A Variability-Based Testing Approach for Synthesizing Video Sequences

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

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

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

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

David Benavides
University of Seville, Spain
benavides@us.es

ABSTRACT

A key problem when developing video processing software is the difficulty to test different input combinations. In this paper, we present VANE, a variability-based testing approach to derive video sequence variants. The ideas of VANE are i) to encode in a variability model what can vary within a video sequence; ii) to exploit the variability model to generate testable configurations; iii) to synthesize variants of video sequences corresponding to configurations. VANE computes T-wise covering sets while optimizing a function over attributes. Also, we present a preliminary validation of the scalability and practicality of VANE in the context of an industrial project involving the test of video processing algorithms.

Categories and Subject Descriptors

D.2.5 [**Software Engineering**]: Testing and Debugging—*Testing tools*; D.2.m [**Software Engineering**]: Reusable Software

General Terms

Theory, Algorithms, Performance

Keywords

Variability, Combinatorial testing, Video analysis

1. INTRODUCTION

Video analysis systems are ubiquitous and crucial in modern society [17, 21]. Their applications range from video protection and crisis monitoring to crowds analysis. *Video sequences* are acquired, processed and analyzed to produce numerical or symbolic information. The corresponding information typically raises alerts to human observers in case of interesting situations or events. For instance, a classical scenario in natural disasters is to recognize survivors by using airborne cameras with the intention of rapidly acting based on information gleaned to achieve strategic or tactical rescue.

Depending on the goal of the video sequence recognition, signal processing algorithms are assembled in different ways. Also, each algorithm is a complex piece of software, specialized in a specific task (e.g., segmentation and object recognition). Even for a specific job, it is difficult to find a one-size-fits-all algorithm capable of being efficient and accurate in all settings. The engineering of video sequence analysis systems, thus, requires choosing and configuring the right combination of algorithms [21].

In practice, engineering such systems is an iterative process in which algorithms are combined and tested on diverse inputs (video sequences). Practitioners can eventually determine what algorithms are likely to fail or excel in certain conditions before the actual deployment in realistic settings. Admittedly, practitioners rely on empirical and statistical methods, based on numerous metrics. However, the major barrier to improve analysis algorithms is to find a *suitable and comprehensive input set of video sequences for testing analysis algorithms*. The current testing practice is rather manual, very costly in time and resources needed, without any qualitative assurance (e.g., test coverage) of the inputs [15, 24].

In a project involving three industrial partners, we observed that collecting videos for testing such algorithms is difficult. The targeted scenarios should challenge algorithms to process video sequences by introducing high *variability*, for example, different kinds of luminosity, altitudes, instability, meteorological conditions, etc. By combining the different variation points of the video sequences, we identified 153000 possible video sequences (three minutes each), corresponding to 320 days of videos to process and sixty-four years of filming outdoors (two hours per day). The numbers were calculated at the beginning of the project and the situation is now worth with billions of possible video sequences. These values show that the current practice, based on a manual elaboration of video sequences, is too expensive in terms of time and resources needed. Moreover, a related problem is that practitioners ignore what kinds of situations are covered or not by the set of video sequences.

In this paper, we present VANE, a variability-based testing approach for synthesizing video sequence variants. The key ideas of VANE are to promote the use of a variability model

ISSTA '14, July 21-25,2014, San Jose, CA, USA

describing what can vary within a video sequence and to exploit the variability model to generate configurations. These configurations are exploited afterwards to synthesize variants of a video sequence. In this research we rely in feature models [4, 5, 20] which are the most popular notation for modeling and reasoning about variability. We use advanced constructs such as attributes for handling numerical parameters and preferences. We apply combinatorial testing [11–13, 16, 18, 19] over feature models with attributes to reduce the number of configurations (combinations of features and attributes).

VANE is a hybrid approach mixing constraint satisfaction problem (CSP) solving techniques and evolutionary algorithms. The CSP is used to obtain T-wise covering sets while the genetic algorithm is used to tackle the multi-objective nature of the problem. A unique property of VANE is that it can obtain the minimal T-wise coverage while optimizing a function over attributes, for example, to minimize a custom attribute such as the video luminance.

Previous research proposed to use different metrics to optimize test-suites for concrete users needs in variability-intensive systems [10, 12, 18, 19]. 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*. Other evolutionary-based approaches do not consider testing aspects [22].

This paper provides the following contributions:

- An original application of variability and testing techniques to the domain of video sequence analysis, in the context of an industrial project.
- CSP encoding and automated techniques to grant T-wise coverage for feature models with attributes. We also develop multi-objective solving techniques to obtain T-wise configurations sets that satisfy multiple testing criteria at the same time.
- An evaluation of VANE's performance in practical settings. We show the time required, the number of configurations generated, and the benefits of the approach in comparison to current practice.

The remainder of this paper is organized as follows. Section 2 describes further the industrial case study and the problem of testing video sequence analysis systems. Section 3 presents a variability-based testing approach and describes the VANE solution to obtain T-wise covering sets while optimizing user functions over quality attributes and functional information. Section 4 describes empirical results we gather when evaluating VANE's performance on the large-scale, realistic feature model designed by video experts. Section 4.2 discusses the variability-based testing approach in the context of the industrial project as well as threats to validity. Section 5 discusses related work while Section 6 presents concluding remarks.

2. INDUSTRIAL CASE STUDY

The acquisition, processing, and analysis of video sequences find applications in our daily life. Surveillance applications, computer-aided medical imaging systems, traffic control, and mobile robotics are some examples. Such software-intensive systems rely on signals captured by video cameras, which are then, processed through a chain of algorithms. Basic signal processing algorithms are assembled in different ways, depending on the goal of image recognition (scene interpretation, follow a specific object, etc.). The algorithms produce numerical or symbolic information helpful for humans or subject to subsequent analysis. For example, rectangles covering the zone of a specific object help tracking people or vehicles in motion.

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, the processing and analysis of video sequences are performed to detect and track, for example, survivors in a natural disaster. Eventually, and based on the information, the operations manager can achieve strategic or tactical goals in a rapid manner. Two organizations 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. Clearly, there is no one-size-fits-all solution capable of handling the diversity of scenarios and signal qualities. This poses a difficult problem for all the partners of MOTIV: which algorithms are best suited given a particular 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 conditions? How to propose innovative solutions able to handle new situations?

