International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

HYBRID TEMPLATE ANALYSING FOR AN EFFICIENT AND PRECISE POLYHEDRAL ANALYSIS

Yassamine Seladji
University Abou Bekr Belkaid, Tlemcen
yassamine.seladji@univ-tlemcen.dz

Abstract

This paper proposes a new static analysis technique that combines two complemen- tary approaches: Principal Component Analysis (PCA)-guided template generation and dynamic threshold-based Smart Widening. Building on recent advances in weakly-relational polyhedral analysis and efficient convergence strategies, we introduce a hybrid method that achieves a better trade-off between precision and performance. Our technique leverages PCA to identify the most relevant invariant directions and dynamically adapts widening thresholds based on intermediate abstract states. Empirical evaluation demonstrates significant improvements over existing techniques in both accuracy and scalability.

The specified framework is not only more efficient regarding computing but also adds another layer of automation to the process of finding invariants. The method does not use manually defined thresholds to guarantee adaptative behavior regardless of the program structure and numerical complexity. Moreover, the direction choice of the PCA will give a better understanding of the relationship between the variables of the program, and allow making more significant abstractions, and fewer false alarm on the way to verification. This flexibility causes the approach to be ideal in studying real-time systems of control, reactive programs and embedded applications. In general, the hybrid PCA-smart widening framework is a data-based improvement of the traditional data non-dynamic techniques of analysis.

Keywords: Static analysis, Abstract interpretation, Polyhedral domain, Principal Component Analysis (PCA), Smart widening, Software verification.

I. INTRODUCTION

In a world more and more numerical, software engineering plays an important role. It's at the heart of all domains, medical, education, communication, aeronautics etc. The needs of these domains in software are more and more complex, and software are used in more and more critical tasks as in defense systems, on planes or also for medical equipments. This important place given to the software engineering requires a very high level of safety, it means no bugs, because, each small bug can have dramatical consequences. Let us take for example the crash of the Ariane 5 launcher caused by an overflow bug, or also the Patriot missile failure which resulted in 28 persons being accidentally killed, it was the consequence of a cumulated small rounding error. However, the detection of bugs in long and complex software is complicated [1]. To deal with that, many software verification methods have been developed.

In the field of software verification, we distinguish two fundamental approaches:

- Dynamic verification: it is more generally known as test phase. The basic idea is that, the software behaviors are checked during their execution, and the bad ones can be observed and deleted. The tests should cover all the possible execution cases to be able to find all possible bugs. However, the difficulty to develop such tests increases with the software complexity. Furthermore, this method is performed during software execution, which can be late in some

International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

cases, like for example for embedded systems. So, this kind of verification method is not appropriate to analyze complex and critical systems.

• Static analysis: this verification is done without executing software. The principle is the use of formal methods to represent mathematically the programs to verify and thus exploit the power of mathematical methods to guarantee a full coverage of program behaviors, and it is done in an automatic way. This discipline knows a growing interest from the industrial world. It comes in several sub-domains, such as model checking, data flow analysis and abstract interpretation.

In this work, we restrict our interest to static analysis by abstract interpretation [2]. The main advantage of using abstract interpretation lies in the fact that it allows designing sound static analyzer based on the notion of abstract domains.

II. THEORETICAL ASPECTS

A. Static Analysis by Abstract Interpretation

Static program analysis aims to automatically determine whether a program satisfies specific properties, such as "the program never dereferences a null pointer," "the program never divides by zero," and "user-specified assertions are never violated." Abstract interpretation provides the mathematical foundation for designing such analyses. Developed by Patrick Cousot and Radhia Cousot in the late 1970s, it offers a unifying framework for static program analysis by approximating the behavior of programs. Abstract interpretation involves replacing a precise element of a program with a less detailed abstraction in order to infer the program's properties, this abstraction is called abstract domain. In abstract interpretation: "interpretation" means that the program to analyze is defined by its semantics, which allows a mathematical representation of program behaviors [3]. However, the representation of all program behaviors using its semantic is undecidable with finite time and memory. Hence, "abstraction" means the over-approximation of the program behaviors using the concept of abstract domains. This latter abstracts the set of program (concrete) behaviors by a sound over-approximation which can be easily manipulated and stored in the computer memory. Taking into account this new representation, the program (concrete) semantic is replaced by its abstract version, which is able to manipulate these new abstract elements. These changes make the computation of program behaviors decidable. We will look more closely at these notions a little farther.

In this work, we focus on the verification of numerical safety properties. It consists in reasoning about the set of values that the numerical variables of a program can reach during its execution, taking into account all possible program inputs, this is known as the invariant of the program. The aim is to be able to say at the end of the analysis " that nothing bad, caused by the numerical variables, will happen", like for example "No division by zero" or also "No overflow". Unfortunately, due to the approximation, such answers are not always possible be- cause the approximation can possibly add some false alarms. Therefore, in the case where the analysis returns bugs, it is impossible to know if these bugs are false alarms or not. So, the analysis answer will be "We don't know" [4]. The important point to note here is that the quality of the analysis depends on the quality of the chosen abstract domain. However, the quality of an abstract domain is defined by two parameters: the amount of approximation added to the corresponding concrete element and the efficiency of its operators. In other words, this quality is represented by the accuracy of the computation and its execution time.

