2017 IEEE/ACM 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)

# Self-Adaptive Video Encoder: Comparison of Multiple Adaptation Strategies Made Simple

Martina Maggio*, Alessandro Vittorio Papadopoulos†, Antonio Filieri‡ and Henry Hoffmann§
*Department of Automatic Control, Lund University, Sweden; Email: martina@control.lth.se
†Mälärdalen University, Sweden; Email: alessandro.papadopoulos@mdh.se
‡Department of Computing, Imperial College London, United Kingdom; Email: a.filieri@imperial.ac.uk
§Department of Computer Science, University of Chicago, USA; Email: hankhoffmann@cs.uchicago.edu

***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.**

## I. Introduction

Many papers propose adaptation strategies for self-adaptive systems to achieve specific run time objectives. For example, one objective might be to ensure web server response time is below a certain threshold. Another example is minimizing the web server's energy consumption. The list of objectives can have multiple elements, together forming a global goal that the adaptation strategy should reach. Objectives often conflict with one another. For example, decreasing web server response time will most likely lead to higher resource utilization, therefore increasing the energy consumption.

In principle, all these strategies have limitations – conditions in which they do not function correctly – and guarantees on what they can achieve. The system designer wants to choose the best adaptation strategy for a specific set of objectives and working conditions. However, when the execution scenario becomes complicated the theoretical comparison of the limitations and guarantees of each adaptation strategy is hardly fair and may be difficult to interpret. To select the best alternative, we would like to compare strategies based not only on qualitative metrics, but also on their quantitative counterparts.

Testing how a technique performs in a software system with conflicting objectives is a hard problem and often the adaptation techniques cannot be tested in isolation. They run alongside many other things that do not strictly belong to the adaptation strategy. For example, in the Tele Assistance System (TAS) [16] exemplar, multiple services can provide implementations for the analysis of a patient physical conditions. These services expose specific guarantees on their reliability, response time, and cost. For each patient, the most appropriate service should be invoked. This choice is based on the satisfaction of conflicting objectives – for example, minimizing cost while guaranteeing that the analysis is performed on time to discover anomalies. However, the adaptation strategy is not a component running in isolation, and failures can happen at any other level. For example, other services are running in the same underlying infrastructure, like ambulance management. In this case study – but also in others, like the DEECo framework [1], [10] or Internet-Of-Things-based exemplar [4] – it is therefore difficult to isolate the adaptation strategy and compare the results obtained with different alternatives. Yet, to test different adaptation strategies, we want a case study that highlights some the issues an adaptation strategy would face in a real deployment scenario.

To define such a case study, this paper presents our Self-Adaptive Video Encoder (SAVE). Our 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`) [15] 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,

978-1-5386-1550-8/17 $31.00 © 2017 IEEE
DOI 10.1109/SEAMS.2017.16

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 [6]. 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.

## II. The encoder

The developed video encoder is a `python` prototype[1]. To use it, it is not necessary to have a camera connected to the system – any mp4 stream can be used as the data source. The video encoder unpacks a list of mp4 streams that the user places in an input folder and extracts a frame for each of the original frames in the streams. The encoder uses the function `convert`[2] to manipulate the frames.

We emulate the type of manipulations that happen in a real camera system; e.g., to simulate an adaptive video surveillance camera. In this hypothetical scenario, we assume that the video produced by the camera has to be sent on a network, and that the network hosts many different cameras. The network bandwidth becomes a precious and potentially scarce resource, therefore the video encoder should achieve predictability in the amount of information streamed for every frame.

This motivates the choice of our first goal. The `size` of each processed frame is measured by the video encoder after the encoding process terminates. SAVE should achieve a predefined size for each frame, specified at command line as a parameter upon execution. Clearly, we also want to convey the information that was encoded in the original frame. To assess if this is the case, we compute a structural similarity index, the `SSIM` [15], for each frame. While some adaptation schemes might maximize the similarity index given a determined size, we test the adaptation strategy in the presence of conflicting requirements. To do so, we include a setpoint also for the `SSIM`, that can be selected at the command line. More precisely:

- $g_1$, the `SSIM` that quantifies the similarity between the original and compressed frames. `SSIM` is a unitless metric that ranges from 0 to 1, with near 1 indicating similar images. $g_{m,1}$ represents the `SSIM` measured value.
- $g_2$, the frame size (in kilobytes), $g_{m,2}$ represents the measured value of the frame size.