The fundamental and common challenge is the *testing* of algorithms. All the partners, being providers or consumers, aim to determine what algorithms are likely to fail or excel in certain conditions. Empirical and statistical methods (based on numerous metrics for assessing non-functional dimensions such as performance or reliability) have been developed and are intensively used for this purpose. Nevertheless, practitioners face severe difficulties to obtain an input test suite (i.e., a set of video sequences) large and diverse enough to test the algorithms. The current practice is indeed to find some existing videos or film video sequences outside. The effort of manually collecting videos is highly consuming in time and resources. First, high costs and complex logistics are required to film video sequences in real locations. Second, the ground truth should be elaborated for every video sequence – it is again time-consuming and also error-prone. Due to the practical difficulties, the number of collected video sequences is too low and the videos are not different enough to test algorithms. In addition, practitioners have limited control over the scenarios covered by the set of video sequences. As a result, the major challenge for testing the algorithms remains: *How to obtain a suitable and comprehensive input set of video sequences?*

3. VARIABILITY-BASED TESTING APPROACH

To overcome the previous limitations, we introduce a generative approach, based on variability modeling and testing principles. The goal of the approach is to automatically synthesize a variant of a video sequence given a configuration (i.e., a selection of desired features). Compared to the current practice, the approach aims to provide more automation, more diversification and more control when collecting input

(a) Variant #1 of video sequence

(b) Variant #2 of video sequence

(c) Variant #3 of video sequence

(d) Variant #4 of video sequence

Figure 1: Four variants of video sequences

video sequences (see also Section 4.2 for a discussion about the benefits in the MOTIV project).

For example, we synthesized four different variants of video sequences (see Figure 1). The first variant (see Figure 1a) is a video sequence with a very intense luminosity and a tank moving on. The second variant (see Figure 1b) differs from the first variant: some birds and other kinds of vehicles are included while the contrast is more intense. Variant #3 (see Figure 1c) introduces shadows to mimic passing clouds. Also, Variant #4 (see Figure 1d) is over expose, thus, some colors are hardly distinguishable. We only describe static parts of the video sequence variants but the dynamic parts is impacted as well (e.g., motion of vegetation due to the wind, appearance of occultants, vibrations of camera, or shadows). Eventually, much more variants than the four depicted in Figure 1 can be synthesized to test computer vision algorithms (e.g., an algorithm in charge of tracking vehicles) in diverse and challenging settings.

As part of the approach, *variability modeling* is used to formally characterize what can vary in a video sequence and delimit the relevant testable configurations. Because of the huge number of testable configurations, *combinatorial testing* techniques are applied to obtain the minimal T-wise coverage while optimizing attributes. An overview of the approach (in the context of the MOTIV project) is given in Figure 2. At the starting point (see the top of the figure), a variability model is elaborated and characterizes a set of configurations (see next section for more details about the so-called feature model with attributes). Testing techniques (see ①) operate over the model and are in charge of producing a relevant subset of video sequences. A transformation (see ②) has been developed to obtain configuration files that consists on variables and values. Lua code developed by one of the MOTIV partner, processes the configuration files and executes some algorithms to alter, add, remove or substitute elements in a base video sequence[1]. We obtain at the end variants of video sequences (see ③).

3.1 Variability Modeling

Feature models are the most popular notation for describing the variability of a system. This formalism is commonly used for modeling software product lines [3–5]. See Figure 4 for a sample of a feature model with attributes. Feature models, though, are not limited to product lines and can also be used for modeling configurable systems (in a broad sense). The main advantages of using feature models are their relative simplicity – domain experts, such as video analysis experts, can easily understand the formalism –, their formal semantics, and a series of efficiently automated analysis techniques [4].

We propose to use feature models to model the variability of a video sequence. Variability in our context means any characteristic of a video sequence that is subject to change[2]: the global luminosity of the video sequence, the local lumi-

[1]Lua is a widely used programming language (`http://www.lua.org/`). Details about the computer vision algorithms in charge of synthesizing video sequences' variants are out of the scope of the paper.

[2]A variation point (change) can be fixed once and for all, or subject to a dynamic adaptation at runtime. For instance, we can a fixed global luminosity value for the whole video; or we can consider that it can vary during the video execution. In our experience, the two kinds of changes are indeed possible.

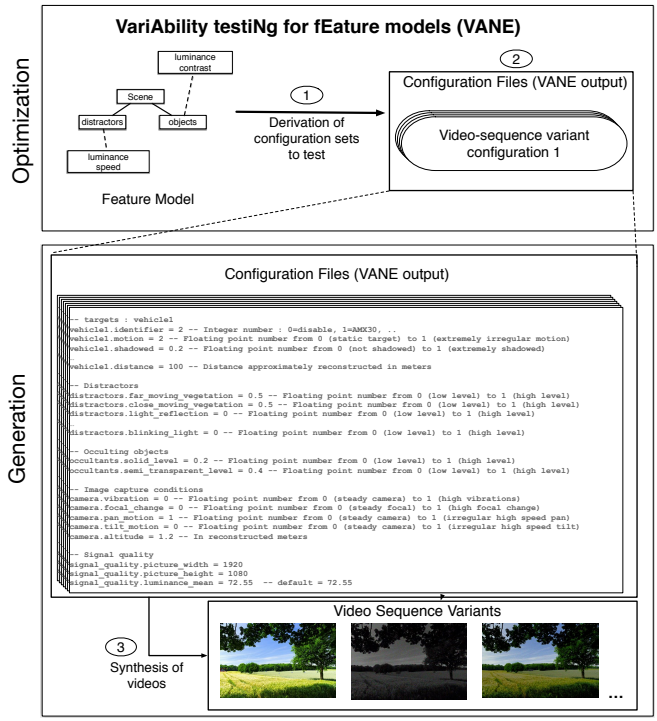

Figure 2: Realization of the variability-based testing approach in the MOTIV project

nosity of some parts of the video sequence, the presence of some noise, distractors, or vibrations, etc.

The variability of a video sequence is described in terms of mandatory, optional and exclusive features as well as propositional constraints over the features (see Figure 5). The features are hierarchically organized starting from the high-level concept (video sequence) to more refined and detailed concepts. The essence of a feature model is to characterize a set of valid configurations, where a configuration is defined as the selection of features and attributes values. Propositional constraints and variability information restrict the valid combinations of features authorized in a video sequence. For instance, only one kind of background is possible.

The domain of video analysis has some specificities.