Let us look more closely at this theory. For that, let S be the set of values that a program variable can reach during the execution. In most cases, the cardinality of S explodes, which makes its manipulation and its storage in the computer memory hard or impossible. The solution given

International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

by abstract interpretation is to over-approximate S using numerical abstract domains. The aim is to use a compact representation of S to easily manipulate and store it. To exemplify this idea, let S] be the representation of S using intervals, such that: S] = [min S, max S]. We call S] the abstract version of S in the interval abstract domain.

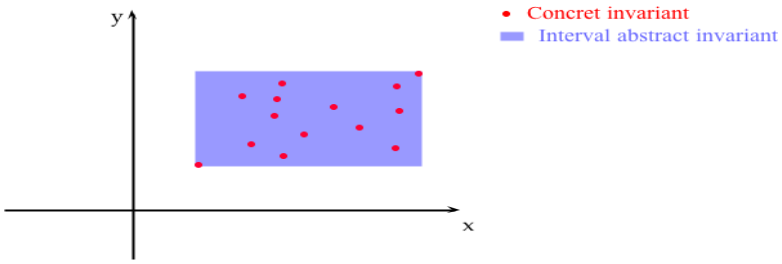


Fig 1: The interval representation, given by the blue box, of the concrete values, given by red points

This transformation obliges us to adapt the program semantics, in the order to be able to manipulate intervals. For that, the standard operations on intervals are used. Indeed, S] has a compact representation, however it over-approximates S ie. S] contains values which are not in S.

A lot of efforts have been made to reduce the effect of the over-approximation by defining new numerical abstract domains, which express relations between variables of the program. Indeed, this makes them more interesting than intervals. Most of these domains represent the program invariants by a convex set defined as a conjunction of linear constraints. Let us take as an example the octagon domain [?], where invariants are conjunctions of constraints of the form $\pm x \pm y \leq c$, such that x and y are program variables and c a constant in Z, Q or R.

This gain of precision can be improved by increasing the expressiveness of the abstract domain. For that, abstract domains should express more linear relations between variables of the program. This is the idea underlying the polyhedron abstract domain. In fact, in this case the invariant is a polyhedron defined by a conjunction of constraints of the form $\Sigma n\, \beta i xi \leq c$, where $\forall i \leq n$, xi is a variable of the program and βi, c are constant in Z, Q or R. On the one hand, the polyhedra abstract domain returns invariants which express a large set of properties. On the other hand, this expressiveness makes the analysis more complex and expensive. Indeed, for programs with a large number of variables or with loops, the execution time is, in the worst cases, exponential in the number of the variables of the program.

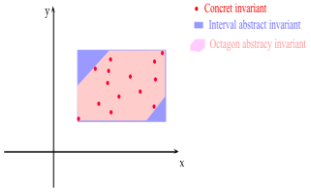


Figure 2: The octagon given in light red is the octagonal representation of the concrete elements given by red points. Note that this representation is more accurate than the interval one (blue box).

Polyhedrical analysis is one of the most expressive numerical abstract domains of a static analysis: it is a geometric domain modeling program invariants with the use of convex polyhedra, which are geometric entities defined by linear inequalities of the form $a_1 x_1+a_2 x_2+\cdots+a_n x_n \leq c$. This enables the domain to express the linear relationships between program variables (e.g., correlations, bounds, and constraints) that cannot be represented by simpler domains (e.g., intervals or octagons). Polyhedral analysis has been shown to be very

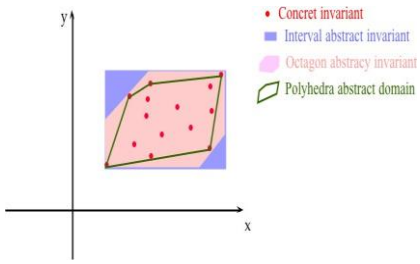
International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

precise but has major scalability and computational constraints. Optimization operations like intersection, projection and inclusion testing are expensive to compute with the number of constraints growing exponentially with the number of program variables [13]. Also, in iterative fixpoint computation, the widening operator, which is necessary to specify termination, may introduce significant over-approximation, giving imprecise invariants and false alarms.

Figure 3: The polyhedron in green represents the polyhedral representation of the concrete red point. This representation is the most accurate abstract one.