[1]The code is available at https://github.com/martinamaggio/save and it was tested on Linux.

[2]https://www.imagemagick.org/script/convert.php

Clearly, these two goals conflict with one another. When a specific frame size is set, this will correspond to a specific value for the `SSIM` on the frame. Similarly, if a specific `SSIM` is reached, the corresponding frame will have a prescribed size. We conduct tests to show how the adaptation strategy trades one goal for the other to achieve the optimal value for the cost function.

Our control systems can change the following parameters:

- $a_1$, the same `quality` parameter used in [6] to specify the compression density. It ranges between $a_{1,\min} = 1$ and $a_{1,\max} = 100$, where 100 preserves all frame details and 1 produces the highest compression.
- $a_2$, the `sharpen` parameter, which specifies the size of a sharpening filter to be applied to the image. The size ranges between $a_{2,\min} = 0$ and $a_{5,\max} = 5$ where 0 indicates no sharpening.
- $a_3$, `noise`, which specifies the size of a noise reduction filter, which varies between $a_{3,\min} = 0$ and $a_{3,\max} = 5$.

The software is extremely modular with respect to the adaptation strategy. In a folder `ctls` all the code for the adaptation strategies is included. The software contains three different pre-specified adaptation strategies:

- `random`: is an adaptation strategy that selects a random value for the actuators. Clearly, this would be used only as a comparison and is not intended to be by any means a good solution for such a complex problem.
- `bangbang`: is a solution that implements a Bang-Bang adaptation strategy [9]. The strategy bounces between the minimum and maximum value for the actuators, depending on the sign of the error for both objectives. In this case, the `size` objective – the primary one – is tackled with the `quality` actuator, while the other two are used to handle the `SSIM` objective.
- `mpc`: is a model predictive controller, synthesized using the same principles of [2]. The controller tries to minimize a cost function containing terms that factor in the distance from each of the objectives and the use of the actuation strategy. It is possible to express preferences over both (1) which actuators should be used and (2) which objective should be reached in case of unfeasible situations. Moreover, the `mpc` uses a Kalman filter [8] to keep an estimate of the current state updated. Section II-B introduces the parameters used for the `mpc` controller.

The main idea behind the case study, however, is to have a very modular adaptation layer, where a adaptation strategy is easy to implement and replace. The choice of introducing a `random` adaptation strategy is motivated by having a very simple piece of code that shows how to actuate on the system. At the same time, the `bangbang` adaptation strategy is the simplest strategy that we could envision using knowledge about the current progress. These two examples are mostly introduced to show to a user how to develop a new control strategy and to ease the learning phase for the software usage. The `mpc` controller, on the contrary, exemplifies a more complex adaptation strategy. It uses a library that we developed

– included in the `libs` folder – that determines the general behavior of a model predictive controller. At the same time, the controller is initialized with models obtained with experiments on one specific video – the Obama Victory Speech video[3], with a resolution of $854 \times 480$. Once the controller is initialized with the model of the software behavior, the controller operation is standard [5].

The code automatically generates some figures in the format shown by Figure 1 and Figure 2 using Ti*k*Z and LaTeX. Images are found in the result folder.

### A. Developing a new adaptation strategy

To introduce a new adaptation strategy, a user simply needs to add a new `python` file in `ctls`, with the code for the adaptation strategy. It is necessary to modify the file `encoder.py` in three points:

- in the import area, import the new adaptation strategy, for example the line `import ctls.mpc as mpccontroller` is the import for the `mpc` controller;
- initialize the object `controller`, that corresponds to the adaptation strategy just developed (lines 93 and 94 are the initialization for the `random` controller – stating `elif mode == "random": controller = randomcontroller.RandomController()`);
- call the adaptation function specified by the newly developed strategy with the needed arguments (lines 129 and 130 include the call for the `bangbang` strategy – `elif mode == "bangbang": ctl = controller.compute_u(current_outputs, setpoints)`).

