Received March 24, 2021, accepted April 12, 2021, date of publication April 19, 2021, date of current version April 27, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3074062

Autonomous Vehicles Scenario Testing Framework and Model of Computation: On Generation and Coverage

ALA' J. ALNASER[1], ARMAN SARGOLZAEI[2], (Member, IEEE), AND MUSTAFA ILHAN AKBAŞ[3], (Member, IEEE)

[1]Department of Applied Mathematics, Florida Polytechnic University, Lakeland, FL 33805, USA
[2]Mechanical Engineering Department, Tennessee Technological University, Cookeville, TN 38505, USA
[3]Department of Electrical Engineering and Computer Science, Embry-Riddle Aeronautical University, Daytona Beach, FL 32114, USA

Corresponding author: Ala' J. Alnaser (ala.aj.alnaser@gmail.com)

This work was supported by the National Science Foundation under Grant CNS-1919855.

ABSTRACT Autonomous vehicle (AV) technology started to shift the perception of the transportation systems. However, for AVs to operate at their optimum capabilities, they need to go through a comprehensive testing and verification process. While a large amount of research and funding has been provided for solving this problem, there is still a lack of a systematic method to develop standardized tests that can be used to judge if the decision-making capability functions are within acceptable parameters. To that end, the tests need to cover all possible situations that an AV may run into. This paper focuses on defining the notion of coverage mathematically when using pseudo-randomly generated simulations for testing. The approach defines new equivalence relations between scenes, which are the systems' various states, to achieve this goal. Considering the substantial need for computation, even with the obtained coverage, we also introduce the mathematical definition of a sub-scene and additional strategies, such as expanding the equivalence classes of scenes and combining actors in scenes, to reduce the amount of testing required to certify AVs.

INDEX TERMS Autonomous vehicles, coverage, model of computation, safety, testing and verification framework.

I. INTRODUCTION

In In recent years, considerable advancements have been made in processing speeds, machine learning, and machine vision. In addition, significant progress has been made in the developing and implementation of sensory technologies such as Light Detection and Ranging (LIDAR) and Radio Detection Ranging (RADAR). These have allowed for remarkable leaps in the advancements of AVs. In the United States alone, 94% of the accidents are completely or partially caused by human errors [1]. Many of these accidents may be reduced in severity or avoided entirely with AVs' deployment on the roads. However, in order to fully take advantage of this technology, there is still a need for constructing test scenarios to verify its safety and security.

The issues and procedures for the safety evaluation of Advanced Driver Assistance Systems (ADAS) have been studied for the last couple of decades [2], [3]. With the increasing interest in AV technologies, we have observed a dramatic shift in verification methods during the past several years. Several approaches used a simulation-based critical-scenario identification platform through a metrics-based filter to define critical situations [4]. Others used a method that focuses on building interesting scenario tests within the constraints of physical test resources [5]. In another instance, the method introduced in [6] only focused on generating test cases that were thought of as critical depending on a specific metric.

The validation and verification of autonomous vehicles have gained increasing attention as the potential impact of these systems on the transportation system is realized [3]. Hence, important companies in the automaker, supplier, and technology industries presented their work on AV validation in a recent report [7] with the goal of a comprehensive representation of the industry's efforts. Despite the great investment and effort in the AV testing and validation, the report

The associate editor coordinating the review of this manuscript and approving it for publication was Lorenzo Ciani.

shows that automated driving solutions still require a systematic demonstration of a low-risk factor compared to traditional human driving. A series of interviews conducted with multiple automotive experts pointed out that"Creating test scenarios" and "ensuring completeness of requirements" as the most important challenges of ADAS and AV testing [8]. Even though these challenges are identified, no complete solution has been proposed, and requirements-based testing is identified as incapable of capturing the test space. Since the testing space for verification scenarios has been realized to be extremely large, these approaches aim for the generation of rare or challenging scenarios and do not tackle the challenge of scenario coverage. In other words, when testing an AV one must be certain that the AV will perform according to acceptable parameters in every instance similar to the challenge in the verification scenario. Without having a systematic approach that is proven to guarantee coverage, the examiner may not be able to determine the number of variations of the same scenario that must be created in order to test the AV under in all possible cases.

A trusted verification approach has been adopted recently from the semiconductors and chip manufacturing verification field [9]. The developed verification approach enables the separation of concerns and focuses solely on the AV's decision-making capability. Building upon results of the approach described in [9], this work concentrates on refining the formulation and construction of scenes mathematically. Furthermore, this paper generalizes the equivalence relation between scenes to produce equivalence classes of scenes that will reduce the amount of testing needed substantially.

A. RELATED WORK

The most commonly used methodologies for providing coverage are search-based testing, pseudorandom test generation, and reinforcement learning-based test scenario generation. The testing methodologies that use a search-based approach are efficient, particularly in the software domain [10], [11]. The main challenge in these approaches is generally the requirement for manual guidance and personnel with expertise. Pseudorandom test generation methods have been used with various simulators to test AV decision and control systems [12]–[15] in a significantly large number of randomly generated scenarios. These methods are generally designed to be calibrated and improved over time using the feedback from conducted tests. Lastly, reinforcement learning has been gaining more interest lately in the testing of autonomous systems [16], [17]. In reinforcement learning-based approaches, testing methods are designed to automatically learn the test case selection and prioritize the improvement of test quality and coverage.

One of the critical considerations in coverage-based approaches is finding the performance boundaries for AVs in test scenarios to limit the verification space. For instance, Waymo drives its simulation testing scenarios with the real-world disengagement data collected by its fleet [18]. Then, a variety of scenarios are generated by fuzzing the collected data to improve the total coverage. In another study, Mullins *et al.* [10] propose generating test scenarios based on performance boundaries and then group the scenarios using clustering techniques. Using this method, the coverage is defined as the ratio of decision boundary regions sampled. Khastgir *et al.* [8] aim to increase the coverage of the test scenario space by using a two-stage approach. While the first stage is a traditional requirements-based testing approach, the second stage focuses on the identification of hazards. Therefore, the system increases the scenario space by creating tests on how the system works and how it may fail. Auto-encoders are also used to reduce the testing space for AVs. Langner *et al.* [19] use the reproduction error as a novelty indicator for the real-world driving data sequence. The approach cannot be verified globally, yet it has been used to identify performance indicators in the driving data for testing.

Another approach that aims to tackle the coverage challenge has been the utilization of multiple types of virtual prototyping. Zofka *et al.* [20] integrated a vehicle mechanics simulation, sensor simulation, and traffic flow simulation and used them in both software and vehicle in the loop simulations. While vehicle mechanics and sensor simulation provide noisy information access, the traffic simulator provides the real-world traffic scenarios that the AV can come across. Kim *et al.* [21] combine multiple types of factors, such as road geometry, environmental factors, and dynamic object behavior, to abstract the virtual world and express all these factors as formally defined test criteria.

Machine learning and artificial intelligence form the core of the perception and decision-making engines of AVs. Even though these technologies are mainly used in the design, there are applications of them in testing, as well. Tuncali *et al.* [22] use machine learning components in their system to generate adversarial test scenarios for AVs. The system uses arrays of test combinations and simulated annealing to find edge scenarios and evolves in time with the feedback from the vehicle's controller. Jenkins *et al.* [23] use recurrent neural networks (RNNs) for the automatic specification of AV test scenarios. The RNNs are applied to existing crash data to create test cases. Deep test [24] is also proposed for the automatic testing of AVs using neural networks. A deep test aims to generate test scenarios that maximize the neuron coverage in deep neural network-based AVs. The scenario parameters such as environmental factors are transformed to form new test cases and show AV behavior errors.

Despite the remarkable work of the verification mentioned above methods with various coverage goals, there is a lack of clear formulation for the definition and generation of test scenarios.

