Transactions of KSAE, Vol. 32, No. 5, pp.431-441 (May, 2024)

pISSN 1225-6382 / eISSN 2234-0149
DOI http://dx.doi.org/10.7467/KSAE.2024.32.5.431

OpenSCENARIO 기반 자율주행 시나리오 형식 변환 및 정합성 검증 방법론

유 다 연[1)] · 오 태 영[1)] · 유 진 우[*2)]

국민대학교 자동차공학전문대학원[1)] · 국민대학교 자동차IT융합학과[2)]

Scenario Format-Conversion and Consistency-Validation Methodology Based on OpenSCENARIO for Autonomous Driving

Dayeon Yoo[1)] · Taeyoung Oh[1)] · Jinwoo Yoo[*2)]

[1)]*Graduate School of Automotive Engineering, Kookmin University, Seoul 02707, Korea*
[2)]*Department of Automobile and IT Convergence, Kookmin University, Seoul 02707, Korea*
(Received 3 January 2024 / Revised 17 January 2024 / Accepted 23 January 2024)

Abstract : A higher level of autonomous driving is required amid advancing autonomous driving technology. Accordingly, there are concerns about the safety of autonomous vehicles under complex driving environments. Autonomous driving system verification technology through simulation has been attracting attention because verification measures by using actual vehicles are time-consuming and expensive. However, since there are different scenario formats required by various simulation tools, it becomes difficult to achieve productive collaboration with various organizations. Therefore, the ASAM (Association for Standardization of Automation and Measuring Systems) developed OpenX, a scenario production standard. However, it is still not applied to some simulators. To solve this problem, this paper introduces a program that converts a standard scenario into a format that can be applied to simulated conditions. The program was created by reflecting a detailed comparative analysis of the scenario format required by OpenSCENARIO and the simulation tool. Thus, the match rate between the converted scenario format and OpenSCENARIO, the format before conversion, was calculated, proving the appropriateness of the program. By using a standardized framework, collaboration between multiple organizations participating in the development and the verification of autonomous driving technology can be efficiently improved. Likewise, the time and cost of sharing scenario files can be reduced, enabling autonomous vehicles to contribute to commercialization.

Key words : OpenSCENARIO(오픈시나리오), Simulation(시뮬레이션), Autonomous driving(자율주행), ASAM(자동화 및 측정 시스템 표준화 협회), Format-conversion(형식 변환), Consistency-validation(정합성 검증)

1. 서 론

자율주행 기술이 고도화됨에 따라 더 높은 수준의 자율주행이 요구되고 있다. 특히, 국내에서는 정부가 자율주행기술개발혁신 사업을 통해 레벨 4/4+의 자율주행 차량 상용화를 추진하고 있다. 국제 표준[1)]에 의하면 레벨 4/4+ 자율주행 시스템은 시스템 스스로 고장에 대응할 수 있고 도심을 포함한 대부분의 도로에서 자율주행이 가능한 단계를 말한다. 이에 따라, 주변 객체와 도로 인프라 등을 고려한 주행 환경하에서 자율주행 차의 안전성에 대한 우려도 함께 제기되고 있으며 기술이 발전할수록 더욱 복잡한 주행 환경에서의 안전성 검증이 필요한 실정이다. 이때, 실제 차량을 사용한 검증은 시간 및 비용이 많이 들고 위험한 상황을 만들 수 있어[2)] 시뮬레이션을 통한 자율주행 시스템 검증 기술이 주목받고 있다. 자율주행 시뮬레이션 기술은 자율주행 시스템 개발에 소요되는 시험 시간과 비용을 크게 단축시키고 실제 차량 시험 이전 단계에서 알고리즘의 신뢰성을 향상해 시험 안전성을 확보할 수 있다는 점에서 자율주행 시스

*Corresponding author, E-mail: jwyoo@kookmin.ac.kr

템 기술 개발 및 평가 검증에 중요한 역할을 한다.[3] 그러나, 자율주행 시뮬레이션 기술을 사용하는 데 있어서 자동차 제조사, 부품 업체를 포함하여 자율주행 시스템 기술 개발 및 평가 검증에 관여하는 IT 기업, 스타트업, 시뮬레이션 소프트웨어 업체들이 사용하는 도구가 통일되지 않고 검증 시나리오 생성에 있어서도 각기 다른 방식을 취하고 있어 협업에 어려움이 발생한다.[4] 이러한 문제를 해결하기 위해 ASAM(Association for Standardization of Automation and Measuring Systems)에서는 시나리오의 동적 구성 요소를 설명하는 OpenSCENARIO와 정적 구성 요소를 설명하는 OpenDRIVE 등이 포함된 시뮬레이션 표준 OpenX[5]를 개발하였다. 그러나 이러한 표준 문서가 있음에도 불구하고 자율주행에서 널리 사용되는 일부 시뮬레이터에서 OpenSCENARIO가 호환되지 않는다는 문제점이 있다. 이로 인해, 시뮬레이션 소프트웨어마다 다른 형식의 시나리오 파일을 제작해야 할 뿐 아니라 시뮬레이터 간의 시나리오 파일 공유에도 많은 시간과 비용이 소요된다. 이에 본 논문에서는 자율주행 개발을 위한 시뮬레이션 도구에 OpenSCENARIO 형식으로 제작된 시나리오 파일을 적용하여 통일화 된 자율주행 시뮬레이션 검증 환경을 구축하는 방안을 제안한다.