The name of the strategy, determined by the `mode` variable is the same name that one uses as command line parameter to invoke that precise strategy. Extensible code was one of the criteria that we had in mind while developing the case study, and it is one of the valuable characteristics of this case study.

### B. Parameters for the Model Predictive Controller

Here we introduce the parameters used for the model predictive controller. The matrices of the system have been identified using a standard identification procedure [11]. The prediction horizon used for the controller is 4, which has shown consistent performance over a set of runs with different videos.

- *Objective weights:* In the model predictive controller (`mpc`), we need to specify weights for the different goals, to indicate some priorities. As `SSIM` is between 0 and 1, we use weight $w_1 = 100$ so that the corresponding component of the cost function is in the hundreds. In the `mpc`, we use a weight $w_2 = 0.001$ so this second goal is considered slightly more important than the first. When the controller can reach only one goal, we prefer to hit the size target, making communication predictable. Since the controller tries to minimize the cost function, a lower weight means that the controller will care less about that component of the cost function, while a higher weight means that even a small improvement in that specific goal will be substantially better from the control perspective.
- *Actuators weights:* We consider the `quality` as our preferred choice for modifying the behavior of the system and we specify a weight $d_1 = 100$ for this actuator in that controller. We select a weight $d_2 = 10^5$ for the `sharpen` actuator, as we would use it as a second choice, together with the following one. Given its reduced range compared to $a_1$, we would like to use it less. For the `noise` actuator, we specify a weight $d_3 = 10^5$, equivalent to `sharpen`.

[3]https://www.youtube.com/watch?v=nv9NwKAjmt0

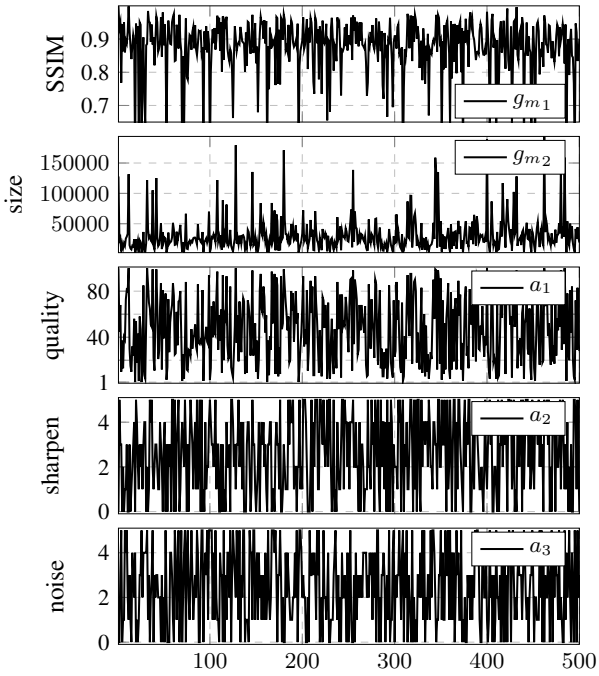



Fig. 1: Results for the video experiment with the Obama Victory Speech video, with the `random` adaptation strategy.

## III. Experiments

We present here the results of some experiments obtained with the developed case study and its adaptation strategies.

Figure 1 shows the first 500 frames encoded with one run of the `random` strategy. As can be seen, all the actuators vary in the determined domain in a random fashion. As a response to that, the `size` and the `SSIM` change. However, it is possible to see that the similarity index value is quite noisy but around 0.86 (the average value, when outliers are eliminated). This testifies that there is a complex relationship between the actuator values that are set by the `random` strategy and the behavior of the desired variables. For this reason, it is quite difficult to achieve a proper and smooth control of the involved quantities.

