**SPM@Unipi-2016/2017**

**Project Report [Video Filtering]**

By: Ahmad Alleboudy

**What has been implemented in the project**

1. A pipeline using fastflow to apply a filter[using OpenCV] on an input video and produce the output result.[parallel.cpp]
2. Same filter implemented in a sequential manner [sequential.cpp]

**Possible major design choices**

1. A fast flow farm with a custom emitter and collector [would need to worry about the frames ordering]
2. A fast flow Ordered Farm with a custom emitter and collector [would not have to worry about the frames ordering as it is already handled by the ordered farm]

I settled with the ordered farm.

**How to compile**

Will need to have OpenCV installed and the fastflow root folder present on the machine

In the project repository main folder run the build.sh file or type in a terminal the following:

g++ -I$FF\_ROOT -I/usr/local/include -I/usr/local/include/opencv/ -I/usr/local/include/opencv2 -L/usr/lib/x86\_64-linux-gnu/ -g -o run parallel.cpp -lopencv\_dnn -lopencv\_ml -lopencv\_objdetect -lopencv\_shape -lopencv\_stitching -lopencv\_superres -lopencv\_videostab -lopencv\_calib3d -lopencv\_features2d -lopencv\_highgui -lopencv\_videoio -lopencv\_imgcodecs -lopencv\_video -lopencv\_photo -lopencv\_imgproc -lopencv\_flann -lopencv\_core -std=c++11 –pthread

Where $FF\_ROOT is where the fastflow root folder

“run” is the name of the resulting compiled binary

“parallel.cpp” is the parallel implementation [could also do sequential.cpp instead]

other imports are for the openCV [should keep only the necessary ones]

**To run the compiled Binary type**

./run path/to/input/video/file.mp4 path/to/output/video/file.mp4 NumberOfWorkersForTheFarm

path/to/input/video/file.mp4 is where the input video resides on the desk

path/to/output/video/file.mp4 is where we wish to have the output video

NumberOfWorkersForTheFarm is the degree of parallelism of the farm