OpenSCENARIO를 시뮬레이터에 적용하기 위해선 OpenSCENARIO 형식으로 작성된 시나리오 파일을 해당 시뮬레이터가 요구하는 파일 형식으로 변환해야 한다. 변환에 앞서, 시뮬레이터마다 사용하는 파일 형식이 모두 다르고 고유한 구조와 특징을 가지고 있기 때문에 이에 대한 세밀한 분석이 필요하다. 본 논문에서는 OpenSCENARIO 기반으로 작성한 시나리오 파일의 구성 인자와 시뮬레이터에서 요구하는 시나리오 파일의 구성 인자를 비교 분석하고 이를 활용하여 변환 프로그램을 제작하는 과정에 대해 설명한다. 비교 분석은 ASAM에서 OpenSCENARIO 사용자를 위해 제공하는 10가지 예제를 대상 시나리오로 선정하여 수행하였다. 비교 분석의 결과를 활용하여 OpenSCENARIO를 CarMaker와 ASM 시뮬레이터에 각각 적용 가능한 형식으로 변환하고 시나리오 참여 객체의 시간에 따른 위치 및 속도를 비교하여 변환 결과의 정합성을 검증하였다.

2. 관련 연구

2.1 자율주행 시뮬레이션

자율주행 시스템은 복잡한 주행 상황을 고려한 검증 절차가 수반되어야 하므로 시뮬레이션을 활용한 검증 방법이 중요한 역할을 한다. 시뮬레이션 검증은 모델, 소프트웨어, 하드웨어에 단계적으로 적용되며 이 검증 프로세스를 통해 실제 차량에서 위험을 동반하는 테스트를 수행할 수 있고 많은 시간이 소요되는 테스트를 자동화하는 등 다양한 작업에 적용될 수 있다. 본 절에서는 자율주행 시스템 개발을 가속화하는 자율주행 시뮬레이션 검증 프로세스에 대해 소개한다.

2.1.1 MILS(Model-in-the-loop Simulation)

PC환경에서 제어 모델의 유효성을 검증하기 위한 방법이다. 제어 모델에 차량 동역학 모델을 적용한 자율주행 시스템 모델을 생성하고 인지 및 판단 기술의 검증을 위한 센서 모델링 단계를 거친다. 그 다음으로 센서 모델과 자율주행 시스템 모델을 통합하여 검증 환경을 구축한다. 앞의 단계를 통해 MILS 검증 환경이 구성되면 검증 시나리오에 대한 시뮬레이션을 수행하고 결과를 분석하는데 이때, 시나리오 파라미터를 자동으로 변경하는 테스트 자동화 기능을 사용해 수만 가지의 시나리오에 대해 시뮬레이션 및 분석 작업을 효율적으로 수행할 수 있다.[6]

2.1.2 SILS(Software-in-the-loop Simulation)

PC환경에서 ECU 단계에서의 소프트웨어를 검증하기 위한 방법이다. 실제 ECU를 사용하진 않지만 ECU 속성을 포함한 시뮬레이션 환경에서 검증이 이루어진다.[7] 메모리 특성, 스케줄링, 변수 범위와 같은 소프트웨어적 속성이 모두 고려된 소스코드를 가상 ECU에서 검증하는 과정을 통해, 실제 ECU에 빌드하기 전 코드 오류를 사전 예방할 수 있다는 이점이 있다.

2.1.3 HILS(Hardware-in-the-loop Simulation)

실제 ECU에서 제어 알고리즘의 성능을 검증하는 방법이다. HILS 기반 검증에서는 ECU 뿐만 아니라 센서, 액추에이터 등의 실제 하드웨어를 시뮬레이션에 포함시켜 실제 하드웨어적 특성이 고려된 자율주행 차량의 제어 시스템 검증이 가능하다. 또한, 최근 차량이 전장화되면서 각 전장 모듈들에 대한 검증이 요구되고 있어 실제 하드웨어를 포함한 HILS 기반 검증이 중요해지고 있다.[8] 개발하고자 하는 시스템을 실제와 가까운 환경에서 시험할 수 있으면서도 실제 환경에서의 시험보다는 비용 절감의 효과가 있다는 장점이 있다.

