Abstract

This paper presents an adaptive video encoder that can be used to compare the behavior of different adaptation strategies using multiple actuators to steer the encoder towards a global goal, composed of multiple conflicting objectives. A video camera produces frames that the encoder manipulates with the objective of matching some space requirement to fit a given communication channel. A second objective is to maintain a given similarity index between the manipulated frames and the original ones. To achieve the goal, the software can change three parameters: the quality of the encoding, the noise reduction filter radius and the sharpening filter radius. In most cases the objectives – small encoded size and high quality – conflict, since a larger frame would have a higher similarity index to its original counterpart. This makes the problem difficult from the control perspective and makes the case study appealing to compare different adaptation strategies.

1998 ACM Subject Classification B.3.3 Performance Analysis and Design Aids, C.4 Performance of Systems

Keywords and phrases self-adaptive software, video encoding, comparison, control theory

Digital Object Identifier 10.4230/DARTS.3.1.2


http://dx.doi.org/10.1109/SEAMS.2017.16

1 Scope

This artifact presents a Self-Adaptive Video Encoder (SAVE). The encoder simulates the recording and manipulation of a video, using an mp4 stream and processing each of the original frames to obtain a compressed version of the stream. The encoding process’ goal is to reach two conflicting objectives: compress the video so that each frame occupies a specific size and obtain a specific value for a well-known similarity index (SSIM) [2] that compares the original frames with the compressed ones.

To achieve these two conflicting goals, the encoder can change three parameters for each frame: the quality of the encoding, the radius of a sharpen filter applied to the image, and the radius of a noise reduction filter applied to the frame. The quality parameter roughly relates to a compression factor for the image. Its value is between 1 and 100 and represents the percentage of information that is kept in the processed image. However, the relationship between the quality and the size is very difficult to predict, because it depends on the frame content, which is a priori unknown. The sharpen and noise filter process the image. For each pixel, they modify a certain number of pixels that are within a specified radius with respect to the original one. This processing can, for example, remove artifacts that appear due to the compression of the original frame. However, the effect of these filters is not obvious until processing takes place, making it very difficult to develop a good adaptation strategy.

This case study is an extension of the video encoder used to generate some of the results in [1]. The encoder shown in the paper could only modify the quality parameter to obtain a specific similarity index. As a control problem, this was clearly easier than the one presented by this artifact. There was no inter-dependency of multiple parameters on the final results and the presence of one single goal simplified the overall solution. Because of these additional difficulties, we believe that this case study has the potential to unveil many of the complications and the research challenges that still have to be solved in building a proper adaptation strategy for software systems.

With our encoder, we also present a random strategy, a bangbang strategy, and a Model Predictive Control (mpc) alternative, that can be used as a baseline comparison to see how existing adaptation alternatives may behave.

2 Content

The artifact package includes:

- code
  - cts
    * __init__.py
    * bangbang.py: the bang bang adaptation strategy
    * mpc.py: the mpc adaptation strategy
    * random.py: the random adaptation strategy
  - latex
    * figure.tex: the latex code to generate the result figure
  - libs
    * __init__.py
    * mpyc.py: a model predictive control library
    * ssim.py: a library to compute the SSIM
    * utils.py: various utility functions used by the encoder
  - encoder.py: the code of the encoder
3 Getting the artifact

The artifact endorsed by the Artifact Evaluation Committee is available free of charge on the Dagstuhl Research Online Publication Server (DROPS). In addition, the artifact is also available at:

- https://github.com/martinamaggio/save

4 Tested platforms

The artifact has been tested on Ubuntu 16.04. Packages required to make the artifact work are python-imaging, python-numpy, python-scipy, python-matplotlib, python-cvxopt, mplayer, texlive-base, texlive-latex-extra, texlive-pictures.

5 License

The artifact is available under GPL3 license.

6 MD5 sum of the artifact

db50a41ad64af79ed5cf7f1864e7c0

7 Size of the artifact

25038169 bytes (25MB on disk)

References