B. BACKGROUND

In order to proceed with the mathematical notion of this paper, we need to have a mathematical model for the scenes and scenarios. Furthermore, we need a mathematical definition of how actors interact within scenarios and how scenarios can be combined or split. Here, we will be using the concepts and definitions introduced earlier in [9].

Definition 1 (A Scene Vector): $C_k \in \mathbb{R}^{n_i}$ is a vector of real values representing the 3-D spherical environment around the vehicle or the Unit Under Test (UUT) within $\mathcal{N}_k$ units of distance at any moment of time ($k = t\Delta t$). The dimension of the vector n_i corresponds to the number of parameters used to represent the UUT and the other actors in the sphere. The distance $\mathcal{N}_k$ will be called the Radius of the Scene Vector. Furthermore, a scene vector is broken down into four main components or sub-vectors:

- The parameters describing the Ego vehicle or UUT; including its dimensions, position, velocity and acceleration vectors, etc.
- The dynamic actors; which are the moving components in this sphere. Each dynamic actor has its velocity, acceleration vectors, and the relative distance between the actor and the UUT.
- The constants or static components; which include the relative distance between the UUT and any non-moving object or structure in the scene, in addition to the dimensions of the object.
- The communication between the UUT and the other actors and components in the scene; which would include physical, electronic and wireless communications between the actors and other components in the scene such as an actor signaling to change lanes in front of the UUT.

To illustrate the scene vector, let us consider a simple case; Suppose at a certain time step k, the UUT is travelling on a one lane road following another vehicle (actor 1). Assume that $r_{ego}(k)$, $v_{ego}(k)$, and $a_{ego}(k)$ are the position, velocity and acceleration vectors of the ego (UUT) at the time step k. Let $r_1(k)$, $v_1(k)$, and $a_1(k)$ be the position, velocity and acceleration vectors of the of the actor at the same time step. Also, let $d_1(k)$ be the distance between actor 1 and the UUT and $\mathcal{N}$ be the radius of the scene. Then the scene vector can be constructed as: $C_k = \big(\mathcal{N}, r_{ego}(k), v_{ego}(k), a_{ego}(k), r_1(k), v_1(k), a_1(k), d_1(k)\big)$. Note that, one can add other parameters describing the road and any other component of the environment to the scene vector when needed.

Definition 2 (A Scene): χ is the 3-D environment constructed by accumulating the scene vectors with consecutive time steps up to the present time step. Also, the **Scene Radius** $\mathcal{N}$ to be the maximum of all the scenes' radii up to the present time step k. A Scene χ can be modeled as

$$\chi \triangleq [C_0, C_1, \ldots, C_k]$$

Remark 1: The scene vectors forming the columns of a scene matrix can possibly have different sizes. Therefore, the empty entries can be filled with zeros and define the size of the scene matrix to be:

$$\max\{\text{number of rows in } \mathbf{C}_i\} \times (k+1), i \in \{0, 1, \cdots, k\}.$$

Remark 2: The scene vector C_k is computed at a specific time step, and it represents the scene as a snapshot of the reality around the UUT at that time step. A scene represents the space-time (4-dimensional space) around the UUT up to the current point of time.

Remark 3: $\mathcal{N}$ is chosen to be large enough to capture effects of neighboring objects on the UUT but small enough for all of the reasonably small number of equivalence classes.

While each scene is defined at a fixed time step, there is a need for a function that allows for the transition between time steps and, therefore, between scenes. Moreover, this function will also need to take into account the changes caused by the environment and the other actors in addition to the changes caused by the UUT.

Definition 3: Given a scene χ and suppose C_k is the scene vector at time step k, the next state can be calculated using the Newtonian laws of motion as

$$C_{k+1} = \zeta(C_k, \text{Ego's Desired Input, other actors' input})$$

where ζ is a function with the input of the current scene vector, communication from other actors, namely their velocities, positions, directions, and the desired action of the ego, such as velocity and position, its output is the scene vector of the following time step that is computed, and C_{k+1} is the vector corresponding to the next time step.

Next, we define *assertions* mathematically, which will allow us to determine if the UUT passes or fails in a certain scene.

Definition 4: Given a scene κ with a radius $\mathcal{N}$, we define the Assertion function ϑ as a function with the domain being the set of scene matrices with an interval range of [0 1], which has a predetermined set of weighted assertions. The output of this function is a probability (or a percentage) calculated as a weighted average based on the predetermined assertions where an output of 0 means the UUT fails and an output of 1 represents the UUT passing.

C. CONTRIBUTIONS

When generating simulations for testing any system, one needs to make sure that every possible situation is covered in the test. However, the amount of simulations necessary to cover every possible case is incredibly large due to the large number of involved parameters and variables. Furthermore, coverage cannot be gained by real-world driving without driving an astronomical number of miles [25], even if driving the needed number of miles is possible, the results will not be accurate due to the amount of time that will be needed and the wear and tear on the vehicles.

Based on the best of our knowledge, none of the current testing and verification techniques in the literature considers the coverage analysis and equivalent classes, which has been proposed in this paper. In this work, a novel mathematical notion for coverage (*coverage matrix*) is introduced by defining the variations of a single scene and identifying how scenes can be equivalent to each other. When two scenes are equivalent, then the AV will behave in the same way. Suppose the AV makes an undesirable or unsafe decision in one of the scenes. In that case, it will make a similar decision in all the other equivalent scenes. Therefore, the AV can be tested at least once, and there is no need to test AVs under equivalent scenes multiple times. Besides, a methodology to compute an upper bound for the number of variations of a scene depending on the scene's Radius (Definition 2) is presented. Finally, two techniques to reduce the computational testing time are introduced.

This paper will follow the following outline:

- Section II describes the proposed flow for creating test cases.
- The mathematical approach to the segmentation and the addition of actors is illustrated in Section III.
- Section IV introduces an equivalence relation between scenes which generates adjustable equivalence classes based on the requirements of the testing.
- The Meaningful Variation of a Scene is presented in Section V.
- Section VI introduces a new method for discretizing the space within a scene.
- A method for enumerating actors and scenes is presented in Section VII.
- Finally, Section VIII develops another strategy that can reduce the amount of testing required to verify an AV system's ability to make safe and correct decisions.

II. TEST CONSTRUCTION FLOW

As described in Subsection I-B, one can consider a scenario as the system consisting of the AV (UUT), the other actors, and its environment. The scenario can be constructed as a sequence of scenes (states of the system) taken at each time-step. The Scene Vector consists of functions of time describing the position, velocity, acceleration, and relative distances, among other variables. These time functions are then discretized using time-steps where we assume that nothing varies within the time-step, but decisions and any other changes such as other actors' input or a change in the road happen between time-steps. Also, the concept of equivalence classes of scenes is introduced in [9] for the first time. The main purpose of the equivalence classes of scenes is to cover all distinct possible situations or scenarios. This reduces the required time needed to generate random scenarios while ensuring sufficient coverage with respect to the parameters of interest.

Assumption 1: This work follows the same primary assumption used in [9]. That is, we will separate our concerns and assume that the UUT has a near-perfect perception of the environment. In other words, we assume that the sensors and sensory input fusion algorithm and the mechanical components are handling all the AV actions are functioning ideally. Thus, this paper focuses on the UUT's decision-making capability and the testing methodology.

The flow of the proposed methodology is illustrated in Figure 1. A route is selected then divided into segments based on points of interest, such as intersections, turns, etc. The route can be extracted from an actual road; this would allow the validation of the simulations in a real-world environment test on an actual road. The division of the route is accomplished in three stages, as shown in Figure 2. The segmentation starts by considering the static components, namely the road structure and non-moving obstacles on the road, such as intersections, merging or splitting lanes, speed limit changes, etc. Afterward, dynamic components and actors can be added with a final layer of assertions then placed on top. These assertions are dependent on each dynamic and static component in the road segment. For instance, if a stop sign and a person riding a bike are introduced into the scene, then the assertions would be defined as the UUT's velocity must equal zero at the stop sign and maintain a safe distance from the biker.