We have conducted some tests with the `bangbang` strategy for the same frames. The strategy is configured such that a minimum value for the `quality` parameter is selected (20)

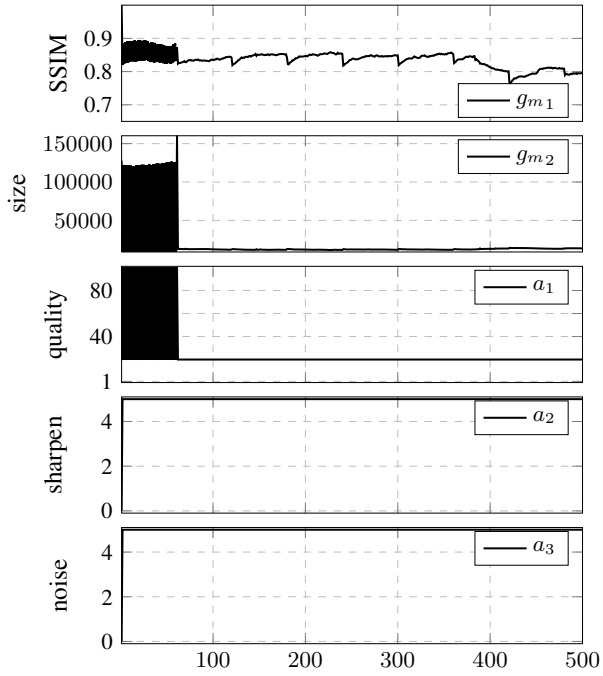



Fig. 2: Results for the video experiment with the Obama Victory Speech video, with the `bangbang` adaptation strategy and a setpoint of 0.7 for the `SSIM` and 10000 for the size.

in case the size is above the desired one, and the maximum value (100) is selected in the opposite case. On the contrary, the `sharpen` and `noise` values are set to the maximum value (5) when the measured `SSIM` is below the setpoint and to the minimum value (0) in the opposite case. As can be seen in Figure 2, the strategy has difficulty achieving both the objectives that correspond to the goal. In the first frames, the `quality` varies tremendously between the maximum value and the minimum one, as a result in the size change. When the size stabilizes, the strategy finds an equilibrium point which is, however, non-optimal.

For the `mpc` solution, we run the video compression example using the Obama Victory Speech video, with different combinations of goals $g_1$ and $g_2$. Specifically, we run all possible combinations where $g_1 \in \{0.7, 0.8, 0.9\}$ and $g_2 \in \{8000, 10000, 15000\}$. Notice that this is a stress test. In fact, even setting the values of `quality`, `sharpen` and `noise` that would achieve the lowest possible `SSIM`, this value hardly ever becomes lower than $0.75$, therefore the $0.7$ setpoint is not feasible. Also, the goals' conflicting nature makes it impossible to reach most goal combinations simultaneously. For example, when $g_1 = 0.9$, the frame size often exceeds $15000$.

Figure 3 shows the nine different experiments with the different values of $g_1$ and $g_2$. In these experiments, there are only two feasible values for the setpoints: (1) $g_1 = 0.8$ and $g_2 = 8000$ (Figure 3b) and (2) $g_1 = 0.9$ and $g_2 = 15000$ (Figure 3i). In all others it is impossible to achieve both the `SSIM` and frame size setpoints. Therefore, as shown in the figures, the controller opts to reach $g_2$, which has an higher relevance: $w_2 \times g_2$. In the first row, Figure 3a shows that $g_2$ and $g_{m2}$ are basically equal, while the achieved `SSIM` $g_{m1}$ is higher than desired. The encoding `quality` $a_1$ is kept low and there is no active `noise` compensation, while the `sharpen` value $a_2$ varies during the execution. Figure 3b shows that both the `SSIM` and the size setpoint are achieved using some sharpening, a small amount of noise reduction, and a quality similar to that used for the previous combination of setpoints. When the `SSIM` goal is increased – so, information loss should be diminished – even more noise correction and sharpening is added, as shown in Figure 3c. The setpoint $g_1$ is reached for some frames, but overall the size limit (and the fact it is weighted more heavily in the cost function) leads to `SSIM` below the setpoint.