First, *attributes* are intensively used. Attributes are needed to add non-functional information to features. They may be used to express some preferences or determine the impact of testing a concrete feature in the total testing time. Attributes also help to model numerical parameters of the video sequence. Second, some features can appear several times in the same video sequence. An example of this variability element is a Vehicle which number *dust* which can be configured independently, i.e., there can be 10 vehicles in the video sequence each having a specific *dust* value. Increasing the number of variables in the problem. With the presence of attributes, a feature model can be naturally further refined using cross-tree constraints over features and attributes. For example, we specify that the selection of a Countryside background implies that less than 10 People appear in the scene (see Figure 5).

How a variation point is realized and actually implemented is out of the scope of the paper.

Using the formalism of feature models with attributes allows video experts to have a more suited expressiveness than with only Boolean constructs. It also opens new kinds of reasoning over non-functional attributes. In practice, the challenge is now to obtain the configurations that optimize user-defined criteria over attributes. For example, in Figure 5, we encode non-functional properties such as the amount of dust generated in a sequence. In some testing scenarios, the goal could be to minimize (or maximize) the amount of dust: practitioners can define objectives function depending on their needs.

Meanwhile, a series of complex constraints involving attributes should be handled. For example, we specify two different constraints i) to ensure that if there is a high *dust* generated by the Vehicle, the background of the scene should be a Countryside one, ii) to ensure that in an urban scenario there will be a crowd of People greater than 40%.

3.2 VANE Solution

Despite the constraints over attributes and features, the number of possible configurations is enormous. Exhaustive testing in such a large space of configurations is clearly unfeasible. Literature in the past proved that most errors can be detected when using pair-wise combinations of inputs [23]. Moreover, Cohen et. al. [6] proved that those results apply to feature models. Our approach is to test configurations of video sequences that cover all possible T feature interactions (*T-wise*). In theory, T-wise dramatically reduces the number of testable video sequences while ensuring reasonable coverage.

VariAbility testiNg for fEature models (VANE) is a solution to obtain T-wise covering sets for feature models with attributes. VANE follows a set of steps to obtain T-wise covering sets of configurations while meeting different user criteria (e.g. minimize the cost of a set of configurations). Figure 3 shows the VANE process. First, developers encode the intensive variability system's variability using an attributed feature model. Second, VANE obtains the valid permutations of features to be covered. Third, VANE encodes the input model as a CSP. Later, VANE adds different constraints to the CSP depending on user requirements.

In the case that the user wants to obtain a multi-objective solution, VANE implements an "a priori" solution by using a genetic algorithm. This solution uses the previously generated CSP to find the weights that return Pareto optimal solutions.

3.2.1 T-wise CSP For Attributed Feature Models

This section describes how VANE uses CSP to derive solutions for T-wise covering arrays. Prior work in the field of automated analysis of feature models achieved to extract information from feature models by using computed-aided mechanisms. Those works yielded a set of different operations and translations into CSP problems [4].

In this paper, we consider the derivation of T-wise covering sets as an automated analysis operation that takes as input attributed feature models and user preferences. After a CSP formulation is defined for obtaining T-wise configuration sets, VANE can derive all the different valid combinations of configurations that fully covers a set of feature pairs.

CSP encoding of the problem. A CSP is a mathematical problem that is composed by a set of variables which value must satisfy a set of constraints. For example, if we

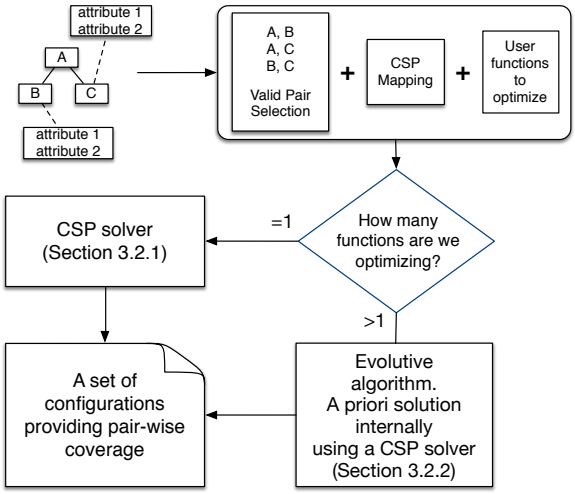

Figure 3: VANE process to obtain optimal T-wise covering sets.

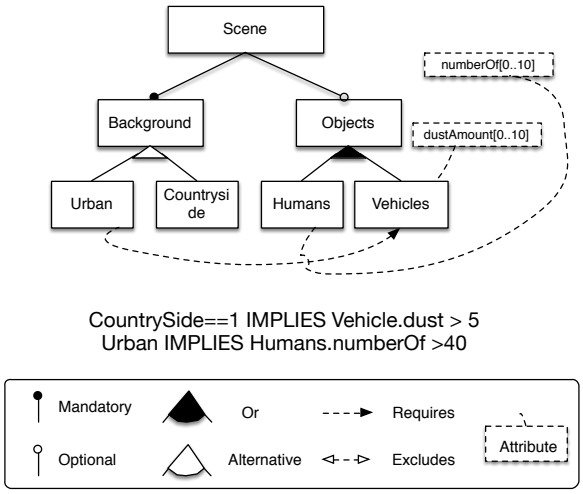

Figure 4: An exemplified feature model with attributes

want to model the problem of buying a certain number of objects that cost 5 $ which a fixed budget of 20 $, the CSP can look like $(A < 20) \wedge (A = B * 5)$ where A is an integer variable describing the total cost of the purchase, B a variable representing the number of products to buy, and the 20$ budget a constraint. A CSP solver is a software artifact that enables to retrieve all possible valid labeling in a CSP. A valid labeling is an assignation of values to variables that satisfy all constraints involved into the problem. Moreover, CSP solvers enable the optimization of different functions such as minimizing the value of an integer variable.

A feature model can be encoded as a CSP as previous research did in the past [4]. In this mapping, each feature F_i in the feature model features set F is represented by a boolean variable f_i in the CSP and each kind of constraint in the feature model is represented by a logic representation of it. For example, a *mandatory* relationship between the feature A and the feature B is mapped as $A \Leftrightarrow B$, meaning that, if the feature A is selected, then, the CSP have to select the feature B.