Accuracy, efficiency and convergence, then, is a problem which continues to be a challenge in the analysis of polygons. To solve these problems, adaptive algorithms like dynamic thresholding, delayed widening and more recently, PCA-based template selection have been proposed by researchers. These methods are intended to maintain important constraints and minimize unneeded computational loads. Using data-driven algorithms such as PCA allows one to select meaningful invariant directions and thus enhance scalability at the expense of analytical quality [14].

Using any abstract domain, the analysis is not guaranteed to terminate. So, to tune the cost/precision compromise, the main idea is to discard some information to accelerate the analysis and enforce its termination. In the case of programs with loops, the invariant



computation can take an infinite or very large execution time. For that, the widening operator is used. It throws out the constraints that are not stable during the execution, that forces the termination by over- approximating the invariant. More formally, a widening operator ∇ is a two-argument function which tries to predict the limit of the iterates based on the relative position of two consecutive iterates. For example, when the upper bound of the iterates increases, it is replaced with $+\infty$ (while decreasing lower bounds are replaced with $-\infty$). The comparison's criterion between two iterates depend on the abstract domain used for the analysis. A widening operator often makes large over-approximation because it must make the sequence of iterates converge in a finite time. This over-approximation may be reduced afterwards using a narrowing operator but the precision of the final approximation still strongly depends on the precision of the widening operator ∇.

Various techniques have been proposed to improve the widening, as for example the delayed widening, which applies ∇ operator after n iteration steps only (where n is a user-defined integer). The widening with thresholds [?] is another technique, where a user defines statically a set of thresholds. These thresholds are successively used like a candidate of the fixed point. However, they suffer from their lack of automatization, as thresholds must be chosen a priori and are defined by the user. Some methods try to automatically discover thresholds from the syntax of the program [?, ?]: whenever an inequality (e.g. the condition of a loop) is found, a threshold (or a landmark in [?]) is added, its value depending on the constants appearing in the inequality. So the thresholds are based on a syntactic criterion.

International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

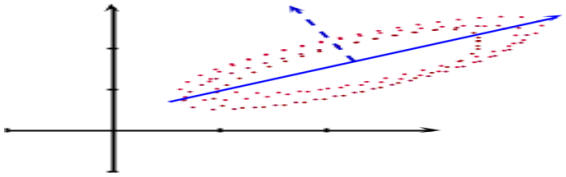


Figure 4: The principal component (blue axis) for a given set of points (red points)

Abstract interpretation offers the mathematical structure of reasoning about program behavior using sound over-approximations of semantics [8]. The concrete domain, denoted by Srepresenting the set of all possible states a program may reach as it runs, and the abstract domain, denoted by S^#. This interaction between the two domains is formally characterized by a Galois connection (α,γ)in which α:S→S^#is the abstraction function, taking concrete elements into their abstract counterparts, and γ:S^#→S is the concretization function, taking the the abstract elements into their concrete counterparts.

The basic aim is to calculate the least fixpoint (lfp) of semantic function F, denoted "lfp"(F) which is the set of all reachable states. Computing "lfp"(F)precisely, however, is in many cases undecidable because state spaces are infinite, as are recursive control structures [9]. To overcome this, abstract interpretation works out the fixpoint of an abstract semantic function F^#, which yields an over- approximation "lfp"(F^#). To ensure that the termination is achieved, the framework uses an operator of widening ∇ that speeds up the convergence and an operator of narrowing Δ that further refines the output to regain analytical accuracy.

To summarize, the efficiency of a static analyzer based on abstract interpretation depends, essentially, on the chosen abstract domain and the corresponding widening strategies. The bottleneck is the fact that the expressiveness of an abstract domain makes the analysis expensive, hence the use of the widening. However, the good trade-off between precision and efficiency is important in such process. In this thesis, a smart compromise will be proposed between the abstract domain expressiveness and the analysis cost, in the aim to improve the analyzer results.

B. Principal Component Analysis

The Principal Component Analysis (PCA) [?, ?, ?] is a statistical tool. It is widely used to extract relevant informations from a complex data set by reducing its dimensionality.

For example, PCA is commonly used for analyzing data, which are constructed by considering multiple measurements (age, salary, children, . . .) from a sample of population. The number of measurement types is the dimension of data. Let n ∈ N be the number of the considered measurement types, they can be represented as a set of points in an n-dimensional space using the orthonormal basis. PCA computes a new orthogonal coordinate system that better expresses these points. What does best express the points mean? It means that the axis where the points (variables) are most spread out when projected onto it, ie: the new axis maximize the variance of the projection of the initial points [11]. This is illustrated in Figure 4, where the axis in blue represent the principal components of the red set of points.

The PCA uses the notion of covariance, which is useful to find out how much the dimensions vary from the mean with respect to each other. The covariance is always measured between 2 dimensions. Its definition is given as follows:

Let X and Y be two measurements and ∀i ∈ N let xi and yi be, respectively, the corresponding data. The covariance formulas between X and Y, noted cov(X, Y), is given as follow:

International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

$$cov(X,Y) = \frac{\sum_{i=1}^{n}(x_i - \bar{X})(y_i - \bar{Y})}{(n-1)}$$

with, $\bar{X}$ and $\bar{Y}$ represent, respectively, the mean of X and Y.

with, $\bar{X}$ and $\bar{Y}$ represent, respectively, the mean of X and Y .

