ViViD: A Variability-Based Tool for Synthesizing Video Sequences

Mathieu Acher, Mauricio Alférez, José Angel Galindo Duarte, Pierre Romenteau, Benoit Baudry

ViViD: A Variability-Based Tool for Synthesizing Video Sequences

Mathieu Acher
DiverSE Team, Inria
University of Rennes 1, France
mathieu.acher@inria.fr

Mauricio Alférez
DiverSE Team, Inria
Rennes, France
mauricio.alferez@inria.fr

José A. Galindo
DiverSE Team, Inria
Rennes, France
jagalindo@inria.fr

Pierre Romenteau
InPixal
Rennes, France
pierre.romenteau@inpixal.com

Benoit Baudry
DiverSE Team, Inria
Rennes, France
benoit.baudry@inria.fr

ABSTRACT

We present ViViD, a variability-based tool to synthesize variants of video sequences. ViViD is developed and used in the context of an industrial project involving consumers and providers of video processing algorithms. The goal is to synthesize synthetic video variants with a wide range of characteristics to then test the algorithms. We describe the key components of ViViD (1) a variability language and an environment to model what can vary within a video sequence; (2) a reasoning back-end to generate relevant testing configurations; (3) a video synthesizer in charge of producing variants of video sequences corresponding to configurations. We show how ViViD can synthesize realistic videos with different characteristics such as luminances, vehicles and persons that cover a diversity of testing scenarios.

Categories and Subject Descriptors

D.2.2 [**Software Engineering**]: Design tools and techniques; D.2.5 [**Testing and Debugging**]: Testing tools; D.2.13 [**Reusable Software**]: Domain engineering

General Terms

Algorithms, Design, Experimentation, Multimedia

Keywords

Variability Modeling, Combinatorial Interaction Testing, Prioritization, T-wise, Video Generation

1. INTRODUCTION

Acquisition, processing and analysis of video sequences has many applications in our modern society [1, 2]. Some application examples are surveillance systems, computer-aided medical imaging systems, traffic control, and mobile robotics. Such systems are software-intensive and rely on signals captured by video cameras, which are then processed through a chain of algorithms. Signal processing algorithms are assembled in different ways, depending on the goal of image recognition (e.g., patterns identification, objects tracking). The algorithms produce information helpful for humans or subject to subsequent analysis. For example, algorithms could determine rectangles that cover the zone of a specific object in a scene. This information is useful for human observers that can react accordingly, e.g., to trigger alarms when tracking the movement of a vehicle or a person.

Problem. Numerous algorithms for video analysis have been and will be developed – each specialized in a specific task (e.g., segmentation, object recognition). Even for a category of algorithm, many alternatives exist; a one-size-fits-all solution capable of handling the diversity of scenarios and signal qualities is hardly conceivable. An algorithm may excel in tracking a vehicle in a desert but completely fails when the luminance is low or some distractors are present in the video scene. Another algorithm may be less performant in a normal situation, but more robust when luminance is changed and distractors are added. The diversity of situations poses a challenge for both developers and users of video algorithms. For consumers (users), the practical problem is to determine what algorithms are likely to fail or excel in certain conditions before the actual deployment in realistic settings. For providers (developers), how to have confidence and produce evidence that the algorithms are capable of handling certain situations?

A fundamental and common challenge is the *testing* of algorithms. Practitioners face severe difficulties to get an input test suite (i.e., a set of video sequences) large and diverse enough to test (e.g., benchmark) the algorithms. The current practice is indeed to find out some existing videos or film video sequences outside. The effort of manually collecting videos is highly consuming in resources and economically not viable. In addition, practitioners have limited control over the scenarios covered or not by the set of video sequences. We give more details in Section 2 about the industrial context and problem. As a result, the major challenge for testing the algorithms remains: *How to obtain a suitable and comprehensive input set of video sequences?*

Solution: ViViD. In this paper, we present ViViD, a variability-based video sequences synthesis tool. ViViD is based on variability modeling techniques and languages for