2.1.4 VILS(Vehicle-in-the-loop Simulation)

실제 차량에 탑승하여 가상 주행 환경에서 실제 차량의 성능을 검증하는 방법이다. 실제 차량을 사용하기 때문에 ECU, 소프트웨어, 동역학적 특성을 고려한 전체 차량의 성능 검증이 가능하다는 장점이 있다. 한편, 일반적인 실제 차량 기반 시험은 보행자 및 주변 차량 등 다양한 외부 요인이 필요하고 충돌의 위험으로 인해 시나리

오 구현에 제약을 갖는다. 반면 VILS 기반 검증 방법은 시뮬레이션 요소를 활용하여 실제 물체를 필요로 하지 않으므로 충돌 위험을 줄일 수 있고 다양한 시나리오의 반복 재현이 가능하다는 이점이 있다.[9)]

2.2 OpenSCENARIO

ASAM OpenX는 자율주행차 검증을 위한 시나리오 제작의 표준으로서 통일화 된 시나리오 제작 방식을 통해 자율주행차 개발 및 검증에 참여하는 업체 간의 효율적인 협력을 제공한다. 표준화된 시나리오 설명 언어를 사용하는 것은 서로 다른 시스템 간 상호 운용성을 확보하여 데이터 공유에 드는 시간 및 비용을 줄일 수 있다는 점에서 중요하다. OpenX는 도로 표면 정보를 포함하는 OpenCRG, 도로 네트워크를 설명하는 OpenDRIVE, 시나리오의 동적 요소를 설명하는 OpenSCENARIO, 시뮬레이션 인터페이스인 OSI, 자율주행 차의 ODD(Operational Design Domain)를 정의하는 OpenODD, 시뮬레이션 핵심 도메인 모델인 OpenXOntology, 객체와 시나리오 주석을 정의하는 OpenLABEL로 구성[10)]되어 있다.

이 중 본 논문에서 핵심적으로 다루는 OpenSCENARIO는 차량, 보행자를 포함한 교통 환경 참여자들의 거동 및 상호 작용을 정의하고 있어 시나리오 설명의 핵심이 되는 부분이다. OpenSCENARIO는 Fig. 1과 같이 7개의 부분으로 구성된다. FileHeader는 버전, 작성자, 작성일 등을 설명한다. ParameterDeclarations은 시나리오에 사용되는 매개변수를 선언하는 부분으로 시나리오의 생성과 관리를 용이하게 한다. CatalogLocations은 시나리오에 사용하는 Catalog의 참조 위치를 지정하는데 이때, Catalog는 재사용이 가능한 시나리오 구성 요소 모음으로 객체 설명과 궤적 등을 포함할 수 있다. RoadNetwork는 시나리오에 사용되는 도로 정보를 정의하며 OpenX의 카테고리 중 도로 정보를 포함하는 OpenDRIVE 파일의 참조 위치를 지정하여 사용할 수 있다. Entities는 시나리오를 구성하는 모든 객체를 정의한다. Storyboard는 객체의 거동과 상호 작용을 설명한다.

2.2.1 Storyboard 구성 요소

Storyboard는 객체의 초기 위치 및 속도를 정의하고 차선 변경, 속도변화, 교통 체증과 같은 동적 컨텐츠를 설명하기 때문에 시나리오를 구성하는데 핵심이 된다.[11)] Storyboard는 크게 시나리오의 초기 상태를 설정하는 Init과 각 객체별로 수행해야 할 거동을 정의하여 주행 상황을 설명하는 Story로 나누어진다. Init은 날씨, 도로 상태 등 시나리오의 환경을 정의하는 GlobalAction과 각 객체별로 초기 위치 및 속도를 나타내는 PrivateAction으로 구성된다. 이때, 객체의 초기 위치는 전역 좌표계를 사용하는 WorldPosition 또는 차량이 위치할 도로의 고유한 Id를 지정하는 LanePosition 등의 다양한 방식을 선택하여 나타낼 수 있다. Story는 여러 객체들이 포함된 복합적인 주행 상황을 설명하기 위해 시나리오 참여 객체 별로 수행해야 할 거동을 설명하는 부분이다. 거동 설명을 위해 Action과 StartTrigger를 사용하는데 Action은 LateralAction 또는 LongitudinalAction 등으로 차량의 종횡방향 움직임을 정의하는 부분이고 StartTrigger는 이러한 움직임의 시작 조건을 설정하는 부분이다. Action과 StartTrigger의 조합을 통해 다양한 종횡방향 거동을 효과적으로 설명할 수 있다. Storyboard는 연속적인 하위 분류 체계를 가지며 Storyboard의 계층적 구조를 Fig. 2에 시각적으로 나타내었다. 실제로는 800여 개의 인자를 포함하고 있지만[5)] 그림에는 핵심 인자만을 선별하여 표현하였다.