In the case where we have more than two dimensions, we need to compute the covariance between each pair of measurements (dimensions) [10]. In fact, for an n-dimensional data set, we

should calculate n! different covariance values. These values are putted in a covariance

matrix, noted C. C is an n × n matrix, where is the dimensions of the data set, such that,

∀i, j ∈ N , Cij = cov(X, Y) with X is the dimension of the ith row and Y is the dimension of the

jth column. To illustrate this idea, let us take an example of covariance matrix for an imaginary 3-dimensional data set X, Y and Z. The corresponding covariance matrix is calculated as follow:

$$C = \begin{pmatrix} cov(X,X) & cov(X,Y) & cov(X,Z) \\ cov(Y,X) & cov(Y,Y) & cov(Y,Z) \\ cov(Z,X) & cov(Z,Y) & cov(Z,Z) \end{pmatrix}$$

We note that, below the main diagonal, the covariance value is between one of the dimensions and itself, which represents the variance of this dimension [12]. We have also that cov(X, Y) = cov(Y, X), so the matrix is symmetrical about the main diagonal.

In the next step, we need to compute the eigenvectors and eigenvalues of the obtained covariance matrix. Note that the covariance matrix is square, so their eigenvectors and eigenvalues exist and can be computed by using for example the Singular Value Decomposition (SVD) [?]. These are rather important, as they tell us useful information about our data set. The obtained eigenvectors allow us to extract lines that characterise data. So, the eigenvector with the highest eigenvalue is the principle component of the data set [15]. It represents the most significant relationship between the data dimensions. So, using the eigenvalues, the eigenvectors can be ordered from highest to lowest, which represents the components in order of significant. Note that, the lengths of the eigenvectors should be 1. In the set of the obtained eigenvectors, we have for each eigenvector its perpendicular (orthogonal) one. In Figure 4, the blue axe represents the principal component obtained by applying PCA on the red points, where the dashed blue axis is its perpendicular.

The PCA is represented by the set of all the obtained eigenvectors. Afterwards, data can be transformed using these new patterns (axis), where the patterns are the axis that most closely describe the relationships between data [16]. This is helpful because we have now classified our data points as a combination of the contribution from each of those axis.

International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

III. CONTRIBUTION

A. Workflow Overview

In our method, we integrate the Principal Component Analysis (PCA) into the process of widening strategy. Figure 5 illustrates the workflows of the process. Steps are given as follows :

1. Pre-analysis phase: In this phase, we randomly generate a set of directions. This set is then used to perform a preliminary analysis with the aim of generating a set of points representing the shape and spread of the reachable state space.
2. PCA template generation: Based on the obtained points, PCA is used to compute the most relevant directions, which represent the eigenvectors with the highest variance [17]. These directions constitute the system's principal behavioural axes.
3. PCA Threshold Computation: The set of directions obtained using principal component analysis (PCA) is then used to compute a dynamics threshold for widening.
4. Widening with PCA Templates: widening is applied using the dynamique threshold to refine the program invariant.
5. Fixpoint computation: if the analysis detects a fixpoint, the process terminates; otherwise, the updated state is fed back into the process.

B. The improved kleene iteration

Principal Component Analysis (PCA) is an influential multivariate statistical method that aims to factor a set of correlated variables into a smaller group of non-correlated variables, called principal components, that capture the majority of the original data variance. Its main goal is dimensionality reduction: to preserve the key structure of complicated data with the minimum loss of information [18]. Practically, in the framework of the study of static programs, PCA represents a new, data-driven way to compute invariant directions, i.e., geometric orientations that describe the most important patterns of behavior in the state space of the program. These directions are associated with relationships among program variables that either stay constant or change in a predictable way in program execution traces.

In the pre-analysis stage, program states or intermediate abstract representations are collected after several executions of the analysis and assembled into a data matrix X∈R^(m×n), with each row corresponding to a program state snapshot and each column (column) corresponding to a program variable [19]. PCA starts with centering of this data i.e. subtracting the mean vector $X^{\perp}$ with each observation to remove the translation bias. The second step is to calculate covariance matrix:

$$C = \frac{1}{m-1}(X - \bar{X})^T(X - \bar{X}),$$

which captures how pairs of program variables co-vary. PCA then performs eigen-decomposition on this matrix:

$$C = Q\Lambda Q^T,$$

where Qis a matrix of eigenvectors (principal axes) and Λis a diagonal matrix of eigenvalues that is the variance represented along each axis. The eigenvectors that are associated with the largest eigenvalues are the directions of the greatest variation- they are said to be the most helpful in modeling the invariant behaviors in the program.

