of the project was to create a parallel implemented application that detects in real time the motion of objects in a sequence of movie frames.

A key factor in the project was the proper implementation of alignment.

2 Description technologies

2.1 Selected technologies

The following will be used to accomplish the task:

- Python language version 3
- MPI Technology: mpi4py
- OpenCV library and NumPy

3 Description functionality

3.1 Launching program

The program was run using **venv** Python. The file **requirements.txt** contains the necessary libraries. Just use the following command in your environment: `pip install -r requirements.txt`

Run the program using the command line. In order to do so, the name of the program and its arguments must be given:

mpiexec -n PROCESSES python move_detect.py [--fn FILENAME] [--fps FRAMES_PER_SECOND] [--area MIN_AREA]

Call Description:

- mpiexec - command to run a program with MPI
- -n PROCESSES - the number of processes used by the program
- move_detect.py - the name of the main program
- -fn FILENAME - the path to the movie file. If this parameter is not specified, the installed webcam will be used instead of the movie file.
- -fps FRAMES PER SECOND - the number of frames per second to process. The default value is the base frame rate of the source.
- -area MIN AREA - the minimum area of detected objects. The default value is 300.

The aquarium.mp4 file is located in the **/files** directory for a quick test of the *-fn* parameter

3.2 Examples launch

- Run with the source to the movie file being analyzed, the number of fps, and the minimum detection area: **mpiexec -n 4 python move detect.py --fn .*files*.mp4 --fps 10 --area 400**
- Run with minimum detection area, camera will be enabled by default: **mpiexec -n 4 python move detect.py --area 400**
- Running without parameters, the camera will be enabled by default, the fps count will default to the source, and the minimum detection area will be 300: **mpiexec -n 4 python move detect.py**

3.3 Disabling program

You can disable the program by clicking **q** or forcing **CTRL+C** in the console.

3.4 Capabilities

- Parallel object motion detection based on the given movie file.
- Parallel object motion detection based on readout from webcam (WebCam).
- Display feedback video with graphical labeling of objects that are moving.
- Real-time action.

3.5 Limitations

- Uploaded videos should be recorded preferably at stillness. For example, on a tripod.
- It is recommended that objects do not fade very much into the background

4 Format data

4.1 Data input

A .mp4 movie file or a readout from a webcam (WebCam).

4.2 Data output

Window with displayed image and framed detected objects.

5 Operation program

5.1 Implementation of motion detection objects

1. The frames are converted to grayscale.

Figure 1: Greyscale image

2. The resulting image is given a *blur* effect to remove noise.

Figure 2: Blur image

3. Two frames (previous and current) are subtracted using the *absdiff* method (absolute difference).

Figure 3: Absdiff subtraction

4. In order to expose smaller elements, an additional *kernel* array is added and the *dilate* method is used which enlarges

surface of the elements

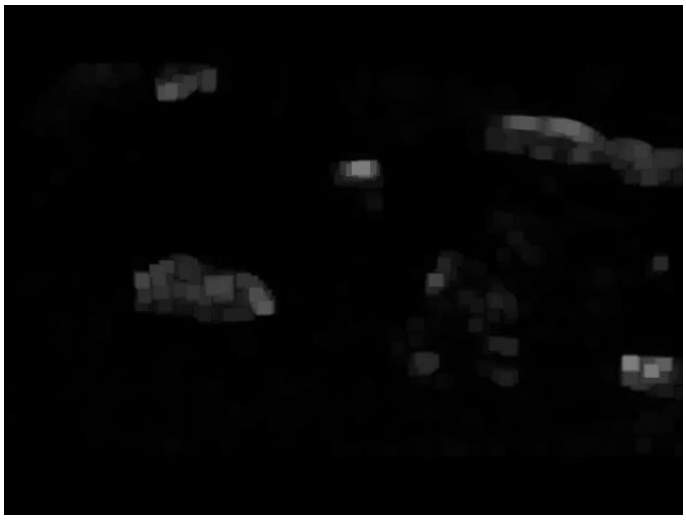

Figure 4: The dilate operation

5. The next step is to obtain a 0/1 matrix using the *thre- shold* method (threshold method)

Figure 5: Operation threshold

6. Finally, the contour of the objects is checked. Parts that are too small are skipped. The outlines are then applied to the unprocessed (original) image.

Figure 6: Final image

6 Measurements

6.1 Comparison of sequential and parallel program operation

Id	Name	Size	Resolution
0	Aquarium	5mb	approximately 480p
1	Webcam	-	approximately 480p
2	Big video	200 mb	1080p

Table 1: Sources examined.

Webcam			
	Sequentially	Parallel N=3	Parallel N=8
approx. 25 FPS	0,012411 s	0,012092 s	0,013129 s
5 FPS	0,062777 s	0,083092 s	0,045694 s

Table 4: Comparison of run times for the "Webcam" source.

Aquarium			
	Sequentially	Parallel N=3	Parallel N=8
30 FPS	0,005602 s	0,002252 s	0,002284 s
5 FPS	0,005202 s	0,002078 s	0,002208 s

Table 2: Comparison of run times for the "Aquarium" source.

Big video			
	Sequentially	Parallel N=3	Parallel N=8
30 FPS	0,122428 s	0,163565 s	0,125910 s
5 FPS	0,136907 s	0,124880 s	0,144244 s

Table 3: Comparison of run times for the "Large Record" source.