An additional set of constraints should be added in case of dealing with attributed feature models. An attribute is defined by a domain, a default value and a null value. First, every attribute A_{ij} for the feature F_i is represented by a variable a_{ij}, later two different constraints are added i) $A_{ij} \in domain$, to grant that the attribute value will be between the limits of its own domain; and ii) $if\ (f_i = 1)\ then\ (A_{ij} = defaultValue)\ else\ (A_{ij} = nullValue)$, to grant that if the feature F_i is selected the attribute value is its default value and the null value otherwise.

After translating all features, attributes and constraints to the CSP, more constraints have to be introduced into the CSP to grant the covering of a concrete set of features. Therefore, if we want to grant that the F_i is covered by the CSP solutions, we need to add the constraint $F_i = 1$ into the solver to grant that the feature i will be selected in all solutions. For example, if we want to cover the pair of features composed by the feature **People** and **Countryside**, then we would add the constrains $People = 1$ and $Countryside = 1$.

Deriving T-wise covering sets for attributed feature models. VANE can reason over all possible of configuration sets that cover a concrete set of T-wise combinations. For example, in the case of the model presented in Figure 4, VANE retrieves the configuration set covering all feature combinations such as Urban and Humans. To cover a set of feature combinations, VANE uses a custom mapping between feature models and CSP. The mapping used is defined by the tuple:

$$< P, F, FC, A, AC, PC >$$

where:

- ***P*** is the set of feature combinations to be covered. P_{ij} represents the feature j of the feature combination i needed to be covered by a configuration in the test-suite.
- ***F*** is a set of variables representing the features in the feature model. If the variable f_i is equal to 1, then the feature F_i is going to be present in the configuration. (e.g if the feature i is a mandatory child of the feature j, the constraint $f_j \Leftrightarrow f_i$ is in this set)
- ***FC*** is the set of constraints representing the different relationships between the model elements. This is between different features and between features and attributes.
- ***A*** is the set of variables representing the different attributes existing in the model.
- ***AC*** is the set of constraints between different attributes. For example, is the cost should be greater than 40 a constraint representing that will be added.
- ***PC*** is the set of constraints representing the constraints granting the coverage of each pair. This is, for each pair P_{ij}, the constraint $F_j = 1 \wedge F_i = 1$ is introduced in the CSP.

This mapping differs from previous approaches because it is intended to derive combinations of valid configurations (covering sets) instead of single configurations. Table 1 shows

the main differences between the previous mapping for single configuration derivation proposed by Benavides et al. [4] and the one used by VANE. Note that in the table, F^P represents the parent feature of the relation. F^{C1} to F^{CX} represent the children features of a relation where X is the *nth* child of the relation .

T-wise covering sets optimization. There is more than one solution for the problem of finding T-wise covering sets. Moreover, some covering sets may fit better concrete user preferences. For example, some users may be interested in reducing the number of configurations to run while others prefer to reduce the time to test or other quality attributes of the test-suite. VANE enables users to maximize a concrete function over features and attributes by adding constraints into the CSP.

Optimizing a function over features and attributes. VANE enables T-wise practitioners to decide which function between attributes and features fits better the user desires.

More complex constraints are also allowed when using VANE. For example, if the user wants to minimize the number of different features when generating the test suite, we need to fix the value of the function to a variable:

$$DifferentNumberOfFeatures \sum_{i=1,\ j=1}^{n\ m} f_{ij}$$

where n is the number of pairs to cover and m the number of features in the feature model.

Minimizing the number of different configurations to use in a T-wise covering set. It is interesting to use as few configurations as possible when performing T-wise coverage. This optimization is known as the minimal or optimal T-wise [14] and different constraints should be introduced in the CSP to obtain it.

First, we need to introduce a set of reified variables that represents if the configuration covering the pair i is different from the configuration covering the pair j.

$$if\ (P_i \neq P_j)\ then\ reified_i = 1$$

Later, we minimize the sum of reified variables, thus, minimizing the number of different configurations used.

$$\min \sum_{i=1}^{n} reified_i$$

For example, the pair-wise combinations existing in the model presented in Figure 4 can be covered by the configurations i) "Scene, Background, Objects, Humans, Vehicles, Urban" and; ii) "Scene, Background, Objects, Humans, CountrySide". This is a minimal set of configurations as we cannot reduce the number of configurations while covering all pair-wise combinations.

3.2.2 An "A Priori" Solution To Obtain Multi-objective Test-suites

In our context, our goal is to generate T-wise covering sets while optimizing more than one objective function. For example, users might want to minimize the value of concrete attributes while still using as less as possible configurations. Note that optimizing different functions at the same time might not yield optimal values for them separately but a good trade-off between them. This problem is commonly known as multi-objective optimization problem [7].

Our initial experiments showed that exact solutions hardly scale when having complex attributes (see section 4). Therefore, we use an *"a priori"* solution based on genetic algorithms [**?**]. In VANE approach, the "a priori" solution is based on a mix of CSPs and genetic algorithms for multi-optimization problems. The hybrid solution VANE internally uses the custom CSP-mapping presented in Section 3.2.1 to evaluate the fitness function.

When defining the multi-objective function for obtaining multi-objective T-wise covering sets, we define a new function to optimize:

$$F(x) = w_1 * F_1(x) + w_2 * F_2(x) + ... + w_k * F_k(x)$$

where k is the number of different functions to optimize and w represents the weights of each function to be determined by the genetic algorithm. Note that all functions should return values normalized between 0 and 1.

Later, we created a genetic algorithm that finds the values of the w_k values that correspond to each Pareto optimal value [7] of the function. The genetic algorithm is defined by:

Gen. A gen in the algorithm is a float representing the weight of a concrete w_k.

Individual. An individual is represented by a set of genes representing all w_k in the function to optimize. Therefore, an individual is a vector of floats representing weights for each function to optimize.

Crossover. In this problem, the crossover operator is based in getting two random genes and switch their values.

Mutation. The mutation operators increment or decrement a gen value in a random quantity. This quantity will affect the precision of the Pareto optimal found. But also, it will increase the search space.

Selection method. There are several methods to select the best individuals from each generation. However, for the sake of simplicity, only ranked based selection methods are used in this problem.

Let us consider an example in Figure 4. If we want to optimize the attribute *dustAmount* and minimize the number of different configurations for a 2-wise coverage, we should use the fitness function: $\max(w_1 * \sum_{i=1}^{n} dustAmount_i + w2 * \sum_{i=1}^{n} numberofEqualConstraints)$. This will return the Pareto solution: "Scene, Background, Objects, Humans, Vehicles, Urban".