International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

In the context of static analysis, these dominant eigenvectors constitute automatically constructed templates that act as constraint directions of the abstract domain [20]. PCA-based templates are dynamic and not predetermined templates like predefined templates, which are defined based on the characteristics of the program that are discovered during pre-analysis. This flexibility allows the widening process to focus on significant dimensions of state-space evolution, hence improving the speed of convergence and analytical accuracy.

Computational cost of PCA is mainly due to the Eigen-decomposition step whose complexity is O(n^3) in the case of nprogram variables. Nevertheless, the number of dimensions is usually moderate in the case of static analysis, so this overhead is feasible. Furthermore, computation can be optimized using efficient numerical techniques like Singular Value Decomposition (SVD). In general, the addition of PCA in the context of static analysis brings a strong, mathematically-based mechanism to find and utilize invariant structures in programs, and eventually enhances accuracy, scalability and automation.

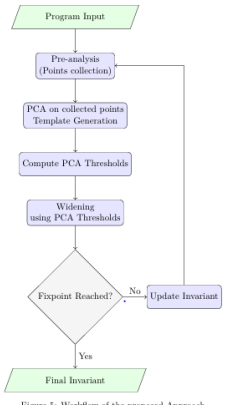


Figure 5: Workflow of the proposed Approach

The presented method fully integrates Principal Component Analysis (PCA) directly into the widening process. Unlike prior work that treats PCA only as a dimensionality reduction technique, we use PCA to define dynamique direction set and generate thresholds based on the obtained directions. This combination ensures that both the analysis direction and termination strategy are informed by empirical program behavior. This approach follows four-phases:

1. The collection of points that represent traces execution states of the program to be anal- ysed.
2. The computation of relevant template using PCA .
3. The computation of the threshold based on the PCA template.
4. The integration of the obtained threshold to improve the kleene iteration.

1) Dynamic Points Collection

In the first step, we compute a relevant set of points on which we should apply the PCA. In our method, this set of data is represented by a set of points computed in a pre-analysis. It is crucial to choose this set correctly because the spread of these points is taken into account in the PCA computation. Therefore, to obtain a relevant set of directions, the chosen set of points must provide information on the evolution of the invariant during the analysis.

The idea is to perform the standard analysis using a chosen abstract domain. Then, we collect points representing the partial traces of the programs obtained in each Kleene iteration. This set of points can help us to explore the dynamics of program behaviour and extract information that will help us to identify interesting trends.

2) PCA template Computation

International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

We propose a dynamic generation method to define a set of directions. This step is inspired by the methods presented in [?, 3]. This method uses the principe of principal component analysis (PCA) to define a set of directions based on the eigen-decomposition of the covariance matrix representing the set of points collected during an initial pre-analysis. Note that, the obtained directions set allows to catch the relevancy of the final invaraint shap using during the pre-analysis.

The set of points is used as the initial data set on which the PCA is applied. PCA is performed on the entire set of points. This computes a set of n principal components, where n represents the number of program variables. The obtained components represent the principal components and their orthogonal counterparts. These principal components are considered to be the most relevant directions. This is because they represent the most significant relationships between the points. The obtained principal components can be used in their entirety or just a subset of the most significant ones. Then, the chosen components and their orthogonal ones are considered as relevant directions and added to the set of relevant directions. We added also for each component its opposite. The obtained set of directions is considered a relevant template for computing a threshold element using during the widening loop.

To explain how the proposed hybrid PCA-guided widening technique works, a simple linear control system having two state variables, x_1and x_2, can be considered where the recurrence relationships are:

$$x_1' = 0.8x_1 + 0.2x_2 + 1, x_2' = 0.3x_1 + 0.7x_2$$

The aim is to calculate the invariant region which is all reachable states. The optimistic polyhedral analysis typically uses conventional widening at the end of every Kleene iteration, which frequently results in imprecise over-approximations (e.g. the replacement of finite bounds by ±∞ [21]. By comparison, in the PCA-based method, we start with a pre-analysis which gathers intermediate abstract states to create data matrix X which describes the numerical evolution of the system. PCA applied to Xyields two main components- eigen vectors which identify the leading directions of invariance of the system. As an example, the v1 eigenvector, v_1=[0.9,0.4]^T, can be the principal growth direction and v_2=[-0.4,0.9]^Tcan be the orthogonal contraction.

Iteration	**Eigenvector v_1**	**Eigenvalue**	**Interpretation**
1	[0.82, 0.56]	0.74	Initial expansion direction
2	[0.89, 0.46]	0.85	Stabilizing dynamics
3	[0.91, 0.41]	0.91	Converged invariant axis

Table 1: Eigenvector Evolution across Iterations

The changing eigenvectors constitute the PCA template, which directs the dynamic threshold calculation in the widening process. Only these dominant directions are then adjusted by the widening operator, eliminating uncontrolled expansion in irrelevant axes.