The segments are then used to build a database of unique road segments that can be used for constructing simulations for the given scenes and other scenes in future testing. Next, the segments will be discretized based on time and space to reduce the scene's complexity and construct equivalence classes for the scenes. The equivalence classes are grouped by similarity so that similar situations can be tested without repetitions. Furthermore, this will also reduce the quantity and time required for testing.

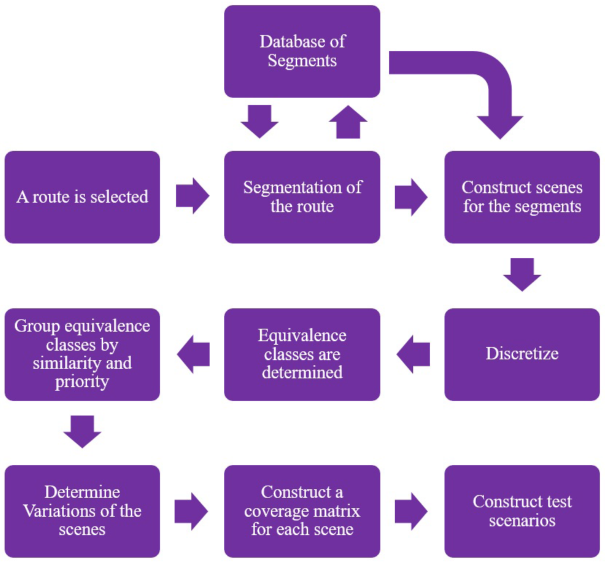
Database of Segments
A route is selected
Segmentation of the route
Construct scenes for the segments
Group equivalence classes by similarity and priority
Equivalence classes are determined
Discretize
Determine Variations of the scenes
Construct a coverage matrix for each scene
Construct test scenarios

FIGURE 1. Flow for creating test cases.

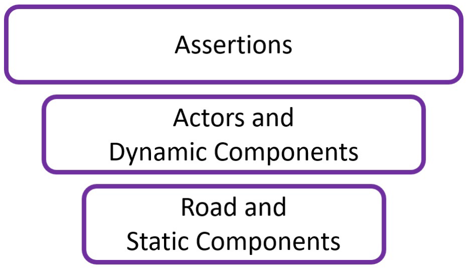

FIGURE 2. Segmentation procedure from the bottom to the top.

Now, we need to determine variations of the class representatives of each equivalence class. Some scenes may vary from each other, but these variations may offer no difference from the abstraction point of view. For instance, one scene might have a bus a certain distance away from the UUT, and another scene might have a truck in approximately the same location as the bus in the first scene relative to the UUT. Therefore, We will only be interested in meaningful variations (see Definition 7 in Section V). This step aims to cover every possible variation of the scene in the test, which is constructed in the following steps. This way, we can be certain that the UUT's decisions will remain consistently correct and safe.

III. SEGMENTATION

One of our objectives is building a database of distinct road pieces that can be used to build a variety of roads. This can be accomplished by mathematically defining the road segments and defining an equivalence relation on the set of segments that allow us to construct a partitioning on it using equivalence classes. Next, a class representative from each equivalence class can be identified and added to the database.

Roads can be modeled as a 3−dimensional parametric surface represented by vector valued function defined over a 2−dimensional region. Hence, dividing the road into segments is equivalent to partitioning the 2−dimensional domain of the parametric surface. Here, one can choose to partition the domain in multiple ways, such as small identical rectangles. However, it would be more useful if the partitioning is done so that the corresponding road segments meet at points where changes in the road occur, such as the points listed previously (intersections, merging lanes, and so on).

The vector valued function representing the road as a parametric surface is given by:

$$r(u, v) = \big(x(u, v), y(u, v), z(u, v)\big), \tag{1}$$

where U, V are fixed positive real numbers, and the 3−dimensional vector $\big(x(u, v), y(u, v), z(u, v)\big)$ is the position of every point on the surface for every (u, v) in $[0, U]\times[0, V]$.

Next, the two intervals $[0, U]$ and $[0, V]$ are divided into n and m sub-intervals respectively. Let $U = u_n$ and $V = v_m$, then for each $1 \le i \le n$ and $1 \le j \le m$, we get

$$r_{i,j}(u, v) \triangleq r(u, v), \quad \forall (u, v) \in [u_{i-1}, u_i] \times [v_{j-1}, v_j]. \tag{2}$$

Now, we can consider the set $S_r = \{r_{i,j} | i = 1, 2, \dots, n$ and $j = 1, 2, \dots, m\}$ and find the maximal linearly independent subset. After re-indexing, we will define a linearly independent set by $S = \{r_{i,j} | i = 1, 2, \dots, l$ and $j = 1, 2, \dots, p\}$. Hence, $r(u, v)$ can be written as:

$$r(u, v) = \sum_{i=1}^{l} \sum_{j=1}^{p} c_{i,j} r_{i,j} \tag{3}$$

for some constants $c_{i,j}$.

Remark 4: The set S_r is a finite set of vectors; therefore, it must contain the maximal linearly independent subset S.

Remark 5: The set S forms a basis of a space of road segments, and thus the representation in Equation 3 is unique. That is, the set of coefficients $c_{i,j}$ required to construct $r(u, v)$ are unique. Furthermore, S can be used to construct many variations of the initial route that we started with.

Remark 6: One can now consider other routes and whenever a route segment cannot be represented as a linear combination of the vectors in S, then the basis will be updated by adding the new vector for this segment to S.

Next, one needs to consider the dynamic components such as pedestrian and other vehicles as well as the assertions that each component (static and dynamic) introduces in the scene. To accomplish this, we can allow the parameters u and v to be functions of time t_l over an interval I_l for each actor in the scene, and consider the expansion on Equation 2.

$$\begin{aligned} r_{i,j}(u(t_l), v(t_l)) &= r(u(t_l), v(t_l)) \\ &\forall (u, v) \in [u_{i-1}, u_i] \times [v_{j-1}, v_j], t_l \in I_l \end{aligned} \tag{4}$$

Here, one can apply multiple assertions in the form of restrictions. For instance, the actors may not collide or be too close to each other in the scene. To accomplish this, simulations can be set up by excluding paths $r_l(t_l)$ that would intersect over the given intervals. Afterward, we can proceed as follows:

- Each segment is then used to construct one scene or more based on the previous steps. Equivalence classes are set up for the scenes, and class representatives are selected.
- The equivalence classes are also created using a similar approach to what was described in the previous step. In particular, for each set of equivalent segments of roads, only one is chosen to produce a set of nonequivalent road segments. Subsequently, for each one of the road segments, actors and other dynamic components are introduced and using their parameters, and the assertions on these parameters equivalence classes can be finally created (see Section IV).
- The set of all the meaningful variations of each scene class representative will then be constructed and put together to form the coverage matrix for that scene (see Section V).
- Verification tests can then be set up using the coverage matrices by putting the UUT through each meaningful variation of each scene.

As a clarification example, consider a T-Junction as represented by the Figure 3. $r(u, v)$ be the entire piece of road with traffic oriented upwards along the $y-$axes and then from left to right along the horizontal section of the junction. Now, since this is a 2-dimensional small example, we will use matrix notation for the vectors, and we will break up $r(u, v)$ into multiple segments and add them up as follows: Let

$$r_1(u, v) = \begin{bmatrix} x(u, v) \\ y(u, v) \end{bmatrix} = \begin{bmatrix} u \\ v \end{bmatrix}, \text{where } -1 \le u \le 1, 0 \le v \le 2,$$

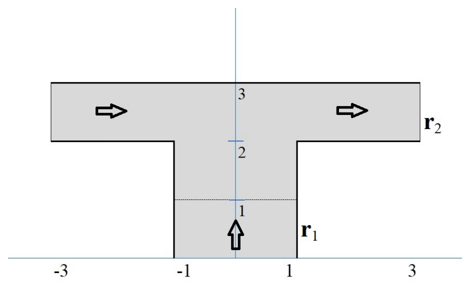

FIGURE 3. **Example of Segmentation.**

and

$$r_2(u,v)=\begin{bmatrix} x(u,v) \\ y(u,v)\end{bmatrix}=\begin{bmatrix} u \\ v\end{bmatrix}, \text{ where } -3\le u\le 3, 2\le v\le 3,$$

then $r(u,v)=r_1(u,v)+r_2(u,v)$.