2.2.2 대상 시나리오 소개

Storyboard의 구성 요소들을 사용하면 다양한 주행 시나리오를 설명할 수 있다. ASAM에서 제공하는 OpenSCENARIO 형식의 10가지 예제[5)] 파일을 통해 시나리오 제작에서 Storyboard가 어떻게 사용되는지 확인할 수 있다. Fig. 3은 예제 시나리오의 전체 상황도를 제시하고 Storyboard의 구성 요소를 어떤 방식으로 사용해 주행 상황을 표현했는지 분석한 그림이다. 예제 시나리오는 차선 유지 또는 변경, 교통 체증, 선회 구간, 교차로 등을 포함하고 있어 복잡한 도심 주행 환경을 설명하기에 적합하다. 따라

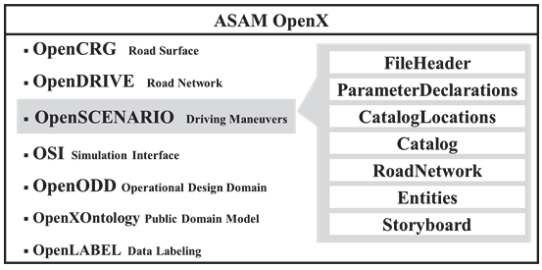

Fig. 1 ASAM OpenX structure and components of OpenSCENARIO

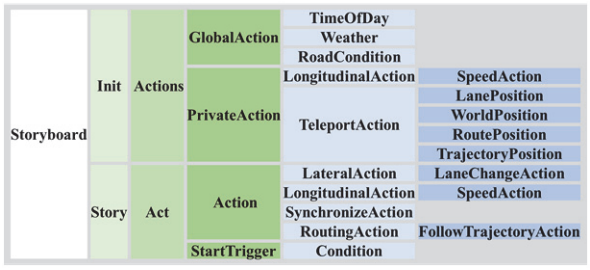

Fig. 2 Storyboard structure of OpenSCENARIO

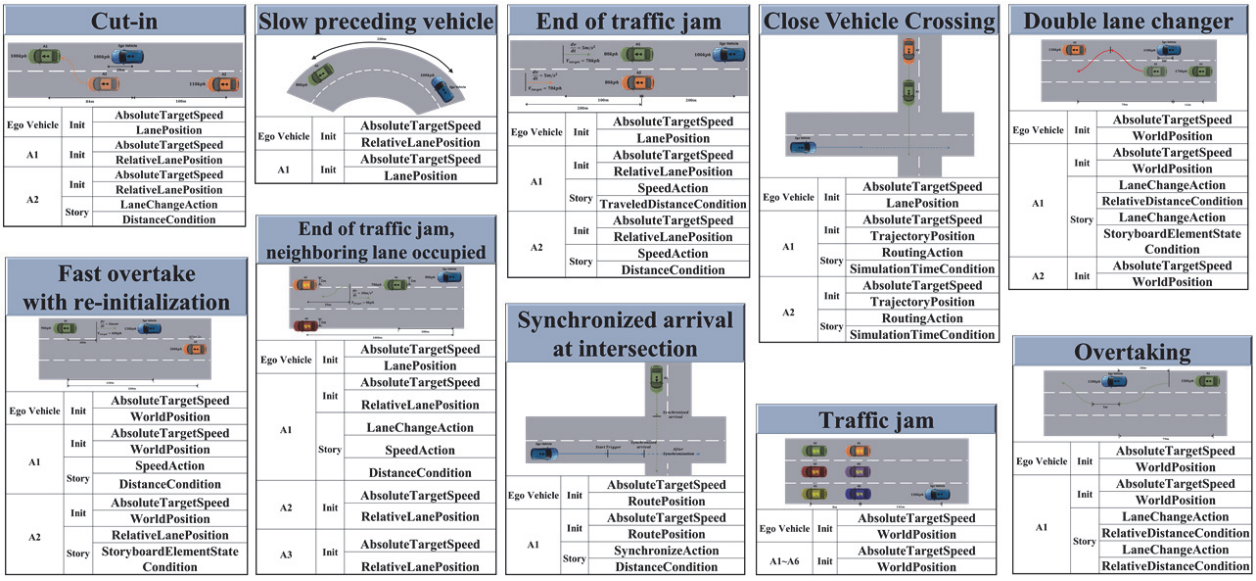

Fig. 3 Schematic diagram of example scenario provided by ASAM

서, 본 논문에서는 이러한 10건의 예제 시나리오를 대상으로 자율주행 시뮬레이션 시스템 환경을 구축하여 높은 수준의 자율주행 시스템 요구 사항에 대응한다.

3. OpenSCENARIO기반 시나리오 형식 변환 방법론

3.1 시나리오 파일 구성 인자 비교 분석

앞 절에서 선정한 대상 시나리오 10건을 분석한 결과, 속도를 설정하는 SpeedAction, 차선 변경의 목표 차로를 지정하는 LaneChangeAction, 객체의 위치를 LaneId 또는 좌표로 지정하는 TeleportAction, 거동의 시작 조건을 설정하는 StartTrigger와 같은 4가지 요소가 공통적으로 사용되는 것을 확인하여 이 4가지를 시나리오 필수 구성 요소로 선정하였다.