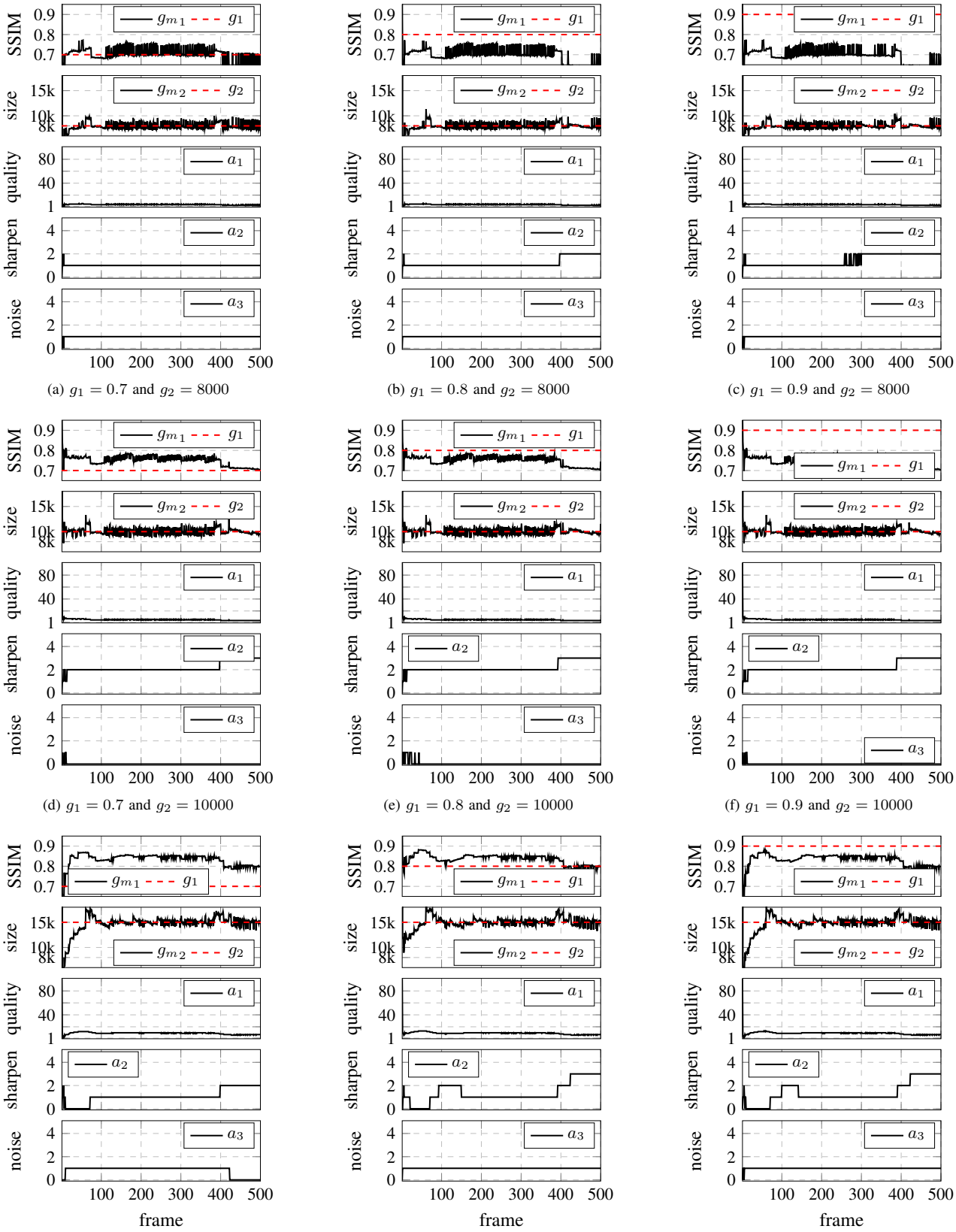
Figures 3d, 3e and 3f show that by allowing a higher frame size the `quality` is increased, however the frame size does not allow for a precise control of the `SSIM` $g_1$, which is not exceeded when its setpoint is equal to $0.7$ and optimized as much as possible when it is $0.8$ and $0.9$. Similar to Figure 3c, the noise reduction is used by the controller in the experiment corresponding to Figure 3f to achieve a better similarity index without increasing the frame size.

Figures 3g, 3h and 3i show that it is possible to achieve $g_1 = 0.9$ and $g_2 = 15000$ by selecting the values of $a_1$, $a_2$ and $a_3$ and the controller therefore selects appropriate values to achieve both the setpoints. In the opposite case, as seen in Figures 3g and 3h, the size setpoint is achieved, while the similarity index is kept as close as possible.