While our aim here is to clarify the process, it is worth noting that it is possible to segment further and use smaller road segments as building blocks. Also, one can also use translation and rotation matrices and the Heaviside function to write $r(u,v)$ as a sum of copies of the unit square oriented upwards.

If $r_0(u,v)$ is the unit square oriented upwards, then to simplify the computation, we can consider r_0 as lifted up 1 above the plane, that is, we let

$$r_0(u,v)=\begin{bmatrix} x(u,v) \\ y(u,v) \\ 1\end{bmatrix}=\begin{bmatrix} u \\ v \\ 1\end{bmatrix}, \text{ where } 0\le u\le 1, 0\le v\le 1,$$

then we set up the matrices

$$A=\begin{bmatrix} 2 & 0 & -1 \\ 0 & 2 & 0 \\ 0 & 0 & 1\end{bmatrix}, \quad B=\begin{bmatrix} 0 & -6 & 3 \\ -1 & 0 & 3 \\ 0 & 0 & 1\end{bmatrix}$$

The matrix A is a translation and stretching matrix that will use the unit square and create the T-junction's vertical leg while keeping the orientation (traffic) pointing upwards. While B causes a clockwise rotation by 90°, and vertical translation then a horizontal stretch. Thus we get

$$r(u,v)=Ar_0+Br_0 \text{ where } 0\le u\le 1, 0\le v\le 1.$$

Lastly, one may not factor r_0 out in an effort to simplify the computation by adding the two matrices. It is important to note that the 2 matrices A and B are composite transformation matrices which means adding them up will change all the operations they are meant to perform.

IV. EXPANDING THE EQUIVALENCE CLASSES

In [9], the authors presented an equivalence relation between scenes that yielded equivalence classes of scenes which can be used to construct testing simulations that use distinct equivalence class representatives. However, the equivalence relations were restrictive, causing the classes to be relatively small in size. This implies that the tests will have to involve a large number of class representatives. Hence, our next objective is to extend the equivalence relation to allow for the construction of larger equivalence classes, each containing a wider range of scenes. This would yield a reduction in the number of class representatives, which will be used to generate verification simulations. We will also expand the equivalence in a manner that can be adjustable producing equivalence classes that can be used for sharper tests. One may adjust the *"passing"* and *"failing"* requirements of the tests; for example, if the UUT does not stop exactly before the line at a stop sign but does not exceed the edge of the curb, that might be allowed. However, differences between the distances the UUT maintains between itself and other actors and the defined "safe distance" at high speeds need to be measured and judged at higher accuracy.

Based on the definitions and methodology in [9], the order of the entries in a scene vector can be changed without actually changing the scene itself. In other words, two scene vectors are equivalent if the scenes they represent are almost identical. That is, the differences that may be allowed are the order of the actors and the order by which their parameters are listed in the scene vectors and a slight variation in the types of actors; for example, a truck can be replaced by a van with almost the same size, location relative to the UUT and traveling velocity.

Hence, We have the following:

Definition 5: Given a route of travel for a UUT where the order of the entries of all the scene vectors for all the scenes along this route is fixed. we define **The Relation** ρ such that two scene vectors $\mathbf{C}_1$ and $\mathbf{C}_2$ related via ρ (denoted by $\mathbf{C}_1\rho\mathbf{C}_2$) if and only if:

1) $\mathbf{C}_1$ and $\mathbf{C}_2$ have the same number of entries. That is, $\mathbf{C}_1$ and $\mathbf{C}_2$ belong to the same real vector space.
2) For each pair of corresponding entries of $\mathbf{C}_1$ and $\mathbf{C}_2$, say $\gamma_{1,i}$ and $\gamma_{2,i}$ respectively, there exists a fixed $M_i\ge 0$ such that

$$\left|\gamma_{1,i}-\gamma_{2,i}\right|\le M_i \quad \text{for each } i.$$

Remark 7: The choice of the constants M_i is ideally dependant on the type of road or junction that the scene covers. For instance, if the road segment is a one-lane road within a housing community with a low-speed limit, say 25 mph, then a small difference between the traveling velocity of an actor or the UUT can have a large effect. Whereas, if the scene is based on a highway segment, then the difference can be a little larger. Also, the choice allows for a change in the level of strictness the test designer requires for some or all the parameters.

Lemma 1: The relation ρ in Definition 5 is an equivalence relation.

Proof: Given a scene vectors $\mathbf{C}_k$, belonging to the real vector space $\mathbb{R}^n$. We will denote $\mathbf{C}_k=\left[\gamma_{k,i}\right]$ for $i=1,\ldots,n$. Now, one can immediately see that the relation ρ is reflexive $(\mathbf{C}_k\rho\mathbf{C}_k)$ for any k by setting the constants $M_i=0$ for $i=1,\ldots,n$.

Also, ρ is symmetric because if $\mathbf{C}_k \rho \mathbf{C}_l$ then there exist constants $M_i \geq 0$ such that $|\gamma_{k,i} - \gamma_{l,i}| = |\gamma_{l,i} - \gamma_{k,i}| \leq M_i$ for each i, hence $\mathbf{C}_l \rho \mathbf{C}_k$.

Finally, Suppose $\mathbf{C}_k \rho \mathbf{C}_l$ and $\mathbf{C}_l \rho \mathbf{C}_m$. In other words, suppose that there exist constants M_i and N_i such that $|\gamma_{k,i} - \gamma_{l,i}| \leq M_i$ and $|\gamma_{l,i} - \gamma_{m,i}| \leq N_i$ for $i = 1, \ldots, n$. Thus we have

$$\begin{aligned} |\gamma_{k,i} - \gamma_{m,i}| &= |\gamma_{k,i} - \gamma_{l,i} + \gamma_{l,i} - \gamma_{m,i}| \\ &\leq |\gamma_{k,i} - \gamma_{l,i}| + |\gamma_{l,i} - \gamma_{m,i}| \\ &\leq M_i + N_i. \end{aligned}$$

Hence, there exists a fixed constant $P_i = M_i + N_i \geq 0$ such that $|\gamma_{k,i} - \gamma_{m,i}| \leq P_i$ for each $i = 1, \ldots, n$. That is, $\mathbf{C}_k \rho \mathbf{C}_m$ and therefore ρ is transitive. □

This equivalence relation naturally gives rise to a set of equivalence classes. Consequently, if given a route, then one can divide the route into multiple scenes. Furthermore, using scenes from different equivalent classes may cover a large variety of scenarios quickly and efficiently. Moreover, by allowing the freedom of choice in selecting the fixed constant size, we have more control over the number of equivalence classes.

In addition, as mentioned previously, we can group together equivalence classes where all the actors are farther away from the UUT than a certain fixed number of cubes depending on the type of environment and consequently the speed of ego and other actors. For example, in an urban environment, objects further than, say 100 meters away, all have the same (little) effect on the UUT's decision. Whereas in a high way setting, that distance must be larger since almost all the objects and actors will be moving at a higher speed.

In the following section, some methods that can be used to reduce the coverage matrix's size and lower the number of scenes and variations of these scenes are discussed that enables testing the ability of the UUT to make the correct decision. This can be done by eliminating similar scenes, scenes that are included in larger scenes, and scenes in which actors have almost no effect on the UUT's decision. The following sections will not focus on the verification aspect heavily. Instead, they pay more attention to the concepts of coverage and generation of equivalent classes of scenes.