각 구성 요소에는 해당 거동을 표현하기 위한 인자들이 포함되어 있다. 예를 들어, 시나리오에서 속도 변화를 설명하고자 한다면 SpeedAction의 AbsoluteTargetSpeed로 원하는 속도 값을 지정하고 SpeedActionDynamics를 통해 속도 변화 양상을 지정하면 된다. 이 필수 구성 요소들을 중심으로 하여 OpenSCENARIO와 시뮬레이션 툴에서 요구하는 시나리오 파일을 비교 분석하면 같은 거동을 나타내는 경우라도 시뮬레이터에 따라 표현 방식이 모두 다른 것을 볼 수 있다. Table 1은 널리 사용되는 시뮬레이션 툴 중 하나인 ASM과 OpenSCENARIO를 비교한 것으로서 같은 거동에 대해 서로 다른 설명 언어를 사용하고 있음을 알 수 있다. 예를 들어, OpenSCENARIO에서는 Ego 차량의 초기 속도를 정의하기 위해

Table 1 Comparison of parameters between OpenSCENARIO and ASM

OpenSCENARIO	**ASM**	
Type	**Ego** Maneuver	**Traffic** Fellow
Init (.Sequence)		
AbsoluteTargetSpeed LanePosition RelativeLanePosition WorldPosition	InitialLongitudinalVelocity InitialLaneIndex StartPosition StartCondition AdditionalLateralOffset	
Story (.Segment)		
RelativeTargetLane SpeedActionDynamics	Activity.LateralType.Activate (“LaneSelection”) Activity.LongitudinalType.Activate (“Velocity”)	
StartTrigger (.Segment.Transition)		
RelativeObjectPostion RelativeDistanceCondition	ConditionDistance ConditionDistanceRefLine	

AbsoluteTargetSpeed를 사용하지만 ASM에서 같은 기능을 구현하기 위해선 Python API로 Maneuver.Sequence.InitialLongitudinalVelocity를 입력해야 한다. Table 2는 OpenSCENARIO와 CarMaker 시뮬레이터를 비교한 것이다. CarMaker는 TestRun이라고 하는 고유의 시나리오 파일을 사용하며 Ego 차량과 주변 차량의 거동을 설명할 때 서로 다른 언어를 사용하는 것이 특징이다. 예를 들

Table 2 Comparison of parameters between OpenSCENARIO and CarMaker

OpenSCENARIO		CarMaker	
Type		**Ego**	**Traffic**
SpeedAction (*m*: Traffic Manuver index, *t*: Traffic index)			
AbsoluteTargetSpeed SpeedActionDynamics		DrivMan.Init.Velocity	Traffic.*t*.Init.v Traffic.*t*.Man.m.LongDyn Traffic.*t*.Man.m. EndCondition
LaneChangeAction			
RelativeTargetLane			Traffic.*t*.Man.m.Limit Traffic.*t*.Man.m.LatDyn
TeleportAction			
LanePosition RelativeLanePosition WorldPosition		Road.VhclStartPos.Kind Road.VhclStartPos DrivMan.Init.LaneOffset	Traffic.*t*.StartPos
StartTrigger			
RelativeObjectPosition			Traffic.*t*.Man.*m-1*.EndCondition
-Rule	Greaterthan		abs(position- EntityRef.position) >Value
	Lessthan		abs(position- EntityRef.position) <Value
RelativeDistanceCondition			Traffic.*t*.Man.*m-1*.EndCondition
-Rule	Greaterthan		abs(Distance-EntityRef.Distance) >Value
	Lessthan		abs(Distance-EntityRef.Distance) <Value

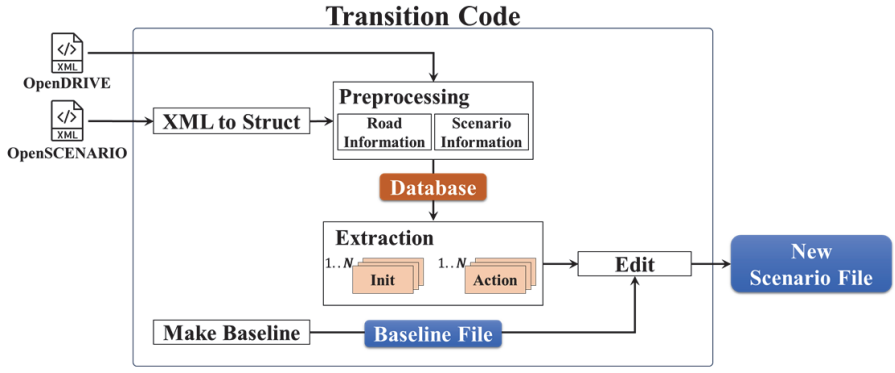

Fig. 4 The overall architecture of the scenario format conversion program

어, OpenSCENARIO에서 Ego 차량 및 주변 객체의 속도는 모두 AbsoluteTargetSpeed로 표현 가능 하지만 CarMaker에서는 Ego의 속도를 나타낼 때 Driv.Man.Init.Velocity를, 주변 차량의 속도를 나타낼 땐 Traffic.Init.v를 사용한다. 또한 거동의 시작 조건인 StartTrigger는 EndCondition을 사용하여 앞선 거동의 종료 조건을 함수 형식으로 작성하는 것으로 대체할 수 있다. 이러한 비교 분석을 기반으로 OpenSCENARIO 형식으로 사용되는 시나리오 설명 언어를 사용하고자 하는 시뮬레이션 도구의 시나리오 파일 형식에 맞게 가공한다. 가공된 인자들은 새로운 형식의 시나리오 파일을 생성하는 데에 사용되며 이러한 과정을 구현한 변환 프로그램에 대해 3.2절에서 설명한다.