4. EVALUATION

We aim to evaluate our variability-based testing approach along two dimensions:

Scalability of VANE. As described in Section 2, very large sets of testable configurations are possible. An implementation of VANE should be able to cope with models encoding large numbers of features and attributes. Specifically, we aim to evaluate how the approach scales when deriving T-wise configurations using the large-scale and realistic feature model of the MOTIV project. We also aim to determine the number of configurations required when optimizing different values of the feature model.

Table 1: Comparison between single derivation CSP for attributed feature models and T-wise covering set derivation CSP.

Relation	Traditional mapping	VANE mapping
Mandatory	$F^P = F^{C1}$	$\forall P_{ij}$ in P , $F^P_{ij} = F^{C1}_{ij}$
Optional	$if(F^P = 0)$ then $(F^C = 0)$	$\forall P_{ij}$ in P, $if(F^P_{ij} = 0)$ then $(F^C_{ij} = 0)$
Or	$if(F^P = 1)$ then $\sum(F^{C1}, F^{C2}, ..., F^{CX})$ in $1..X$ else $\sum(F^{C1}, F^{C2}, ..., F^{CX})$	$\forall P_{ij}$ in P, $if(F^P = 1)$ then $\sum(F^{C1}_{ij}, F^{C2}_{ij}, ..., F^{CX}_{ij})$ in $1...X$ else $\sum(F^{C1}_{ij}, F^{C2}_{ij}, ..., F^{CX}_{ij}) = 0$
Alternative	$if(F^P = 1)$ then $\sum(F^{C1}, F^{C2}, ..., F^{CX}) = 1$ else $\sum(F^{C1}, F^{C2}, ..., F^{CX}) = 0$	$\forall P_{ij}$ in P, $if(F^P_{ij} = 1)$ then $\sum(F^{C1}_{ij}, F^{C2}_{ij}, ..., F^{CX}_{ij}) = 1$ else $\sum(F^{C1}_{ij}, F^{C2}_{ij}, ..., F^{CX}_{ij}) = 0$
Excludes	$if(F^P > 0)$ then $(F^{C1} = 0)$	$\forall P_{ij}$ in P, if $(F^P_{ij} > 0)$ then $(F^{C1}_{ij} = 0)$
Requires	$if(F^P > 0)$ then $(F^{C1} = 1)$	$\forall P_{ij}$ in P, if $(F^P_{ij} > 0)$ then $(F^{C1}_{ij} = 1)$
Link between features and attributes	Not required	$\forall P_{ij}$ in P, if $(F_{ij} > 0)$ then $Attr_{ij} = value$
Complex Constraints	Not required	These constraints will be mapped depending on the constrain itself
Pair-wise constraints	Not required	$\forall P_{ij}$, $F_j = 1 \wedge F_i = 1$

Practical benefits and limits. of the approach in the context of an industrial project (see Section 2 and the MOTIV project). The introduction of variability modeling and testing techniques aims at improving current practice. We discuss qualitative properties of the approach as part of an action-based research we conduct.

4.1 Scalability of VANE

Experimental data and platform. we used the feature model of the MOTIV project to test the scalability of VANE. The feature model was elaborated during several meetings with the MOTIV partners. Their expertise in the development and quantification of (embedded) video processing systems helped us to know more about the video domain. It should be noted that each configuration of a feature model can be translated to a configuration file (through assignment of values to attributes)(see Figure 2). The design of the configuration file helps to design realistic and concrete correspondences at the implementation level.

An excerpt of the resulting feature model is shown in Figure 5. This feature model contains a large set of attributes and constraints that reduces the total number of permutations between input combinations. For example, this model prevents us of generating video-sequences having more than 50% of *dustAmount* when using an Urban background. It should be noted that in Figure 5, we mark the non-inheritable attributes with the $\sim$ mark – the rest of attributes are held by their associated feature and its children.

Our experiment have been tested using a Java implementation of VANE. This implementation is provided as part of the FAMILIAR tool [2]. Internally, this solution uses the Choco CSP solver [1]. The experiments were performed on a Intel Core i5 running at 1.5, 8 GB of RAM memory, the 1.7 Oracle java virtual machine and Os X v10.9.1 as operating system.

Hypothesis: VANE derives pair-wise sets in an affordable time when optimizing a concrete value. One of the main aspects of the VANE solution is the ability of maximizing or minimizing custom functions over attributes and features. We measured the time and the number of different configurations required to obtain a pair-wise coverage for each quality attribute in the feature model. Our hypothesis is that VANE is able to derive optimal pair-wise configuration sets for video-sequences in an affordable time. In this experiment we are only considering the maximization of one attribute per execution. That is, each time we call the CSP solver, we ask for the variable assignation maximizing a function which is the value of one attribute. We consider that the VANE solution must derive pair-wise combinations in less than 15 minutes to determine if the VANE is valid for the purposes of the MOTIV project.

Results. Figure 6 shows the time required by VANE when optimizing each attribute present in the model. In this figure, the vertical axis shows the number of seconds required to optimize the attribute in the horizontal axis. The results show that none of the optimization operations take more than 15 minutes to finish. Moreover, the time vary from the 171.64 seconds (Vehicles.distance) to 680,75 seconds (TiltMotion). The time required to generate a video-sequence manually is largely higher than 30 minutes, thus, we think that spend $5 \sim 15$ minutes is worth to reduce the number of video sequences to generate. Nevertheless, Figure 7 shows the number of configurations required to use when optimizing each attribute. The number of pairs to cover is 767, thus, by using VANE it was decreased by at least 269 configurations. This amount of configurations is obtained not by minimizing the number of configurations but only when maximizing an attribute. This is, for this experiment, we did not minimize the number of configurations to use and the testing cost reduction is very noticeable.