V. COVERAGE

Coverage is obtained when one constructs a test that can account for every possible variation of a UUT situation. However, the number of variations a single scene can have is very large and, in some cases, might be infinite. Thus, to attain sufficient coverage in testing the UUT in a single scene, one needs to find every possible equivalent scene, remove them from the test, and only use non-equivalent scenes when creating the test. For instance, if a scene consists of a one-lane road where the UUT is following another vehicle, then the UUT accelerating or the other vehicle brakes are equivalent situations. The result is that the distance between the two vehicles is decreasing, which can cause an accident. Therefore, both of these situations are considered to be equivalent. Also, suppose there is an obstacle on the road. In that case, all that matters are a general size (volume) of the obstacle and its location on the road; its exact shape would be of little relevance. If the obstacle is a rock in the middle of the road or a wooden box of similar size, then both situations would be considered equivalent. Therefore, the second objective of this paper is to address equivalent classes mathematically.

It has been shown in [9] that one could formulate a scene as a vector containing the physical parameters of the UUT, the actors, etc. Furthermore, the concept of equivalence classes of scenes is introduced. However, the definition was very general and yielded an incredibly large number of equivalence classes while the classes themselves were very small in terms of the number of scenes they may contain. The main purpose of the equivalence classes of scenes into *cover* representatives for all possible situations or scenarios is to avoid spending an incredibly long time generating random scenarios and hoping to have sufficient coverage. We aim to refine the equivalence classes and expand them. This allows for much faster and more efficient testing using much larger equivalence classes. Our expansion will allow for an adaptable test where the equivalence can be adjusted by changing the constant upper bounds for the absolute difference between the parameters depending on the actual route the test is using. Or by adjusting the actual mathematical model representing the actors.

Once equivalence classes are established, one may consider the number of scenarios or scenes that the equivalence can represent or model. To define coverage precisely, we need first to define what we would consider a variation of a single scene.

Definition 6 (A Meaningful Variation of a Scene): α is another scene β from a different equivalence class that have the same scene radius as α.

Notice that if two scenes are not equivalent but have the same radius, then they would differ in the number or types of actors in the scene. However, they would have a similar type of road structure (such as the number of lanes). Furthermore, since we have a finite scene radius, then there is a finite number of *meaningful variations* to any given scene.

More specifically, if $\mathcal{V}_{UUT}$ is the volume of the rectangular box representing the UUT in a scene of radius $\mathcal{N}$, then we have a remaining volume of $\frac{4}{3}\pi\mathcal{N}^3 - \mathcal{V}_{UUT}$. Hence, if, for instance, we consider two scenes to be a meaningful variation of each other if they have the same radius but a different number of actors, then we have an upper bound for the number of meaningful variations given by:

Once equivalence classes are established, one may consider the number of scenarios or scenes that the equivalence can represent or model. In order to define coverage precisely, we need first to define what we would consider a variation of a single scene.

Definition 7 (A Meaningful Variation of a Scene): α is another scene β from a different equivalence class that have the same scene radius as α.

Notice that if two scenes are not equivalent but have the same radius, then they would differ in the number or types of actors in the scene. However, they would have a similar type of road structure (such as the number of lanes). Furthermore, since we have a finite scene radius, then there is a finite number of *meaningful variations* to any given scene.

More specifically, if $\mathcal{V}_{UUT}$ is the volume of the rectangular box representing the UUT in a scene of radius $\mathcal{N}$, then we have a remaining volume of $\frac{4}{3}\pi\mathcal{N}^3 - \mathcal{V}_{UUT}$. Hence, if, for instance, we consider two scenes to be a meaningful variation of each other if they have the same radius but a different number of actors, then we have an upper bound for the number of meaningful variations given by:

$$\mathcal{M}_a = \frac{\frac{4}{3}\pi\mathcal{N}^3 - \mathcal{V}_{UUT}}{m_v}, \tag{5}$$

where $\mathcal{M}_a$ is the Maximum number of actors and $m_v = min\{\text{volume of the actors}\}$. Hence, we can define the following;

Definition 8 (The Coverage of a Scene): is the set of all possible Meaningful Variations of a scene. Also, **The Coverage Matrix of a Scene** is a matrix with its columns consisting of the scene vectors of equivalence classes representatives of all the meaningful variations of the scene.

VI. DISCRETIZING THE SPACE IN A SCENE

The bound in Equation 5 is not a sharp upper bound. Furthermore, if one considers the continuous space inside the scene's sphere, then the upper bound could be infinite. However, this is impossible in real life. Therefore we will move the problem from a continuous setting to a discrete setting. We have the following;

Definition 9: Let ϕ be a fixed constant distance such that the relative distance between any two distinct objects (such as actors or the UUT) in a scene is either greater than that or equal to ϕ or equal to zero apart from the road. That is, no two different objects can be within ϕ units from each other. Otherwise, we consider them as one object or that there has been a collision.

Now, let us consider a scene at a time-step k as a three-dimensional sphere with a radius N centered at the UUT. The UUT is represented by a rectangular box centered as the sphere's center. Next, the remaining space inside the sphere will be divided into equally sized cubes. These cubes are ϕ units apart (see Figure 4).

Next, for each actor in the scene, the rectangular box representing the actor intersects with some of the cubes, then the union of these cubes is considered the actor (see Figure 5).

Remark 8: The size of the cubes used to divide the space inside the sphere is not necessarily constant. In fact, we set up the cubes such that the size (length of the sides) is proportional to ϕ (say $\aleph_\phi$). Consequently, by taking a limit as ϕ approaches zero, we will effectively fill in the gaps between the cubes and increase the number of cubes indefinitely. Hence, we will get back to continuous space.

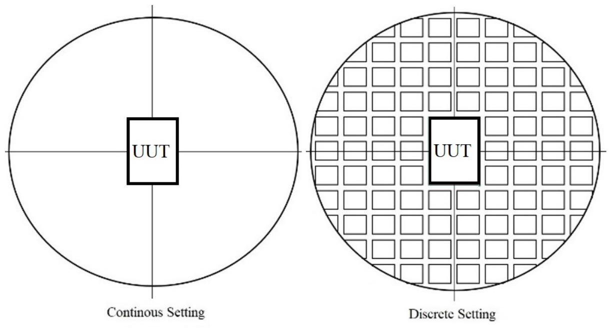

FIGURE 4. Continuous to Discrete Settings.

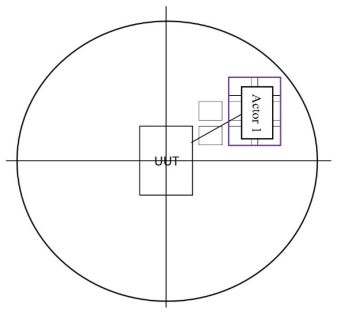

FIGURE 5. The Actor considered as a union of cubes.

Remark 9: Suppose a scene is considered at the timestep k and the cube size and the distance ϕ are set for the scene. Also, assume that some actors are close to each other where a cube cannot be fitted between them. Then these actors can be combined and thought of as a single (larger) object or actor at the time step.

Remark 10: Given an actor $a = Actor$ 1 with the relative distance d_a between it and the UUT. Suppose that there are $\widetilde{d}_a$ cubes between the UUT and the actor a. Thus, the relative distance d_a can be approximated by $\widetilde{d}_a$ a where

$$d_a \geq \widetilde{d}_a a\aleph_\phi + (\widetilde{d}_a + 1)\phi. \tag{6}$$

Hence, the relative distance d_a is bounded from below and above by the radius of the scene divided by the boxes' size. Now, we may consider a new discrete measure $\widetilde{d}_a$ as the distance between the actor and the UUT. For instance, in Figure 5 above we have $\widetilde{d}_a := 1$.