3.2 변환 프로그램 구현

3.1절에서 수행한 시나리오 파일 구성 인자 비교 분석에 따라 시뮬레이터에서 요구하는 시나리오 파일에 입력 가능한 형태로 인자를 가공하고 가공된 인자를 활용해 새로운 시나리오 파일을 제작하는 프로그램을 구현한다. Fig. 4는 프로그램의 전체 구성도이다. 변환할

OpenSCENARIO 형식의 시나리오 파일과 시나리오에 사용되는 도로 파일인 OpenDRIVE를 입력하고 전처리를 통해 두 파일로부터 시나리오 데이터베이스를 확보한다. 시나리오 파일에 사용되는 방대한 데이터로부터 변환에 사용할 주요 인자만을 추출하고 Table 1, Table 2를 참조하여 각 시뮬레이터에 사용할 수 있는 형태로 시나리오 구성 인자를 가공한다. 가공된 인자들을 이용해 작성한 새로운 형식의 시나리오 파일을 결과물로 산출한다. 변환 코드는 MATLAB에서 작성되었다.

3.2.1 도로 정보를 활용한 초기 위치 및 변경 차로 지정

변환에 사용할 시나리오 파일의 구성 요소를 한 데 모은 데이터베이스를 구축하기 위해 OpenSCENARIO의 구조적 특성을 활용한다. OpenSCENARIO는 XML형식[12]의 파일인데 이는 데이터를 정의하기 위한 규칙을 제공하는 마크업 언어로써 하나의 태그 안에 여러 하위 태그를 포함하는 계층적 구조를 갖는다. 이러한 구조적 특성을 활용하여 OpenSCENARIO를 구성하는 데이터를 MATLAB에서 쉽게 접근하고 활용할 수 있도록 구조체 형식으로 변환하는 과정(XML to Struct)을 거친다. OpenSCENARIO 구조체에 포함된 요소들을 시뮬레이터에 맞도록 가공해야 하는데 시뮬레이터가 OpenSCENARIO의 모든 구성 인자를 완벽하게 지원하는 것은 아니다. 특히, OpenSCENARIO가 LanePosition을 사용하여 차량의 초기 위치를 차로ID로 정의하는 경우, 일부 시뮬레이터에서 정의하는 차로ID와 일치하지 않기 때문에 지정한 차로에 차량을 배치할 수 없다. 이러한 문제를 해결하기 위해LanePosition을 WorldPosition으로 변환하는 방안을 제시한다. 이는 시뮬레이터와 OpenSCENARIO의 전역 좌표계가 일치한다는 점에서 착안하였으며 시나리오에 사용되는 도로 정보를 포함한 OpenDRIVE 파일을 활용한다.

OpenDRIVE에는 도로 시작점과 끝점 좌표, 도로를 구성하는 차로의 폭, 그리고 도로가 x축으로부터 회전된 정도를 나타내는 각도 등의 정보를 포함한다. 이를 기반으로 도로의 진행 방향을 x축 위에 위치시키는 회전 변환을 수행하면, 도로 중심선의 y좌표가 0이 되므로 1차로 폭의 절반이 1차로 중심선의 y값이 된다. Fig. 5에서 나타내는 것처럼 2차로 중심선의 y값은 1차로 폭과 2차로 폭의 절반을 더하여 구할 수 있다. 같은 방식으로 3차로 중심선의 y값은 1차로 폭과 2차로 폭에 3차로 폭의 절반을 더하여 구한다. 식으로 나타내면 (1)과 같고 이 식을 통해 OpenSCENARIO에서 LaneId로 정의된 차량의 위치를 전역 좌표계로 변환하여 초기 위치를 설정할 수 있다.

$$c_i = \frac{w_i}{2} + \sum_{k=1}^{i-1} w_k \tag{1}$$

c_i : $y\ coordinate\ of\ i_{th}\ lane$

w_i : $Width\ of\ i_{th}\ lane$

OpenSCENARIO에서 정의한 구성 인자 Action 중 LateralAction에 해당하는 LaneChangeAction을 시뮬레이터에 모사하기 위해선 목표 차로의 ID를 알고 있어야 한다. 일례로, Carmaker에서는 차선 변경을 나타내기 위해 현재 차량이 위치한 LaneId로부터 상대적인 값을 목표 차로로 지정한다. 현재 차로의 LaneId가 -2인 경우 목표 차로에 -1을 입력하면 오른쪽으로 한 번 이동하라는 의미로 LaneId가 -3인 곳으로 이동하고, 1을 입력하면 왼쪽으로 한 번 이동하라는 의미로 LaneId가 -1인 곳으로 이동한다. ASM은 차로의 고유한 LaneId를 입력하여 이동할 목표 차로를 설정한다. 따라서, 차량의 위치 정보가 WorldPosition으로만 주어진 경우라면 초기 위치 설정은