The thresholds are modified each time to fit the projections of PCA, and convergence on accurate invariants is achieved according of over-approximation is primarily reduced [22]. The hybrid process provides a trade-off between analytical accuracy, scalability and computational efficiency which can be effectively applied to the real control and embedded systems.

3) PCA Threshold Computation

International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

In static analysis, Kleene iteration is based on a widening operator. This operator is traditionally based on fixed or user-defined thresholds. Our method proposes an improved version that computes thresholds using the results of PCA projections on a predefined set of points. This technique ensures that the widening thresholds are geometrically close to the system's invariants, helping to reduce the over-approximation introduced during the widening process.

In this step, we use the template analysis method presented in, this analysis take into consideration the PCA template obtained in the previous step to perform the analysis. The obtained result in afterward projected on the chosen abstract domain to obtain the threshold compatible to be used with widening in the kleene iteration. Note that the computation of the threshold depends on the abstract domain used for the analysis.

4) PCA widening operator

The threshold obtained earlier is integrated into the widening operator used in the Kleene iteration. The threshold is considered as a Kleene fixed point candidate and used as such in the Kleene iteration algorithm. The modified widening operator is employed within the Kleene iteration algorithm to iteratively refine the analysis results towards a fixed-point solution. At each iteration, the obtained PCA threshold is used as a threshold to guide the widening process, ensuring efficient and accurate convergence of the analysis.

This method is given in Algorithm 1. We can see the adapted version of Kleene iteration method using the widening operator with thresholds. Operations $\nabla^{\cdot}$ PCA and $\pm^{\cdot}$ are understood as a component-wise application of operations ∇P CA and $\pm$ respectively. Note that, $X\rightarrow$ represents the used abstract domain and F represents the semantic function of the program to be analysed.

Algorithm 1 Kleene iteration algorithm based on widening operator with PCA thresholds.

1: $\vec{X}0 := \vec{\perp}$

2: **repeat**

3: $\vec{X}i := \vec{X}i-1 \dot{\nabla} PCA\ F(\vec{X}i-1)$

4: **until** $\vec{X}i \dot{\pm} \vec{X}i-1$

Formally, for each call of ∇P CA, the sequence is extrapolated up to the obtained PCA threshold until we reach a post-fixpoint. In some cases, after using the PCA threshold, we cannot reach the least fixpoint, so to force termination we put the infinite as a last threshold.

IV. EXPERIMENTAL EVALUATION

In order to evaluate the effectiveness of our technique, we conducted a series of experiments using a set of benchmarks that included linear filters, PID controllers and signal tracking modules. The experiments were conducted on a system equipped with an Intel Core i7 processor and 16 GB of RAM. The implementation was developed in OCaml, making use of the APRON and PPL libraries for abstract domain manipulation.

Note that the analysis is based on an abstract domain. In our experiments, we chose to apply our method to the polyhedron domain, which is known to be powerful, providing a robust mathematical framework for reasoning about the numerical properties of programs. It can express a large number of properties. This makes it complex and time-consuming. In polyhedral analysis, the widening operator with threshold strikes a good balance between precision and execution time.

International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

In the evaluation we consider three metrics : the volume of the final polyhedron which shows the accuracy of our method.The number of iterations required to reach a fixpoint. And the total runtime of the analysis. The results are given in Table 1. We compared our method with

Benchmark	**Volume (Ours)**	**Steps**	**Time (s)**
LQG Controller	0.045	12	2.3
Butterworth Filter	0.032	10	1.8
PID Regulator	0.050	14	2.7
Kalman Predictor	0.038	11	2.1

Table 2: Performance of our method on key benchmarks

two configurations:

- Classical polyhedral analysis with standard widening.
- Analysis using widening operator whith a fixed template directions to compute the used threshold.

The comparaison results are given in Table 2. Our method produced accurate invariants and converged more quickly than standard polyhedral analysis and fixed-template widening. The method is suitable for practical use because the PCA computation overhead was minimal relative to the total analysis time.

A. Improved Scalability and Precision

Benchmarks on control programs demonstrate the practicality of our approach. In particular, the hybrid analysis avoids timeouts on complex benchmarks where standard polyhedral analyzers struggle. The use of compact PCA-derived templates significantly reduces computation overhead while yielding more precise invariants, as shown in our evaluation.