Remark 11: Based on the previous remarks, we can now consider an enumeration of actors in a scene by their discrete relative location and distance from the UUT. Notice that if more than one actor has the same distance from the UUT, we can enumerate them using ordered pairs (or triplets in 3D). We can number each actor using the closest cube coordinates that the actor covers to the UUT. The first coordinate of each cube will be the number of cubes present between the UUT and that cube horizontally. The second coordinate will be the number of cubes present ahead or behind the UUT between the UUT and the cube, and so forth. We will follow

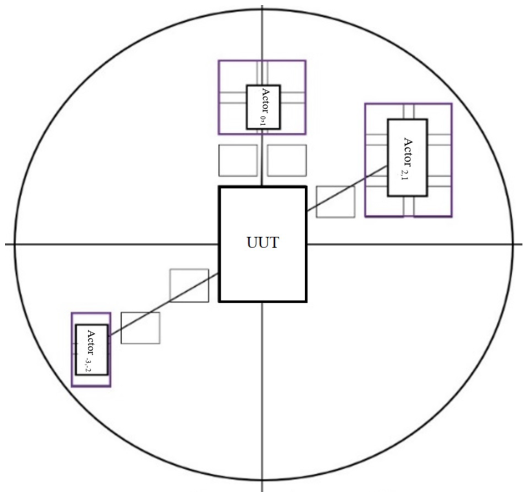

FIGURE 6. The Enumeration of 3 Actors.

the standard orientation by assigning the cubes to the right and ahead (as well as above in 3D setting) the UUT the "+" sign and cubes to the left and to the rear of the UUT "−" sign. For instance, in the scene in Figure 6 we have 3 actors assigned the enumeration (or coordinates) (2, 1), (0, 1) and (−3, −2).

One advantage of measuring distances and defining positions in the sphere of the scene discretely is that we can now enumerate objects and actors. We can also consider actors that further away than a certain number of cubes as "less important" in the scene since they would have little to no effect on the decisions made by the UUT. Another advantage is that as we move from time-step, k to the following time-step $k+1$ if the discrete relative distances between the actors and each other and the UUT do not change, then the scene at time-steps k and $k+1$ will appear almost identical. Furthermore, unless the velocity, acceleration, as well as other parameters of the scene, do not vary beyond a certain range, then the scenes at the two time-steps can be considered equivalent as previously discussed in Section IV. In this manner, we can combine time steps, which will yield fewer computations and faster testing.

For a given scene we can refine the rough upper bound we previously listed for the maximum number of actors $\mathcal{M}_a$ as follows; For a given scene with radius $\mathcal{N}$ and discretized with a fixed distance ϕ and cubes of size $\aleph_\phi$ then by assuming that the *Minimum Actor Volume* $= m_v \geq \aleph_\phi$ then equation 5 yields

$$\mathcal{M}_a \leq \frac{\frac{4}{3}\pi\mathcal{N}^3 - \mathcal{V}_{UUT}}{\aleph_\phi^3}. \tag{7}$$

Next, since the radius of the scene is fixed, then the features of the scene that can produce a variation of the scene are:

- Number of actors.
- Locations of Actors within the scene sphere.
- The velocity and acceleration of each actor.

Now, the inequality 7 gives an upper bound which can be adjusted (sharpened or loosened) based on the choices of ϕ and consequently the value of $\aleph_\phi$. Therefore, since the number of actors cannot exceed a certain fixed value, then their locations within the sphere will also be limited and identified using the enumeration scheme presented in Remark 11. Finally, later in Section IV, we will expand the definition of the equivalence relation between scenes. This will allow us to say that the velocity or the acceleration of the actors has to be significantly different for two scenes with the same radius and the same number and locations of actors not to be equivalent to each other and therefore to be counted as meaningful variations of each other.

A. SCENARIOS AND THE NEXT STATE FUNCTION IN THE DISCRETE SETTING

In [9] we introduced a method for moving from one time-step to the next. That was the *Scenario or Next State function* ζ. ζ takes the of the current scene vector, the desired action of the UUT, and communication from other actors as its input, and the output is a newly computed scene vector.

$$C_{k+1} = \zeta(C_k,\ \text{Ego's Input, Actors Communications}). \tag{8}$$

Since ζ depends on Newtonian Physics laws that are continuous in nature, this could raise an issue when we move to the discrete setting described previously, for instance, suppose that at time-step k, the UUT decided to decelerate to increase the safety distance $d_a(k)$ in response to an actor a ahead of it. Then as we move from time-step k to time-step $k+1$, the distance between the UUT and the actor head of it should be more than it was in the previous time-step (that is, $d_a(k+1) > d_a(k)$). In the discrete setting, that deceleration might not have been enough to put an additional cube or more between the UUT and the actor; therefore, the discrete distance would appear to be the same. To avoid this issue, we will keep all parameters in the scene vector such as position, velocity, and acceleration of all the actors and the UUT as well as relative distances as continuous functions of time and overlay the cubes and the constant distant ϕ after the evaluation of this parameter at each time-step. Afterward, we can adjust the enumeration of that actor, indicating that there is a change in the distance between the actor and the UUT, as explained in Remark 11 above.

B. CONCATENATING SCENES

The operator $\uplus$ is defined as the binary operator used to combine scenes in specific order [9]. As in the previous subsection VI-A, this operator depends on the continuity of the position function, as well as the next state function. Given two scenes going through multiple time-steps $\Lambda_1 = [C_0, C_1, \cdots, C_n]$ and $\Lambda_2 = [D_0, D_1, \cdots, D_m]$. The operation $\Lambda_1 \uplus \Lambda_2$ must connect the last column of the first scene with the first column of the second scene. That is accomplished by identifying $D_0 = C_{n+1}$. Also, we assume that the scenes are not overlapping (or might have a slight overlap).

This may be accomplished in the discrete setting as well. We may assume that the size of the cubes we are using to discretize both scenes is the same; we will also assume that

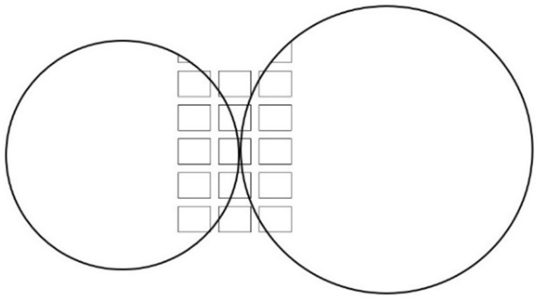

FIGURE 7. **Concatenating by merging a cube.**

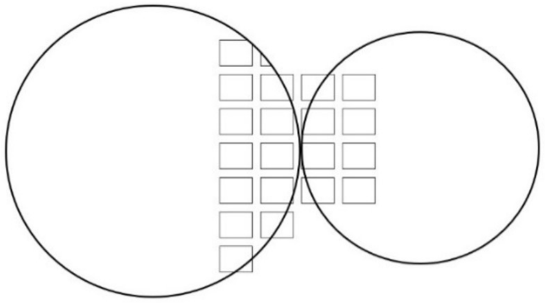

FIGURE 8. **Concatenating by adding ϕ then going to the next cube.**

the Empty space, the constant distance between the cubes ϕ, is the same for both scenes. Now, if the point where the scenes are supposed to meet is at the end of a cube from the first scene, then we can simply add distance ϕ and then start at the first you have the second scene. If, on the other hand, the first scene ends with this section over the cube, then we can merge that section with the first cube in the second scene (see the Figures 7 and 8). Since sudden changes in road conditions and structure do not happen in real life, there will be a situation where an additional small scene must be inserted between the two scenes to model the transition from the first scene to the second realistically. In this case, one may insert the transitional scene and merge it with the other two, as mentioned previously.

The discretization method introduced here allows unique scenes and meaningful variations of these scenes in terms of coverage. Furthermore, it allows us to enumerate the set of meaningful variations of a scene.