SPLC '14 Florence, Italy

(a) Variant #1 of video sequence

(b) Variant #2 of video sequence

(c) Variant #3 of video sequence

(d) Variant #4 of video sequence

(e) Variant #5 of video sequence

(f) Variant #6 of video sequence

Figure 1: Six variants of video sequences synthesized with ViViD

synthesizing video variants – only video sequences that fulfill some criteria are generated. The first criterion to generate a set of video sequences (a.k.a., a test data set) is that any pair of values of features/attributes (e.g., luminance_mean=72, tilt_motion=0.4) will be covered in at least one video sequence. ViViD can also prioritize the input data sets according to one or more optimization functions, for example, synthesis time or any ad-hoc function. ViViD is a comprehensive solution with a textual editor (integrated in Eclipse), solvers, and a connection with an industrial video generator in charge of synthesizing video sequence variants. ViViD is build in a way that mimic feature model automated analysis [3] process to obtain only meaningful video sequences.

Previous research proposed to use different metrics to optimize test-suites for concrete users needs in variability-intensive systems [4–6]. These approaches allowed assigning more importance to some inputs than others when testing. However, they only focused on functional testing of the main system features without considering different testing objectives including quality *attributes* as part of the testing space. Other evolutionary-based approaches do not consider testing aspects [7].

The intended audience of the demonstration are twofold. First, SPL practitioners and researchers can learn a new variability modeling language with advanced language constructs (e.g., attributes, meta-information, deltas over values) and an efficient reasoning backend to support combinatorial testing and multi-objective prioritization. ViViD is built upon the theoretical foundations exposed in [8] and our industrial experience in designing a variability language `https://github.com/ViViD-DiverSE/VM-Source`. Second, the application of an SPL-based tool in an original

domain (video) and in an industrial context can be of interest both for researchers and practitioners. We will demonstrate how ViViD can synthesize numerous realistic video variants (see Figure 1).

2. AN INDUSTRIAL PROBLEM IN THE VIDEO DOMAIN

We face the challenge of synthesizing a high number of diverse video sequences variants in an industrial project. The MOTIV project aims at evaluating computer vision algorithms such as those used for surveillance or rescue operations. A targeted scenario is usually as follows. First, airborne or land-based cameras capture on-the-fly videos. Then, a video processing chain analyzes video sequences to detect and track objects, for example, survivors in a natural disaster. Based on that information the operation manager triggers a rescue mission quickly based on the video sequence analysis information.

Two companies are part of the MOTIV project as well as the DGA (the French governmental organization for defense procurement). The two companies develop and provide numerous algorithms for video analysis. The diversity of scenarios and signal qualities poses a difficult problem for all the partners of MOTIV: which algorithms are best suited given a specific application? From the consumer side (the DGA), how to choose, select and combine the algorithms? From the provider side (the two companies), how to guarantee that the algorithms meet a large variety of situations? How to propose innovative solutions able to handle new situations?

Our partners need to collect video sequences to test their video analysis solutions and different detection algorithms. Synthesizing a high diversity of video sequences is difficult as there are many ways in which a video can change (e.g., physical properties, types and number of objects, backgrounds). Yet, synthesizing video sequences is still more feasible than filming them in real environments. Our partners calculate that an initial input data set of 153000 video sequences (of 3 minutes each), corresponds to 320 days of video footage and requires 64 years of filming outside (working 2 hours a day). These numbers were calculated at the starting point of the project based on the previous experiences of the partners. A related problem to the difficulty of synthesizing video sequences is that practitioners ignore what kinds of situations are covered or not by the set of video sequences. Therefore, it is not possible to ensure the quality of the video analysis solutions and detection algorithms in all situations. Overall, more *automation* and *control* are needed to synthesize video sequence variants and cover a diversity of testing scenarios.

3. THE ViViD TOOL

We promoted the use of variability techniques to address the problem: (1) a variability model to formally document what can vary within a video sequence; (2) an exploitation of the variability model to generate a certain number of testable configurations; (3) a synthesis of video variants corresponding to configurations.

We developed a tool, called ViViD, in close collaboration with the industrial partners of MOTIV. A website with all the components of ViViD is available at `https://github.com/ViViD-DiverSE/`.The ViViD tool supports a novel approach that improves the current practice of filming different video sequences. We describe this tool by means of its functionality, architecture and special features.