The experiments presented in this section show: i) the amount of time required by VANE to optimize the set of configurations to execute; and ii) the reduction in terms of the number of configurations. We conclude that the execution time of the tool is worth for the application of pair-wise testing in video-sequences generation. As previously shown,

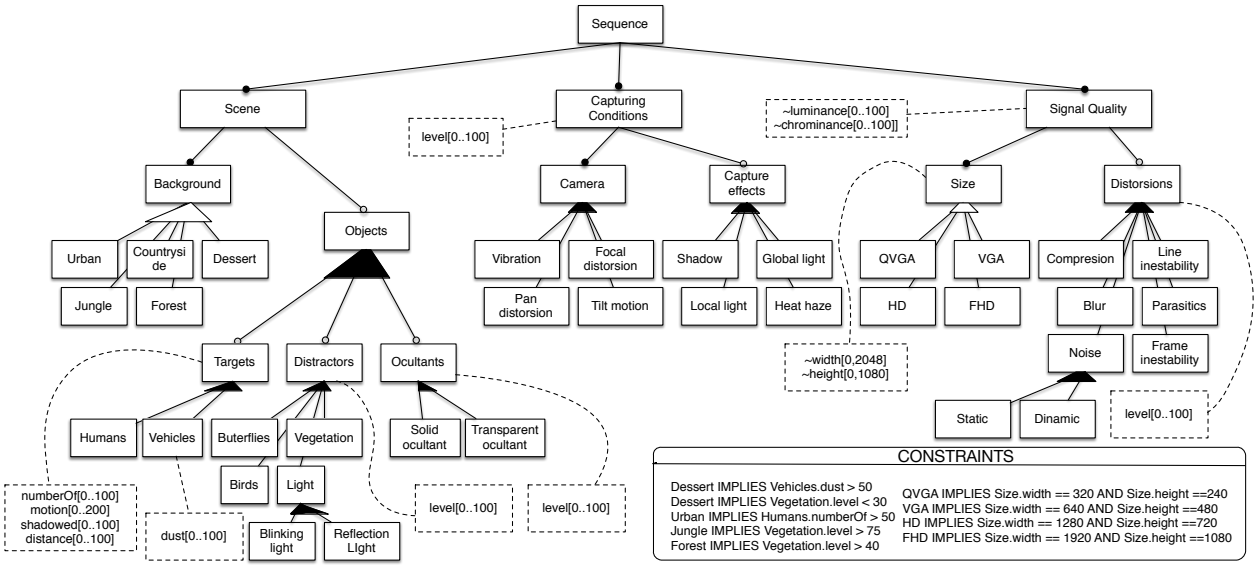

Figure 5: Feature model, excerpt of the model used in the MOTIV project, to represent variability (through features and attributes) of a video sequence.

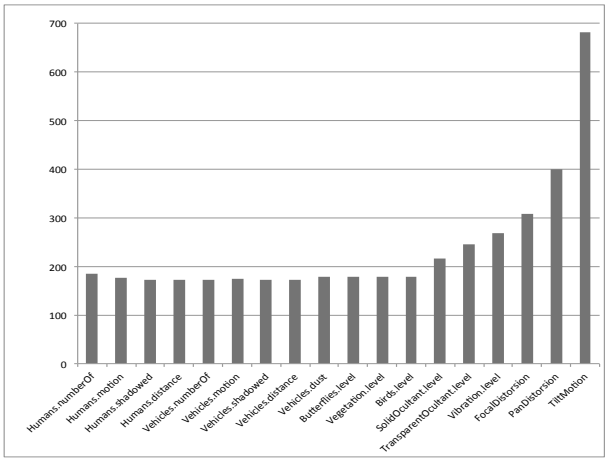

Figure 6: Required time when VANE optimizes one quality attribute.

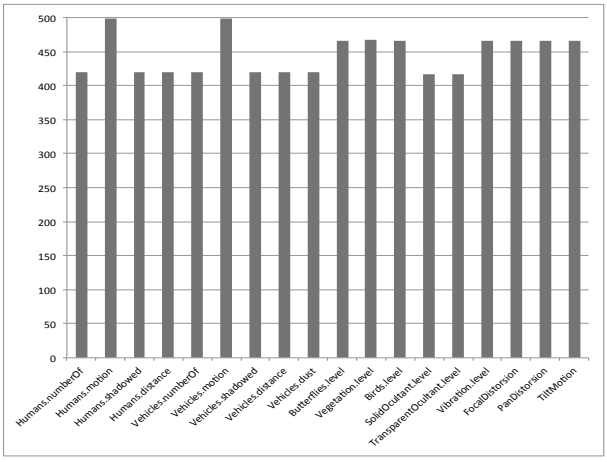

Figure 7: Required number of different configurations when VANE optimizes one quality attribute.

the reduction of costs carried out when generating video-sequences is noticeable.

Scalability and multi-objective. As part of the experiment, we observed that the complexity of *multi-optimization* problems applied for T-wise covering sets carries out high a memory and CPU consumption. For example, the time required to use multi-objective functions in our feature model can take about twelve hours (the previous experience only considers one objective function; more details about the multi-objective results can be found online). The practical impact is limited though, since experts compute once and for all the configurations; then the variants of video sequences are synthesized offline (see the next section for a discussion about other aspects of the approach). We admit that our current solution prevents an application in contexts requiring fast responses such as it happens in dynamically re-configurable systems.

4.2 Benefits and Limits of the Approach

We discuss the approach in the context of the MOTIV project. The project is still ongoing, but we already observed some practical benefits and limits; we also identified some open questions. Our report is based on discussions we had with the partners in nine formal and informal meetings (i.e., workshop sessions and visits to companies).

Automation. The feature model used in the MOTIV project encodes more than 240 different Boolean parameters among other quality attributes. Compared to a manual approach, VANE selects around 500 pairs that maximize a custom attribute. This amount of configurations cannot be handled properly by a manual approach. As stated in Section 2, preparing a test case (video sequence) requires to film or find out a realistic video, to prepare the ground truth, etc. The MOTIV partners estimate that the preparation of a test case requires at least 2 hours of work per men.

In practice, handling five hundred configurations is clearly unfeasible. Another benefit of automation is to reproduce the synthesis with other parameters for targeting more specific scenarios.

Covering. Without a proper and explicit variability modeling, practitioners ignore what test cases are actually handled and covered. The knowledge of the coverage is very important since most situations are covered, thus, practitioners are more confident in terms of robustness, performance and reliability of the algorithm. It is especially important for an institution like DGA to have such a strong coverage guarantee. It is as important for the two industrial partners to cover the maximum kinds of situations and determine if the algorithms behave accordingly. VANE grants, by construction, the validity of the T-wise configuration set while enabling multi-objective testing optimization. Another argument in favor of combinational (like T-wise) and optimization techniques is that the number of testable configurations should not be too high. Indeed the synthesis of a video sequence variant is resource consuming and can take 1.5h according to the MOTIV partners[3]. This is an additional reason that motivates the reduction of the number of tests through combinational covering.