VII. ENUMERATING MEANINGFUL VARIATIONS OF A SCENE

Once all meaningful variations of a scene are identified. The next step is to identify an enumeration scheme for these scenes. In other words, we need to identify a One to One mapping from the set of all meaningful variations of a scene and the naturals or ordered tuples of the naturals.

Given a scene α, we will define the set $\mathcal{S}_\alpha$ to be the set of all meaningful variations of α.

One might use the same actors' coordinates discussed previously in Remark 11 to construct a possible enumeration. That is, say β is a variation of α with n actors then we may assign β the $(2m+1)-$tuple $(n, x_1, y_1, x_2, y_2, \ldots, x_n, y_n, 0, 0, \ldots, 0)$ where m is the maximum number of actors the scene can contain. Each pair x_i, y_i are the discrete coordinates (the enumeration) of $Actor_i$ starting with the closest actor to the furthest. This method will not yield a one to one function. Since if we consider the following case for instance:

Suppose a route is given to a UUT, and at a scene, α was taken at a time-step k. Let β and γ are two scenes in S_α and both assigned the tuple $(1, 1, 1, 0, 0, \ldots, 0)$. That is, both have a single actor, and the coordinate of the cube the actor covers that is closest to the UUT is $(1, 1)$. Now, suppose $Actor_{\beta,1}$ in β is a bus which covers the cubes that extends from the coordinate $(1, 1)$ to $(3, 1)$ horizontally and from $(1, 1)$ to $(1, 5)$ vertically and $Actor_{\gamma,1}$ in γ is a bike covering the cube with coordinates $(1, 1)$. Then these scenes are different and may require different decisions to be made by the UUT.

Using the average of coordinates will be just as misleading as the method discussed above, and again it may not produce a one-to-one mapping. However, consider the following:

Definition 10: Given a scene α let $\mathcal{S}_\alpha$ is the set of all meaningful variations of α. Let m be the maximum number of actors the scene can contain. Define the function $\Upsilon : \mathcal{S}_\alpha \longrightarrow \mathbb{N}^{4m+1}$ where $\mathbb{N}^{4m+1}$ is the set of $(4m+1)-$tuples of natural numbers such that for any $\beta \in \mathcal{S}_\alpha$,

$$\Upsilon(\beta) = (n, x_{1,1}, y_{1,1}, x_{1,2}, y_{1,2}, \ldots \\ \ldots, x_{n,1}, y_{n,1}, x_{n,2}, y_{n,2}, 0, 0, \ldots, 0)$$

where n is the number of actors in β and each pair $(x_{i,1}, y_{i,1})$ are the discrete coordinates of the cube covered by $Actor_i$ that is closes to the UUT and $x_{i,2}, y_{i,2}$ are the discrete coordinates of the cube covered by $Actor_i$ that is furthest from the UUT. The coordinates of actors are also listed starting with the closest actor to the furthest.

Remark 12: As an example, if we consider the scene in Figure 6 as δ then

$$\Upsilon(\delta) = (3, 0, 1, 0, 2, 2, 1, 3, 3, -3, -2, -3, -3, 0, \ldots, 0).$$

In addition, the two scenes mentioned above will be assigned different tuples by the function Υ.

Remark 13: It is worth mentioning that one can adjust the value of ϕ, which also means adjusting the cubes' size in the scene sphere to gain more accuracy in the outputs of Υ, which will help in distinguishing different scenes.

Remark 14: Again let us assume that β and γ are two meaningful variations of α, with

$$\Upsilon(\beta) = (n, x_{1,1}, y_{1,1}, x_{1,2}, y_{1,2}, \ldots \\ \ldots, x_{n,1}, y_{n,1}, x_{n,2}, y_{n,2}, 0, 0, \ldots, 0) = \Upsilon(\gamma)$$

then there might be still some differences between β and γ, such as β having a bus whereas γ has a truck that's around the same size as the bus and the same cubes in the scene. However, if we consider the actors as the union of the cubes they cover, then the abstraction of both scenarios β and γ will be identical. Therefore, the mapping Υ is a one-to-one

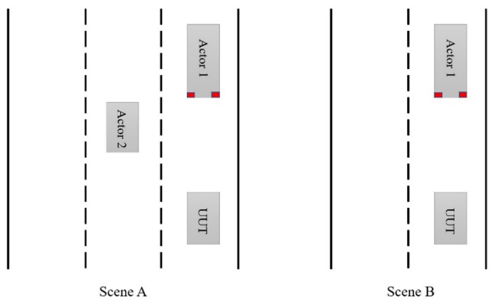

FIGURE 9. **Subscenes.**

mapping which we can use to identify and enumerate the scenes in the set $\mathcal{S}_\alpha$.

VIII. SUBSCENES

This section will focus scenes that can be considered parts of a larger scene. These *Subscenes* can then be ignored when a scenario-based test is generated to verify an autonomous system's competency since the questions they may raise will be asked by the *larger* scenes that contain them.

Consider the two scenes A and B in Figure 9. In both scenes, the UUT is driving behind another vehicle (Actor 1), slowing down. In addition, scene A has a second object or vehicle on the road (Actor 2). Assume that, with the exception of Actor 2 in scene A, the two scenes are identical.

In this situation we have the following definition.

Definition 11: Given two scenes A and B where all the information in scene B is contained in scene A and scene A contains information that is not in scene B. Then scene B is a ***Proper Subscene*** of scene A if and only if the set of assertions imposed on scene B is a subset of the set of assertions of scene A.

Hence, if scene B is a proper subscene of scene A and if $\mathbf{C}_B$ and $\mathbf{C}_A$ and the vectors representing scenes B and A respectively, then $\mathbf{C}_B$ is a subvector of $\mathbf{C}_A$. That is, after re-ordering the entries of $\mathbf{C}_A$ (if needed), we can place the rows of $\mathbf{C}_B$ first, and we will have;

$$\mathbf{C}_A = \begin{bmatrix} [\mathbf{C}_B] \\ * \\ * \\ * \end{bmatrix} \tag{9}$$

Next, we consider the relationship between the assertion matrices of scenes A and B. Suppose Γ_A and Γ_B are the assertion matrices of scenes A and B, respectively. Then Γ_B will be a sub-matrix of Γ_A

$$\Gamma_A = \begin{bmatrix} \begin{bmatrix} \Gamma_B \end{bmatrix} & * & * \\ * & * & * \\ * & * & * \end{bmatrix} \tag{10}$$

Hence, if δ_A and δ_B are the verification functions of scenes A and B respectively. Given by

$$\delta_A(\mathbf{C}_A) = \Gamma_A \mathbf{C}_A - C_{A,0} \tag{11}$$

and

$$\delta_B(\mathbf{C}_B) = \Gamma_B \mathbf{C}_B - C_{B,0}, \tag{12}$$

where $C_{A,0}$ and $C_{B,0}$ are the vectors of acceptable values of the parameters for passing for scenes A and B respectively. Furthermore, since all the assertions of B are also assertions of A, then $C_{B,0}$ is a sub-vector of $C_{A,0}$, that is;

$$\mathbf{C}_{A,0} = \begin{bmatrix} [\mathbf{C}_{B,0}] \\ * \\ * \\ * \end{bmatrix} \tag{13}$$

Therefore, we have the following result;

Theorem 1: Given two scenes A and B with B is a subscene of A, then if the UUT passes scenes A then it will pass scene B.

Proof: Follows immediately from the definition of the verification functions above. □

However, this also implies that if the UUT fails in scene B then it will fail scene A, but if it fails scene A it might still pass scene B.