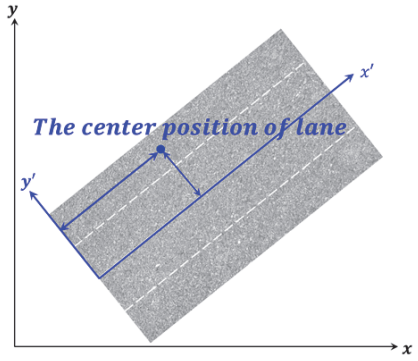

Fig. 5 Transformation from world position to lane position using road rotation matrix

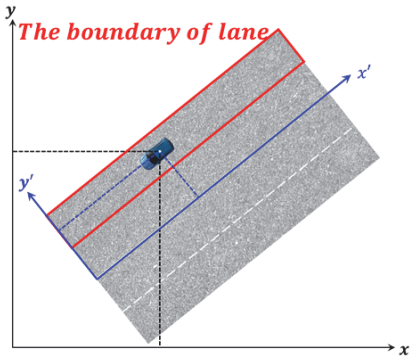

Fig. 6 Transformation from lane position to world position using road bounding box

가능하나 차선 변경 거동을 구현할 수 없는 문제점이 발생한다 따라서, WorldPosition으로 위치가 정의된 경우에는 LanePosition으로 변환하여 해당 좌표가 도로에서 어떤 LaneId를 갖는지에 대한 정보를 추출해야 한다. 이는, 도로의 진행 방향을 x축 위에 위치시키는 회전 변환과 더불어 차량의 좌표도 같은 각도로 회전 변환하여 구할 수 있다. 차로 폭 정보를 알고 있으므로 Fig. 6과 같이 차로 별 경계 박스를 설정할 수 있고 변환된 차량의 좌표가 특정 차로의 경계 박스 내에 존재하는지에 따라 몇 번째 차로 위에 있는지 LaneId를 결정한다. 최종적으로, 전처리를 통해 만들어진 데이터베이스에는 차량의 초기 위치에 대해 WorldPosition과 LanePosition 두 가지 정보를 모두 포함하게 된다.

3.2.2 시나리오 핵심 데이터베이스 생성

3.2.1에서 구조체 형식으로 생성한 OpenSCENARIO 데이터베이스의 구성 요소 중 Action은 차량의 거동을 결정하므로 중요한 역할을 한다. 그러나 이러한 정보는 구조체에서 가장 하위 항목으로 정의되어 있어 이를 직접 사용하면 Action 구성 인자를 불러와 활용하는데 상당한 시간이 소요된다. 따라서, 이러한 문제점을 해결하기 위해 변환에 사용되지 않는 값들은 제외하고 실제 거동에 결정적인 역할을 하는 인자들을 추출해 데이터 베이스를 재구성한다(Extraction). 재구성된 데이터 베이스에는 객체 별 초기 위치 및 속도에 대한 정보를 나타내는 Init과 거동을 나타내는 Action이 포함된다. 이 과정에서 한 개 이상의 객체 및 거동을 모두 파악하기 위해 Fig. 7과 같은 구조의 알고리즘을 활용한다. 모든 객체에 대해 탐색하고 그 안에서도 해당 객체가 수행하는 모든 Action과 각 Action별 Trigger를 탐색하여 간소화된 데이터베이스로 저장한다.

Algorithm 1 : ScenarioFileFormatTransition

Input : OpenSCENARIO, OpenDRIVE
Output : Scenario in the Simulator format

```
Maker Datebase From OpenSCENARIO
OpenSCENARIO ← XMLtoStruct(OpenSCENARIO)
DB ← Preprocessing(OpenSCENARIO,OpenDRIVE)
for i = Actors
  SimpleDB.{Actors}.{Init} ← Extraction(DB.InitVel, DB.InitPos)
  for j = Action
    SimpleDB.{Actors}.{Action} ← Extraction(DB.Action.Type)
    SimpleDB.{Actors}.{Trigger} ← Extraction(DB.Action.Trigger)
  end
end
NewScenarioParameter ← ParameterMatchingCode(SimpleDB)
Edit(NewScenarioParameter,BaselineFile)

return NewScenarioFile
```

Fig. 7 Pseudocode for scenario format transition program

3.2.3 새로운 시나리오 파일 작성

간소화된 데이터 베이스를 기반으로 시나리오 참여자별 Init과 Story 구성 값에 대해 3.1절에서 제시한 비교 분석 표에 따라 시나리오 파일에 입력할 수 있는 형태로 가공한다. 가공된 인자를 시나리오 파일에 작성해야 하는데 이 때, 각 시뮬레이터에서 아무것도 정의하지 않은 시나리오 파일을 생성하고 이를 기본 틀(Make Baseline)로 사용한다. CarMaker는 값이 기입되어 있지 않은 TestRun을 기본 틀로 삼고 ASM은 Python API 문법 구조를 기본 틀로 삼는다. 이러한 기본 틀에 가공을 거친 요소들을 채워 넣음(Edit)으로써 시뮬레이터에서 사용할 수 있는 형태의 새로운 시나리오 파일이 완성된다. TestRun은 파일을 그대로 불러와 시뮬레이터에서 실행할 수 있으며 ASM은 Python API의 내용을 ASM 내부의 Python Interpreter에 입력함으로써 XML형태의 시나리오 파일을 실행할 수 있다.