3.1 Functionality

The main functionality of ViViD are as follows:

- support of variability modeling with a dedicated language, called VM. We design the language throughout different iterations `https://github.com/ViViD-DiverSE/VM-Source` with the partners. In particular the language can be used for modeling physical, numerical properties of videos while the information of the variability model can be exploited by a video generator;
- reasoning support to prioritize pair-wise configurations while maximizing or minimizing ad-hoc objective functions. The rationale is that the number of valid configurations in a variability model can be huge. Practitioners thus need efficient techniques to get a relevant and representative subset. Combinatorial techniques are applied to produce a subset that covers many kinds of situations. Objective functions help practitioners to focus on certain kinds of situations (e.g., by prioritizing some features)
- support for synthesizing video sequences. We reuse the video generator developed by the industrial partner. The generator is written in Lua[1] language. There are extra activities that support the creation of the core assets that compose video sequences. For example, an engineering effort is needed to model 3D objects and creating masks to allow to overlap moving objects in the background. We take those extra activities as prerequisites of the ViViD tool. They are not part of the tool demonstration and out of the scope of the paper.

3.2 Architecture and Main Components

Figure 2 relates the above mentioned actions to the ViViD tool chain. We mark with red the video property luminance_mean and its trace through the entire tool chain.

In the variability model specified with the VM language, luminance_mean was modeled as a real attribute of the feature signal_quality. The attribute will automatically receive a value in each one of the configurations, for example, the upper-most Lua configuration file assigns a value of 72.55 to the signal_quality.luminance_mean of the configuration file for one video sequence variant. Finally, the video sequences generator has functions in charge of producing a video sequence with the features and attribute values established in the configuration files. The ViViD tool chain is composed of three main tools:

3.2.1 VM *model editor*

Developers and video domain experts use the VM language editor to create a VM model of the video domain. The VM language is used to model attributed features models while adding extra information that eases the analysis of variability. This extra information is listed in Section 3.3.

This editor was created using the open source Xtext language workbench (`http://www.xtext.com`). Xtext helped us to covers all aspects of the VM language infrastructure,