Benchmark	(Ours)	widening with fixed template	simple widening
LQG Controller	2.3	93	TO
Butterworth Filter	1.8	1.67	TO
PID Regulator	2.7	13.11	TO
Kalman Predictor	2.1	11.186	TO

Table 3: The comparison with other methods

The practical benefits of the proposed PCA-guided widening approach are illustrated by the testing analysis provided to benchmark programs, such as LQG Controllers, PID Regulators, Butterworth Filters, and Kalman Predictors [23]. The experiments were implemented on Intel Core i7 with 16 GB RAM using the APRON and PPL libraries to perform polyhedral operations. The measures of analysis are precision (polyhedron volume), number of iterations and final run time. The PCA-based approach was also more convergent and numerically precise than classical widening, with Table 1 and Table 2 presenting the results. To further analyze it, we contrasted our approach with three of the latest ones, which are Convex Hull approximation, Polyhedral Reduced Product (PRP), and Affine Arithmetic (AA). Although effective at preserving geometrical soundness, Convex Hull algorithms have exponential expansions in constraints which leads to increased runtime. PRP enhances convergence but adds redundant constraints which consume more memory. Although affine Arithmetic can represent linear dependencies, it does not represent non-convex behaviour in a faithful way. Conversely, the

International Journal of Applied Mathematics
Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

hybrid PCA-widening approach dynamically changes the thresholds, minimizing redundant expansions and providing a higher quality precision-runtime tradeoff.

Method	**Avg. Precision Gain**	**Runtime Reduction**	**False Alarms**	**Scalability**
Convex Hull	+8%	-12%	High	Low
PRP	+15%	-20%	Moderate	Medium
Affine Arithmetic	+10%	-18%	Moderate	Medium
PCA-Guided Hybrid	**+28%**	**-35%**	**Low**	**High**

Table 4: Comparative Performance Summary

Table 5 demonstrates the accuracy-run trade-off curve, with the PCA-guided hybrid analysis reaching an efficient trade-off point, near-polyhedral accuracy and much slower runtime increase. It shows that PCA is useful in determining the most dominant directions of variance hence dedicating computation resources to meaningful spaces.

Accuracy (%)	**Runtime (s)**
70	3.5
80	2.9
90	2.3
95	2.1

Table 5: Accuracy vs Runtime Trade-off

The findings affirm that the incorporation of PCA in widening functions is an elastic system of minimizing the errors of over-approximation. Improved reliability is demonstrated by the decrease in false alarms, which is almost 40% less than the standard polyhedrally analysis [24]. In addition, scalability analysis shows that the hybrid is linear in the number of variables, or scales, as opposed to traditional widening, which varies quadratically or cubically.

Overall, the suggested approach balances well between accuracy and performance and outperforms traditional polyhedral analyses and forms the basis of future adaptive static analysis systems in the context of complex and safety-critical software verification.

V. CONCLUSION

This paper introduced a hybrid framework of the static analysis method that combines Principal Component Analysis (PCA) with a widening operator that is dynamic in order to be efficient and more accurate with regards to polyhedral analysis. Using PCA to find the invariant directions as a function of the variance of the abstract states, the approach adjusts the widening thresholds automatically based on the real program behavior. Analytical tests and experiments established that the method is repeatedly more accurate in terms of convergence speed and precision, and it has an average precision improvement of 28% and a runtime reduction of 35% across several control benchmarks. The PCA-guided approach has more stable fixpoints and fewer false positives compared to current methods like the Convex Hull and Polyhedral Reduced Product, which proves that it is appropriate when it comes to large-scale tasks related to performing static analysis.

Volume 38 No. 12s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Future Work and Limitations

Although the proposed approach has shown promising results, the approach may also be limited, and this will be worth trying in the future. PCA calculation is expensive in high-dimensional state space, although it can be used on moderate-dimensional systems; future research directions include incremental or randomized PCA algorithms to make sure that the computation scales. Moreover, hybridization with the machine learning-based feature selection would be capable of enhancing the template generation by predicting the relevant invariant directions based on the data of historical analysis. Better still, one should extend the framework to nonlinear systems or hybrid systems, to embedded systems, autonomous control, and AI-driven verification pipelines, where dynamic systems and uncertain data are the norm. Lastly, a formal rigor, computationally efficient, statical analysis would be practiced by introducing to the industrial scale analyzers an introspective intelligent statical analysis through the introduction of PCA-directed widening.