## IV. Discussion

We have done a quite extensive set of tests with the proposed case study, trying to identify a suitable linear Multiple-Input Multiple-Output (MIMO) model that could represent the behavior of the software when `quality`, `sharpen`, and `noise` are changed. More precisely, we tried to find a linear model that could describe the variation of the `size` and the `SSIM` given the variation of our actuation quantities. In this process we have discovered that there is no good linear approximation of the model that would work with many different videos and in all the different phases that the videos expose (for example, a video that records a conference talk is mostly static, while a video of a sport event is very dynamic with frequent changes of scene). This introduces a non-negligible difficulty for adaptation strategies based on *control theory* and because of that difficulty we consider this an interesting case study for control-theoretical adaptation [3], [7], [14].

Another difficulty in this case study is the interference between the actuators' values and the goals. All three actuators influence both the goals, in conflicting ways, so that it is often impossible – even testing all the possible configurations – to meet all the objectives at the same time. This interference is a problem regardless of the adaptation strategy and this case study has the potential of fostering research on how to




(a) $g_1 = 0.7$ and $g_2 = 8000$ (b) $g_1 = 0.8$ and $g_2 = 8000$ (c) $g_1 = 0.9$ and $g_2 = 8000$

(d) $g_1 = 0.7$ and $g_2 = 10000$ (e) $g_1 = 0.8$ and $g_2 = 10000$ (f) $g_1 = 0.9$ and $g_2 = 10000$

(g) $g_1 = 0.7$ and $g_2 = 15000$ (h) $g_1 = 0.8$ and $g_2 = 15000$ (i) $g_1 = 0.9$ and $g_2 = 15000$

Fig. 3: Results for the video experiment with the Obama Victory Speech video, with the `mpc` controller, various setpoints for both the `size` and the `SSIM`.

*recognize* and *exploit* the inherent trade-offs present in real case studies with complex interactions between goals and actuators.

Another reason why developing an adaptation strategy for this specific case study is challenging is that a new video will require adjustments. These adjustments should be automatic at least to the extent this is possible – if one thinks of a scenario in which the video is a stream of a surveillance camera, there is no way to optimize for uncertain situations. There is always a probability that the scene will be different (partially or completely). In a recorded video, this can happen because the scene has changed, for example from the speaker or the athlete to the public. In a live stream because some people have entered the camera range. The uncertainty in the domain of video encoding makes this case study very interesting for all the research that deals with uncertainty reduction and uncertainty management at runtime.

However, one must also note that not all the challenges in developing adaptation strategies are now covered by this case study. For example, acuation latency [12], [13] is not a concern for this case study. One could extend the current work to include for example network latency in the control signal actuation, as if the computation was done remotely and a specific quality was requested by a streamer.

We also believe that there are many other extensions where SAVE becomes one of the components in a more complex system. This more complex system could, for example, stream the video at a certain rate that depends on the client that is requesting the video. Per-video optimization can be envisioned in the case of a streaming-on-demand service, where videos are retrieved from a common source.

## V. Conclusion

In this paper we have presented SAVE, a self-adaptive video encoder that realizes a case study for the development and comparison of adaptation strategies. The code for SAVE can be found at https://github.com/martinamaggio/save We have developed SAVE with extensibility in mind and we have equipped it with simple strategies that demonstrate how to write an adaptation strategy without having to read – or even understand – the encoder. We have highlighted why we believe that SAVE is an excellent case study to compare the behavior of different adaptation strategies and what we believe are the challenges there.

**Acknowledgements**: This work was partially supported by the Swedish Research Council (VR) for the projects "Cloud Control" and "Power and temperature control for large-scale computing infrastructures", through the LCCC Linnaeus and ELLIIT Excellence Centers. It was also partially supported by the Swedish Foundation for Strategic Research under the project "Future factories in the cloud (FiC)" with grant number GMT14-0032. Henry Hoffmann's work on this project was partially funded by the U.S. Government under the DARPA BRASS program, by the Dept. of Energy under DOE DE-AC02-06CH11357, by the NSF under CCF 1439156, and by a DOE Early Career Award.