[1] `http://www.lua.org/`

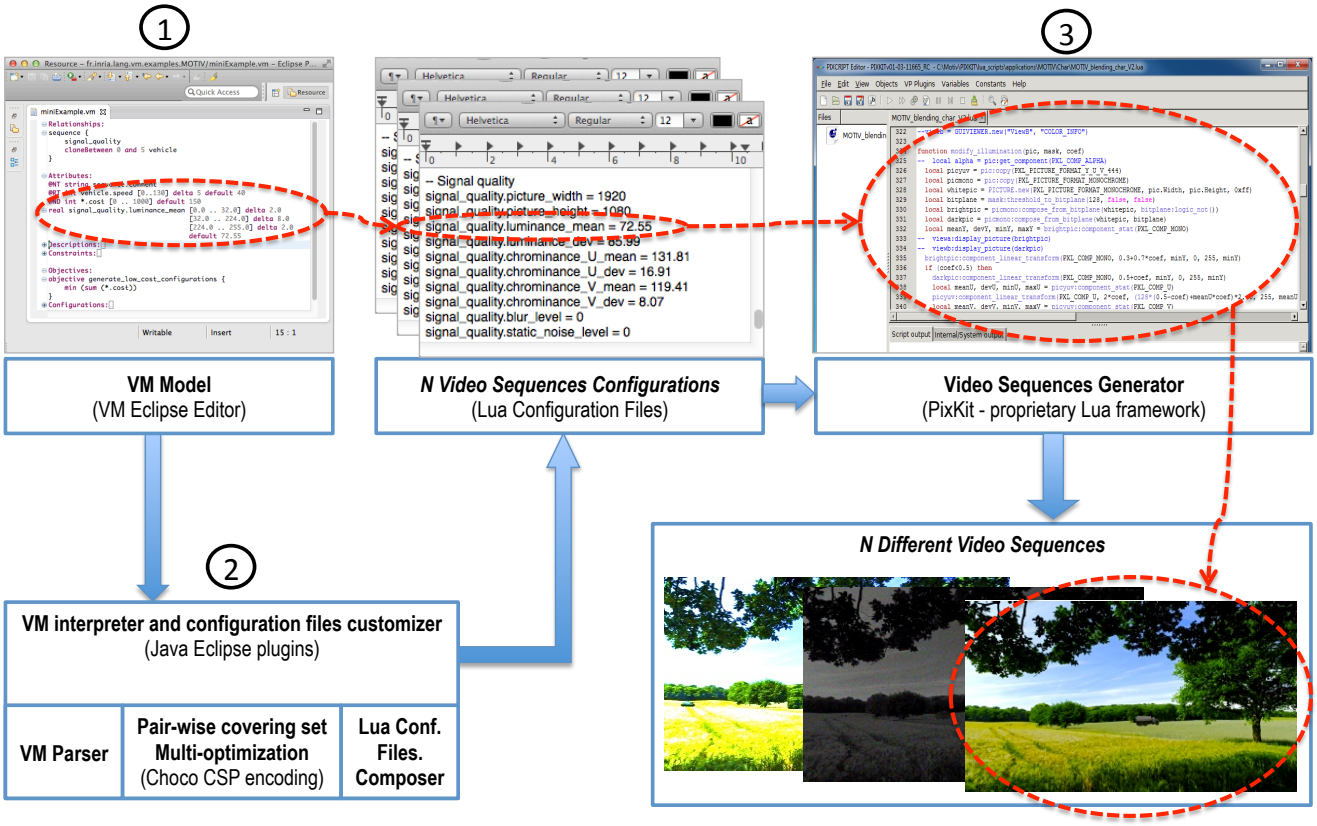

Figure 2: ViViD Tool chain

such as parsing, interpretation, user plugins code generation and Eclipse IDE integration.

3.2.2 VM *interpreter and configuration files customizer*

The VM interpreter and configuration files customizer is a complex piece of code that supports three main activities: i) to parse the VM model , ii) to translate the VM model to a Constraint Satisfiability Problem (CSP) to prioritize pair-wise video sequences configurations while maximizing or minimizing ad-hoc objective functions, and iii) to calculate and compose one video sequence by each one of the N Lua configuration files.

The VM interpreter was created as a Java application that communicates with an advanced CSP solver called Choco (`http://www.emn.fr/z-info/choco-solver/`) to calculate optimal video sequences configurations.

The information about the configurations is stored internally by the VM model and can be displayed in different formats: i) the VM language format for configurations, and ii) the Lua configuration files format. The first format is useful to create extra configurations manually or to visualize multiple configurations in a same file using a nice and abbreviated concrete syntax (e.g., grouping value assignments by attribute instead by feature). The second format is useful for the video sequences generator as this is the format that it reads faster.

3.2.3 *Video sequences generator*

The video sequences generator is composed of an editor and video manipulation libraries created entirely in C code and the embeddable scripting language Lua. The reason for using Lua tools is that it is very suitable to be used in small devices as it runs fast and it is considered very lightweight (the entire Lua language engine source contains around 20000 lines of C code).

Lua combines simple procedural syntax with powerful data description constructs based on associative arrays and extensible semantics. Lua is dynamically typed, runs by interpreting bytecode for a register-based virtual machine, and has automatic memory management with incremental garbage collection. These characteristics made Lua ideal for configuration, scripting, and rapid prototyping activities required by the ViViD tool.

3.3 Special Features in Action

One of the special features in the ViViD tool chain is the large support for variability modeling in the video domain. The domain is characterized by the presence of parameters with values defined in continuous domains (e.g., real numbers), and the use of configurable copies of objects, called multi-features (e.g., multiple vehicles and people).

Listing 1 shows a small example of the video sequences variability model written in the VM language that is part of the ViViD tool. The VM language is special as domain experts can specialize advanced variability models by including additional information. The most common types of additional information are:

- *default* values that help to complete partial configura-

```
1  Relationships:
2  sequence {
3    signal_quality
4    cloneBetween 0 and 5 vehicle
5    //. . .
6  }
7
8  Attributes:
9  @NT string sequence.comment
10 @RT int vehicle.speed [0..130] delta 5 default 40
11 @ND int *.cost [0 .. 1000] default 150
12 real signal_quality.luminance_mean
13    [0.0 .. 32.0] delta 2.0
14    [32.0 .. 224.0] delta 8.0
15    [224.0 .. 255.0] delta 2.0
16    default 72.55
17 //. . .
18
19 Descriptions: //. . .
20
21 Constraints:  //. . .
22
23 Objectives:
24 objective generate_low_cost_configurations {
25   min (sum (*.cost))
26 }
27 Configurations: //. . .
```