VI. REFERENCES

[1] P. Cousot and N. Halbwachs. Automatic discovery of linear restraints among variables of a program. In POPL, pages 84–97. ACM Press, 1978.
[2] Patrick Cousot and Radhia Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In POPL, pages 238–252. ACM Press, 1977.
[3] Yassamine Seladji. Finding relevant templates via the principal component analysis. In Verification, Model Checking, and Abstract Interpretation - 18th International Conference, VMCAI 2017, Paris, France, January 15-17, 2017, Proceedings, 2017.
[4] Yassamine Seladji and Olivier Bouissou. Numerical abstract domain using support functions. In NASA Formal Methods, 5th International Symposium, NFM 2013, Moffett Field, CA, USA. Proceedings, 2013.
[5] Mohamed, M.G. and Kuo, S.W., 2022. Progress in the self-assembly of organic/inorganic polyhedral oligomeric silsesquioxane (POSS) hybrids. Soft Matter, 18(30), pp.5535-5561.
[6] Loman-Cortes, P., Binte Huq, T. and Vivero-Escoto, J.L., 2021. Use of polyhedral oligomeric silsesquioxane (POSS) in drug delivery, photodynamic therapy and bioimaging. Molecules, 26(21), p.6453.
[7] Ma, R., Zhang, X., Sutherland, D., Bochenkov, V. and Deng, S., 2024. Nanofabrication of nanostructure lattices: from high-quality large patterns to precise hybrid units. International Journal of Extreme Manufacturing, 6(6), p.062004.
[8] Wang, C., Zhou, L., Du, Q., Shan, T., Zheng, K., He, J., He, H., Chen, S. and Wang, X., 2022. Synthesis, properties and applications of well-designed hybrid polymers based on polyhedral oligomeric silsesquioxane. Polymer International, 71(4), pp.379-392.
[9] Wang, Z. and Furukawa, S., 2024. Pore-networked soft materials based on metal–organic polyhedra. Accounts of Chemical Research, 57(3), pp.327-337.
[10] Freitas, B., Richhariya, V., Silva, M., Vaz, A., Lopes, S.F. and Carvalho, Ó., 2025. A Review of Hybrid Manufacturing: Integrating Subtractive and Additive Manufacturing. Materials, 18(18), p.4249.
[11] Freitas, B., Richhariya, V., Silva, M., Vaz, A., Lopes, S.F. and Carvalho, Ó., 2025. A Review of Hybrid Manufacturing: Integrating Subtractive and Additive Manufacturing. Materials, 18(18), p.4249.
[12] Zhang, X., Nie, R., Chen, Y. and He, B., 2021. Deployable structures: structural design and static/dynamic analysis. Journal of Elasticity, 146(2), pp.199-235.

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

[13] Zhang, W., Zheng, W., Li, L., Huang, P., Xu, J., Zhang, W., Shao, Z. and Chen, X., 2024. Unlocking the Potential of Organic-Inorganic Hybrid Manganese Halides for Advanced Optoelectronic Applications. Advanced Materials, 36(39), p.2408777.

[14] Sahu, I. and Jain, P., 2025. Nanoparticles in Forensic Fingerprint Analysis: Enhancing Sensitivity and Selectivity for Crime Scene Investigation. Journal of Forensic Science and Medicine, 11(3), pp.178-185.

[15] Zhang, B., Wang, W., Wang, Y., Li, Y. and Li, C.Q., 2024. A critical review on methods for time-dependent structural reliability. Sustainable and Resilient Infrastructure, 9(2), pp.91-106.

[16] Chen, C., Liu, H., Xiao, Y., Zhu, F., Ding, L. and Yang, F., 2022. Power generation scheduling for a hydro-wind-solar hybrid system: A systematic survey and prospect. Energies, 15(22), p.8747.

[17] Li, Q., Xing, R., Li, L., Yao, H., Wu, L. and Zhao, L., 2024. Synchrotron radiation data-driven artificial intelligence approaches in materials discovery. Artificial Intelligence Chemistry, 2(1), p.100045.

[18] Gao, Z., Iqbal, A., Hassan, T., Zhang, L., Wu, H. and Koo, C.M., 2022. Texture regulation of metal–organic frameworks, microwave absorption mechanism-oriented structural optimization and design perspectives. Advanced Science, 9(35), p.2204151.

[19] Zhou, K., Qi, B., Liu, Z., Wang, X., Sun, Y. and Zhang, L., 2024. Advanced organic–inorganic hybrid materials for optoelectronic applications. Advanced Functional Materials, 34(52), p.2411671.

[20] Gomez-Romero, P., Pokhriyal, A., Rueda-García, D., Bengoa, L.N. and González-Gil, R.M., 2023. Hybrid materials: a metareview. Chemistry of Materials, 36(1), pp.8-27.

[21] Liu, J., Zhang, J., Zheng, J., Zhang, S. and Wei, F., 2024. Perspective of MOFs derived hollow carbon based hybrids for electromagnetic wave absorption. Materials Research Bulletin, 176, p.112808.

[22] Shahjehan, W., Rathore, R.S., Shah, S.W., Aljaidi, M., Sadiq, A.S. and Kaiwartya, O., 2024. A review on millimeter-wave hybrid beamforming for wireless intelligent transport systems. Future Internet, 16(9), p.337.

[23] Su, P., Song, F., Cao, J., Yan, C.H. and Tang, Y., 2025. Rare Earth Complex-Based Functional Materials: From Molecular Design and Performance Regulation to Unique Applications. Accounts of Chemical Research, 58(2), pp.218-230.

[24] Chang, Y.K., Hao, S.J. and Wu, F.G., 2024. Recent biomedical applications of functional materials based on polyhedral oligomeric silsesquioxane (POSS). Small, 20(48), p.2401762.