IX. DISCUSSIONS AND CONCLUSION

Autonomous and connected mobility can substantially improve the transportation system's efficiency and safety and even reduce its cost. With the current advancements in machine perception and machine intelligence, there is no doubt that this new generation of vehicles will have an excellent perception of their environment. It is left for us to make sure that these Autonomous vehicles can and will make the correct and safe decisions. To that end, a clear and specific testing structure must be established. This verification and validation methodology must be based on the Newtonian Physics that governs the world that these vehicles function within. It must also follow a fixed model of computation that allows for the generation of equivalence classes and includes steps that allow us to draw conclusions on the vehicle's performance, whether within acceptable parameters.

Randomly, or pseudo-randomly, generated simulations for testing a specific parameter are common techniques used for a long time in many fields, including testing AVs. However, in the latter case, the process is much more complicated since driving in the real world involves an incredible number of variables. To construct a comprehensive test for autonomous or connected vehicles, one would need to make sure that every possible scenario has been simulated. While this is maybe close to impossible to do using real-world driving and real-world conditions, it is still possible to use computer simulations. Although, the simulations need to guarantee that every possible scenario is discovered.

To obtain desired coverage, we have presented a mathematical definition for coverage of scenes and a methodology for constructing a Coverage Matrix, which can be used for constructing comprehensive tests for the autonomous system. We have proven that the coverage matrix is bounded; furthermore, we have proven that one can consider an equivalence relation between scenes that can generate equivalence classes. We have also shown that one can modify equivalence relations between the scenes and reduce the amount of testing. In addition, a technique to discretize scenes was introduced. This yields a strategy to enumerate scenes and actors within a scene. Consequently, this enables for further reduction of the amount of testing needed. It also allows for joining actors or ignoring them if they satisfy certain criteria. If the actors are far enough from the UUT, they will not influence the UUT's decision.

In our future work, we plan to run simulations and tests to evaluate the model presented in this paper as well as present our results from hardware in the look and software in the loop tests.

ACKNOWLEDGMENT

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] S. Singh, "Critical reasons for crashes investigated in the national motor vehicle crash causation survey," NHTSA's Nat. Center Statist. Anal., Washington, DC, USA, Tech. Rep. DOT HS 812 115, 2015.

[2] M. Rasch, P. T. Ubben, T. Most, V. Bayer, and R. Niemeier, "Safety of assessment of automated driver assistance systems," *RDO J.*, vol. 1, no. 2, pp. 27–31, 2019.

[3] R. Razdan, M. I. Akbas, A. Sargolzaei, A. J. Alnaser, S. Sahawneh, S. Alsweiss, and J. Vargas, "Unsettled technology areas in autonomous vehicle test and verification," SAE (Soc. Automot. Eng.) EDGE Res., San Diego, CA, USA, Tech. Rep. EPR2019001, 2019.

[4] S. Hallerbach, Y. Xia, U. Eberle, and F. Koester, "Simulation-based identification of critical scenarios for cooperative and automated vehicles," SAE Warrendale, PA, USA, Tech. Rep. 01-1066, 2018.

[5] A. Heinz, W. Remlinger, and J. Schweiger, "Track-/scenario-based trajectory generation for testing automated driving functions," in *8 Tagung Fahrerassistenz*, 2017.

[6] L. Li, W.-L. Huang, Y. Liu, N.-N. Zheng, and F.-Y. Wang, "Intelligence testing for autonomous vehicles: A new approach," *IEEE Trans. Intell. Vehicles*, vol. 1, no. 2, pp. 158–166, Jun. 2016.

[7] A. Aptiv, M. Baidu, F. C. Continental, H. Automobiles, I. Infineon, and D. Volkswagen, "Safety first for autonomous driving (safad)," Aptiv, White Paper, 2019.

[8] S. Khastgir, S. Birrell, G. Dhadyalla, and P. Jennings., "The science of testing: An automotive perspective," Warrendale, PA, USA, Tech. Rep. 2018-01-1070, 2018.

[9] A. J. Alnaser, M. I. Akbas, A. Sargolzaei, and R. Rahul, "Autonomous vehicles scenario testing framework and model of computation," *SAE Int. J. Connected Automated Vehicles*, to be published.

[10] G. E. Mullins, P. G. Stankiewicz, R. C. Hawthorne, and S. K. Gupta, "Adaptive generation of challenging scenarios for testing and evaluation of autonomous vehicles," *J. Syst. Softw.*, vol. 137, pp. 197–215, Mar. 2018.

[11] A. Gambi, M. Mueller, and G. Fraser, "Automatically testing self-driving cars with search-based procedural content generation," in *Proc. 28th ACM SIGSOFT Int. Symp. Softw. Test. Anal.*, 2019, pp. 318–328.

[12] N. Li, D. W. Oyler, M. Zhang, Y. Yildiz, I. Kolmanovsky, and A. R. Girard, "Game theoretic modeling of driver and vehicle interactions for verification and validation of autonomous vehicle control systems," *IEEE Trans. Control Syst. Technol.*, vol. 26, no. 5, pp. 1782–1797, Sep. 2018.

[13] J. D'Ambrosio, A. Adiththan, E. Ordoukhanian, P. Peranandam, S. Ramesh, A. M. Madni, and P. Sundaram, "An MBSE approach for development of resilient automated automotive systems," *Systems*, vol. 7, no. 1, p. 1, 2019.

[14] C. Medrano-Berumen and M. I. Akbaş, "Scenario generation for validating artificial intelligence based autonomous vehicles," in *Proc. Asian Conf. Intell. Inf. Database Syst.* Phuket, Thailand: Springer, 2020, pp. 481–492.

[15] C. Stark, C. Medrano-Berumen, and M. İ. Akbaş, "Generation of autonomous vehicle validation scenarios using crash data," in *Proc. SoutheastCon*, Mar. 2020, pp. 1–6.

[16] D. Yeh, "Autonomous systems and the challenges in verification, validation, and test," *IEEE Des. Test*, vol. 35, no. 3, pp. 89–97, May 2018.

[17] S. Feng, Y. Feng, C. Yu, Y. Zhang, and H. X. Liu, "Testing scenario library generation for connected and automated vehicles—Part I: Methodology," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 3, pp. 1573–1582, Mar. 2021.

[18] (2018). *Waymo Safety Report: On the Road to Fully Self-Driving*. [Online]. Available: https://storage.googleapis.com/sdc-prod/v1/safety-report/SafetyReport20%18.pdf

[19] J. Langner, J. Bach, L. Ries, S. Otten, M. Holzäpfel, and E. Sax, "Estimating the uniqueness of test scenarios derived fromphrecorded real-world-driving-data using autoencoders," in *Proc. IEEE Intell. Vehicles Symposiumph (IV)*, Jun. 2018, pp. 1860–1866.

[20] M. R. Zofka, S. Klemm, F. Kuhnt, T. Schamm, and J. M. Zöllner, "Testing and validating high level components for automated driving: Simulation framework for traffic scenarios," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2016, pp. 144–150.

[21] B. Kim, Y. Kashiba, S. Dai, and S. Shiraishi, "Testing autonomous vehicle software in the virtual prototyping environment," *IEEE Embedded Syst. Lett.*, vol. 9, no. 1, pp. 5–8, Mar. 2017.

[22] C. E. Tuncali, G. Fainekos, H. Ito, and J. Kapinski, "Simulation-based adversarial test generation for autonomous vehicles with machine learning components," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2018, pp. 1555–1562.

[23] I. R. Jenkins, L. O. Gee, A. Knauss, H. Yin, and J. Schroeder, "Accident scenario generation with recurrent neural networks," in *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2018, pp. 3340–3345.

[24] Y. Tian, K. Pei, S. Jana, and B. Ray, "Deep test: Automated testing of deep-neural-network-driven autonomous cars," in *Proc. 40th Int. Conf. Softw. Eng.*, 2018, pp. 303–314.

[25] N. Kalra and S. M. Paddock, "Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability," *Transp. Res. A, Policy Pract.*, vol. 94, pp. 182–193, Dec. 2016.