Flexibility. The VANE approach enables practitioners to control the synthesis of T-wise covering sets. Traditionally, testing approaches (mainly manual or exploratory) rely on the expertise of the practitioners. Different alternatives can be employed for this purpose:

- Putting additional constraints and specializing the feature model for specific scenarios. For instance, a specific Background (e.g., Urban) can be set up since the application is known to be deployed in a specific military ground. In turn, the testing machinery will then consider only configurations with Urban. The benefit is that practitioners can focus on specific testing scenarios, specializing the test suite to realistic cases.

- Optimizing different objective functions over attributes: practitioners can specify the relative importance or cost of a feature, fix some parameters, etc. Again, it aims at customizing the test suite to fit realistic needs.

In all cases, VANE provides the flexibility of optimizing parameters without altering the coverage of feature combinations (because of the T-wise criterion).

Realisticness. The major threat when synthesizing video sequences is to produce unrealistic video sequence or videos that present limited interest when testing algorithms. For instance, some natural scenes may not be reliably recreated or the global luminosity eases too much the tasks of an algorithm. Until now, we have not encountered such situations: the variants of video sequences reviewed by the experts so far have been assessed as realistic and ready to test algorithms. However, the experts have not reviewed the *entire* test suite. It is still unclear how a reviewing process could look like and how we can ease that task – pointing out to experts what are the features activated in the video sequence seems an interesting option. In case of detecting an unrealistic variant, practitioners can add some constraints and exploit the *flexibility* of the approach (see above).

[3]As part of our experiment we did not measure the time needed for synthesizing video sequences – we stop at the generation of configurations.

Exploration and incremental synthesis. Another promising direction is to allow practitioners to explicitly report on unrealistic variants. Also, the testing techniques should be revised/adapted to take this information into account, i.e., so that the test suite no longer contains an unrealistic video sequence. More generally it would enable an incremental synthesis process where practitioners modify objective functions, attributes, and constraints of the feature model to improve the suitability/realisticness (if needs be) of the video sequences.

Effort and reuse. The effort needed to realize the approach mainly consists in i) elaborating a variability model, ii) selecting a "base" video sequence, and iii) developing video processing functions to modify elements (e.g., luminosity) or inject new elements (e.g., a tank). Our observation is that the major effort resides in the third step. It is unclear, at this step of the research, to determine if the effort pays off and can be reused for other "base" video sequences.

4.3 Threats to Validity

External validity. The inputs used for the experiments presented in this paper represent a realistic feature model. We consider that the feature model is realistic since numerous experts were involved in the design. Moreover, the model has proven to have an implementation counterpart (the configuration files) useful for synthesizing variants of video sequences. However, it is possible that the feature model do not reflect properly the same structure as other realistic models. The major threats to the external validity are: *population validity*, since the model used in the experiment represent only one concrete instance of the problem (there might be other models that do not mirror the model presented in Figure 5); *ecological validity*: VANE analysis were executed 10-times and we report on averages to minimize the impact of third-party threads in the time being measured.

Internal validity. The CPU capabilities required when analyzing a feature model depend on the number of features, percentage, and nature of cross-tree constraints. However, multi-optimization and pair-wise derivation add other new variables affecting the performance, such as the number of the pairs to cover. We experimented with multiple variations of quality attributes to limit the internal validity. We plan to vary objective functions as well, based on knowledge and requirements of video experts.

Construct validity. The results looks promising in terms of time required to solve problems related to our feature model. However, we might need to perform a higher scale experimentation when referring to multi-objectives configuration problems.

5. RELATED WORK

This section first reviews existing works in the domain of video sequence analysis. We then compare VANE with existing approaches for modeling, testing and reasoning about variability-intensive systems.

5.1 Video Analysis

There is a plethora of work related to the domain of computer vision (and by extension to the analysis of video sequences). Many algorithms have been designed, evaluated, benchmarked and form the basis of many crucial applications of modern society. An original idea of the paper is to

propose automated techniques to synthesize variants of video sequences with the intention of testing algorithms.

Admittedly, there are attempts and technology to produce artificial videos. For example, the use of artificial videos gives an opportunity to students to study phenomena for which real videos may not be easy or even possible to find (this usually happens in the study of kinematics using different gravities and physical values [9]). To the best of our knowledge, there is no other generative approach which is guided by a high-level specification and supported by automated techniques.

Numerous initiatives have emerged to create benchmarks (for example [15]) or evaluation methods to reduce the user intervention when collecting input data (tests) (for example [24]). Benchmarks are specific to an application domain or a kind of algorithm, and rather hard to reuse for specific domains (such as military applications). Our approach aims to ease the synthesis of such benchmarks and test suites.

5.2 Variability Modeling and Testing

T-wise, multi-objective optimization, and quality-attribute management techniques have been considered and experimented for testing or reasoning about variability-intensive systems. We develop an hybrid approach (VANE) that relies on and adapt some reasoning techniques so that we can they can performed over feature models with attributes and multi-features.

Combinatorial testing aims at reducing testing costs when dealing with large and complex systems with many input combinations of inputs to test. The key insight underlying combinatorial testing is that not every parameter contributes to every fault and many faults are caused by interactions between a relatively small number of parameters. Different approaches have been proposed to help in this task being T-wise techniques one of the most accepted by researchers. VANE proposes the use of combinatorial testing by returning pair-wise coverage configuration sets.

Researchers proposed *combinatorial pair-wise testing* to reduce the number of products to test in a product line. Pair-wise techniques are a subset of combinatorial testing techniques that aim at minimizing the number of configurations to test while covering each pair of features. CSP [8] and algorithmic [12] approaches have been proposed to obtain pair-wise covering configuration sets. However, none of the existing approaches considers quality attributes as a target to maximize when testing. VANE improves previous approaches by taking into account quality attributes while returning pair-wise covering configuration sets. Moreover, VANE is able to deal with multi-objective functions that can minimize the number of different products at the same time that maximize a concrete function.

Quality attribute management when testing. Feature models can be attributed with non-functional information such as price, cost, and time to deploy. Johansen et al. [11] proposed to use different weights depending on the importance of each software product line, in that way, they give more importance to some features than to others. VANE improves their solution by enabling maximization of different model properties such as the number of different features involved in a configuration, or maximizing concrete quality attributes.