Find the artifact at: http://www.martinamaggio.com/papers/seams17/, Username and password for the virtual machine are respectively 'user' and 'seams2017'.

## References

[1] R. Al Ali, T. Bures, I. Gerostathopoulos, P. Hnetynka, J. Keznikl, M. Kit, and F. Plasil. Deeco: An ecosystem for cyber-physical systems. In *Companion Proceedings of the 36th International Conference on Software Engineering*, ICSE Companion 2014, pages 610–611, New York, NY, USA, 2014. ACM.

[2] K. Angelopoulos, A. V. Papadopoulos, V. E. Silva Souza, and J. Mylopoulos. Model predictive control for software systems with cobra. In *Proceedings of the 11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS '16, pages 35–46, New York, NY, USA, 2016. ACM.

[3] D. Arcelli and V. Cortellessa. Challenges in applying control theory to software performance engineering for adaptive systems. In *Companion Publication for ACM/SPEC on International Conference on Performance Engineering*, ICPE '16 Companion, pages 35–40, New York, NY, USA, 2016. ACM.

[4] A. Bennaceur, C. McCormick, J. G. Galán, C. Perera, A. Smith, A. Zisman, and B. Nuseibeh. Feed me, feed me: An exemplar for engineering adaptive software. In *Proceedings of the 11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS '16, pages 89–95, New York, NY, USA, 2016. ACM.

[5] E. Camacho and C. Bordons. *Model Predictive Control*. Advanced Textbooks in Control and Signal Processing. Springer London, 2004.

[6] A. Filieri, H. Hoffmann, and M. Maggio. Automated design of self-adaptive software with control-theoretical formal guarantees. In *Proceedings of the 36th International Conference on Software Engineering*, ICSE, pages 299–310, New York, NY, USA, 2014. ACM.

[7] A. Filieri, M. Maggio, K. Angelopoulos, N. D'Ippolito, I. Gerostathopoulos, A. B. Hempel, H. Hoffmann, P. Jamshidi, E. Kalyvianaki, C. Klein, F. Krikava, S. Misailovic, A. V. Papadopoulos, S. Ray, A. M. Sharifloo, S. Shevtsov, M. Ujma, and T. Vogel. Software engineering meets control theory. In *Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS '15. IEEE, 2015.

[8] A. Gelb. *Applied Optimal Estimation*. MIT Press, 1974.

[9] H. Hermes and J. P. Lasalle. *Functional Analysis and Time Optimal Control*. Mathematics in Science and Engineering. Elsevier Science, 1969.

[10] M. Kit, I. Gerostathopoulos, T. Bures, P. Hnetynka, and F. Plasil. An architecture framework for experimentations with self-adaptive cyber-physical systems. In *Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS '15, pages 93–96, Piscataway, NJ, USA, 2015. IEEE Press.

[11] L. Ljung. *System Identification: Theory for the User*. Prentice Hall information and system sciences series. Prentice Hall PTR, 1999.

[12] G. A. Moreno, J. Cámara, D. Garlan, and B. R. Schmerl. Proactive self-adaptation under uncertainty: a probabilistic model checking approach. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2015, Bergamo, Italy, August 30 - September 4, 2015*, pages 1–12, 2015.

[13] G. A. Moreno, J. Cámara, D. Garlan, and B. R. Schmerl. Efficient decision-making under uncertainty for proactive self-adaptation. In *2016 IEEE International Conference on Autonomic Computing, ICAC 2016, Wuerzburg, Germany, July 17-22, 2016*, pages 147–156, 2016.

[14] T. Patikirikorala, A. Colman, J. Han, and L. Wang. A systematic survey on the design of self-adaptive software systems using control engineering approaches. In *Proceedings of the 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS '12, pages 33–42, Piscataway, NJ, USA, 2012. IEEE Press.

[15] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[16] D. Weyns and R. Calinescu. Tele assistance: A self-adaptive service-based system examplar. In *Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS '15, pages 88–92, Piscataway, NJ, USA, 2015. IEEE Press.