Listing 1: Example variability model written in the VM language

tions,

- *deltas* to discretise real domains,
- controlled *descriptions* in natural language about the model itself, features, attributes and constraints,
- *multi-ranges* and *multi-deltas* that allow to use several ranges and deltas to define the domain of the values of an attribute (e.g., luminance_mean in Line 12 has three ranges),
- meta-information annotations such as: *not translatable –@NT* (tags a purely informative element, such as the scene *comment* in Line 9), *not decidable –@ND* (a element that does not help to differentiate (visually) one video sequence from other, such as *cost* in Line 11), and *runtime –@RT* (a feature or attribute that vary only at runtime, such as vehicle speed in Line 20), and
- *objective functions*, such as generating only low cost video sequence configurations, as shown in Line 23-25.

A website with the VM grammar, full-fledged editor and user guide is available at `http://mao2013.github.io/VM/`.

4. CONCLUSIONS

We presented ViViD, a tool that synthesizes video sequence variants. ViViD is based on variability techniques: (1) a variability language and an environment to model what can vary within a video sequence; (2) variability-aware generation of relevant, testing configurations. A video generator produces variants of video sequences corresponding to configurations. ViViD can synthesize realistic videos with different characteristics (e.g., variations of luminance, adding of vehicles and persons) to cover a diversity of testing scenarios.

We are developing ViViD with industrial partners in the context of a research project (MOTIV). We are now in the process of launching a large-scale testing experience involving hundreds of video sequences – something not possible at the beginning of the project. The ViViD tools has different licenses, while the VM language and the VANE tool are distributed as open source, the video-sequence generator has a close license. Those free parts of the tool can be found in the tool website and are being integrated in FaMiLiar [9].

We hope that the demonstration of the ViViD tool and its design can inspire other practitioners to apply variability modeling, combinatorial testing, multi-objective prioritization and product synthesis techniques in other original domains and industrial contexts.

Acknowledgements

This works was financed by the project MOTIV of the Direction Générale de l'Armement (DGA) - Ministère de la Défense, France.

5. REFERENCES

[1] J. R. Parker, *Algorithms for image processing and computer vision*. Wiley. com, 2010.

[2] J. Ponce, D. Forsyth, E.-p. Willow, S. Antipolis-Méditerranée, R. d'activité RAweb, L. Inria, and I. Alumni, "Computer vision: a modern approach," *Computer*, vol. 16, p. 11, 2011.

[3] D. Benavides, S. Segura, and A. Ruiz-Cortés, "Automated analysis of feature models 20 years later: a literature review," *Information Systems*, vol. 35, no. 6, 2010. [Online]. Available: http://dx.doi.org/10.1016/j.is.2010.01.001

[4] C. Henard, M. Papadakis, G. Perrouin, J. Klein, and Y. L. Traon, "Multi-objective test generation for software product lines," in *Proceedings of the 17th International Software Product Line Conference*, ser. SPLC '13. New York, NY, USA: ACM, 2013, pp. 62–71. [Online]. Available: http://doi.acm.org/10.1145/2491627.2491635

[5] M. F. Johansen, O. y. Haugen, and F. Fleurey, "An algorithm for generating t-wise covering arrays from large feature models," *Proceedings of the 16th International Software Product Line Conference on - SPLC '12 -volume 1*, p. 46, 2012. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2362536.2362547

[6] G. Perrouin, S. Oster, S. Sen, J. Klein, B. Baudry, and Y. Traon, "Pairwise testing for software product lines: comparison of two approaches," *Software Quality Journal*, pp. 605–643, Aug. 2011. [Online]. Available: http://www.springerlink.com/index/10.1007/s11219-011-9160-9

[7] A. S. Sayyad, J. Ingram, T. Menzies, and H. Ammar, "Scalable product line configuration: A straw to break the camel's back," in *ASE*. IEEE, 2013, pp. 465–474.

[8] J. A. Galindo, M. Alférez, M. Acher, and B. Baudry, "A variability-based testing approach for synthesizing video sequences," in *ISSTA*, To appear in 2014.

[9] M. Acher, P. Collet, P. Lahire, and R. France, "Familiar: A domain-specific language for large scale management of feature models," *Science of Computer Programming (SCP) Special issue on programming languages*, p. 22, 2013.