When referring to *multi-objective optimization*, different approaches can be followed to find Pareto frontiers. There are a priori solutions and a whole family of genetic algorithms that deals with such complex objective [7]. Moreover, there are some authors that pointed out the need to use multi-objective problems when testing software product lines [10]. Sayyad et al. presented evolutionary heuristics to find sound and optimum configurations of very large variability models in the presence of competing objectives [22]. In this paper, we proposed the use of a hybrid approach that uses CSP as an exact solution to retrieve T-wise configuration sets and an "a priori" approximate solution based on a genetic algorithm to optimize multi objectives over feature attributes.

6. CONCLUDING REMARKS

We cannot test everything. The domain of video analysis is not an exception to the rule: testing the computer vision algorithms under all combinations of inputs (video sequences) is not feasible. The manual elaboration of a test suite of video sequences is a possible solution, but as reported in an industrial project, the process is very costly in time and resources. As a result, practitioners face severe difficulties to collect video sequences able of covering the diversity of targeted video analysis scenarios.

In this paper, we have described an original approach combining variability and testing techniques. A formal variability model (i.e., feature model with attributes and multi-features) documents what can vary within a video sequence. Combinational and multi-objective optimization techniques have been presented to generate a certain number of configurations. The configurations are exploited afterwards to synthesize variants of video sequences.

Specifically, our solution called VANE computes T-wise combinations of the parameters while maximizing a custom function over quality-attributes. The VANE implementation and experiments described in this paper are available in open source form as part of the FAMILIAR tool. Moreover, we provide the experiments inputs and the results `https://github.com/ViViD-DiverSE/Experiments`.

As reported in the context of an industrial project, the approach has the potential to provide (w.r.t to the current practice) more automation, flexibility while enabling more diversity when synthesizing a test suite of video sequences. The next step of our research is to further assess the practical benefits for domain experts and to investigate the applicability of the approach for the rich domain of video analysis. We are also studying how to support an interactive process in which practitioners can report on or avoid some unrealistic (if any) variants of video sequences, with the ultimate goal of getting a comprehensive and suitable test suite.

7. ACKNOWLEDGEMENTS

We thank all the industrial partners of the MOTIV project, supported by the French DGA. This work has been partially supported by the European Commission (FEDER), the Spanish Government under project TAPAS (TIN2012-32273) project and the Andalusian Government under projects THEOS (TIC-5906) and COPAS (P12-TIC-1867).

8. REFERENCES

[1] `http://www.emn.fr/z-info/choco-solver/`. Acceced: June of 2014.

[2] 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, page 22, 2013.

[3] S. Apel, D. Batory, C. Kästner, and G. Saake. *Feature-Oriented Software Product Lines*. Springer, 2013.

[4] D. Benavides, S. Segura, and A. Ruiz-Cortés. Automated analysis of feature models 20 years later: a literature review. *Information Systems*, 35(6), 2010.

[5] Berger, Thorsten and Rublack, Ralf and Nair, Divya and Atlee, Joanne M. and Becker, Martin and Czarnecki, Krzysztof and Wasowski, Andrzej. A survey of variability modeling in industrial practice. In *VaMoS'13*. ACM, 2013.

[6] M. B. Cohen, M. B. Dwyer, and J. Shi. Coverage and adequacy in software product line testing. In *Proceedings of the ISSTA 2006 workshop on Role of software architecture for testing and analysis*, pages 53–63. ACM, 2006.

[7] K. Deb. Multi-objective optimization. *Multi-objective optimization using evolutionary algorithms*, pages 13–46, 2001.

[8] T. Dohi and B. Cukic, editors. *IEEE 22nd International Symposium on Software Reliability Engineering, ISSRE 2011, Hiroshima, Japan, November 29 - December 2, 2011*. IEEE, 2011.

[9] M. R. Gallis. Artificial Video for Video Analysis. *The Physics Teacher*, 48:32–34, Jan. 2010.

[10] 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*, SPLC '13, pages 62–71, New York, NY, USA, 2013. ACM.

[11] M. F. Johansen, O. Haugen, F. Fleurey, A. G. Eldegard, and T. Syversen. Generating better partial covering arrays by modeling weights on sub-product lines. In *Proceedings of the 15th International Conference on Model Driven Engineering Languages and Systems*, MODELS'12, pages 269–284, Berlin, Heidelberg, 2012. Springer-Verlag.

[12] 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*, page 46, 2012.

[13] B. P. Lamancha and M. P. Usaola. Testing product generation in software product lines using pairwise for features coverage. In *Proceedings of the 22Nd IFIP WG 6.1 International Conference on Testing Software and Systems*, ICTSS'10, pages 111–125, Berlin, Heidelberg, 2010. Springer-Verlag.

[14] Y. Lei and K.-C. Tai. In-parameter-order: A test generation strategy for pairwise testing. In *High-Assurance Systems Engineering Symposium, 1998. Proceedings. Third IEEE International*, pages 254–261. IEEE, 1998.

[15] S. Oh, A. Hoogs, A. Perera, N. Cuntoor, C.-C. Chen, J. T. Lee, S. Mukherjee, J. K. Aggarwal, H. Lee, L. Davis, E. Swears, X. Wang, Q. Ji, K. Reddy, M. Shah, C. Vondrick, H. Pirsiavash, D. Ramanan, J. Yuen, A. Torralba, B. Song, A. Fong, A. Roy-Chowdhury, and M. Desai. A large-scale benchmark dataset for event recognition in surveillance video. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 3153–3160, Washington, DC, USA, 2011. IEEE Computer Society.

[16] S. Oster, F. Markert, and P. Ritter. Automated incremental pairwise testing of software product lines. In J. Bosch and J. Lee, editors, *SPLC*, volume 6287 of *Lecture Notes in Computer Science*, pages 196–210. Springer, 2010.

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

[18] 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*, pages 605–643, Aug. 2011.

[19] G. Perrouin, S. Sen, J. Klein, B. Baudry, and Y. L. Traon. Automated and Scalable T-wise Test Case Generation Strategies for Software Product Lines. *2010 Third International Conference on Software Testing, Verification and Validation*, pages 459–468, 2010.

[20] K. Pohl, G. Böckle, and F. J. van der Linden. *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer-Verlag, 2005.

[21] 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*, 16:11, 2011.

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

[23] K.-C. Tai and Y. Lei. A test generation strategy for pairwise testing. *Software Engineering, IEEE Transactions on*, 28(1):109–111, 2002.

[24] H. Zhang, J. E. Fritts, and S. A. Goldman. Image segmentation evaluation: A survey of unsupervised methods. *Computer Vision and Image Understanding*, 110(2):260–280, 2008.