3.3 OpenSCENARIO기반 시나리오의 변환 정합성 평가

변환 프로그램을 통해 OpenSCENARIO 기반으로 작성된 대상 시나리오를 새로운 시나리오 파일로 변환하고 이를 OpenSCENARIO와 비교함으로써 변환 방법의 적절성을 평가한다. Fig. 8에 대상 시나리오 중 하나인 Cut-in 상황을 OpenSCENARIO 형식으로 나타낸 경우와 변환 프로그램을 통해 TestRun 형식으로 바꾼 후 CarMaker에서 실행한 결과를 나타내었다. Fig. 8(a)은 객체 별 주행 경로를 나타낸 것인데, 이는 Fig. 9을 통해 시각적으로 확인할 수 있다. (a)는 Cut-in 시나리오로 상대 차량이 Ego차량과 선행 차량 사이에 끼어드는 경우이다. (b)는 고속도로 선회 구간에서 Ego차량과 상대 차량이 일정한 속도로 차선 유지 주행하며 달리는 시나리오이다. (c)는 고속도로의 가장 오른쪽 및 왼쪽 차선이 정지된 차량으로 막힌 경우를 나타내며 선행 차량이 우측 차선의 차량과의 충돌을 피하기 위해 중앙 차선으로 차선 변경을 수행하는 시나리오이다. (d)는 Ego 차량이 감속하는 선행 차량에 접근하는 동안 옆 차선에서 다른 차량이 추월하는 상황을 나타낸다. 추가적으로, 같은 방법론을 적용하여 OpenSCENARIO형식의 시나리오 파일을 ASM에 적용 가능한 형식으로 변환하고 이를 MotionDesk에서 실행한 결과는 Fig. 10에 제시하였다. (a)는 Fig. 9(a)와 같은 Cut-in 시나리오이다. (b)는 교차로에서 연속적인 두 차량과 Ego차량이 마주치는 시나리오이다. (c)는 교통체증 상황에서 같은 차선과 옆 차선의 선행 차량들이

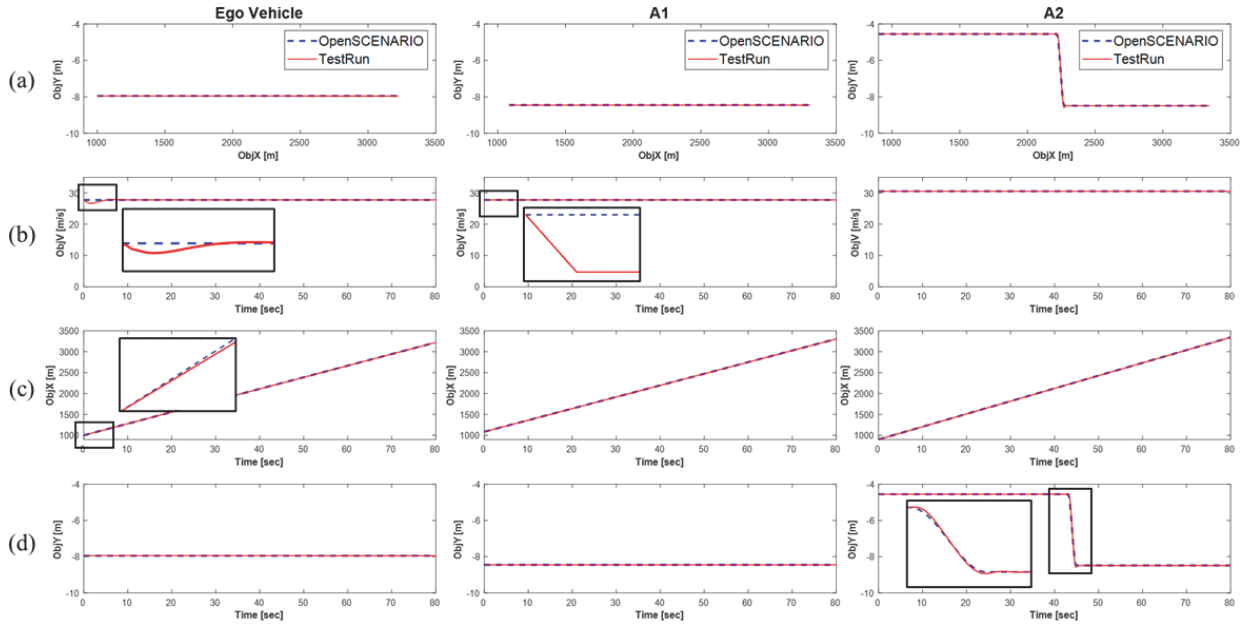

Fig. 8 Cut-in scenario in CarMaker and OpenSCENARIO

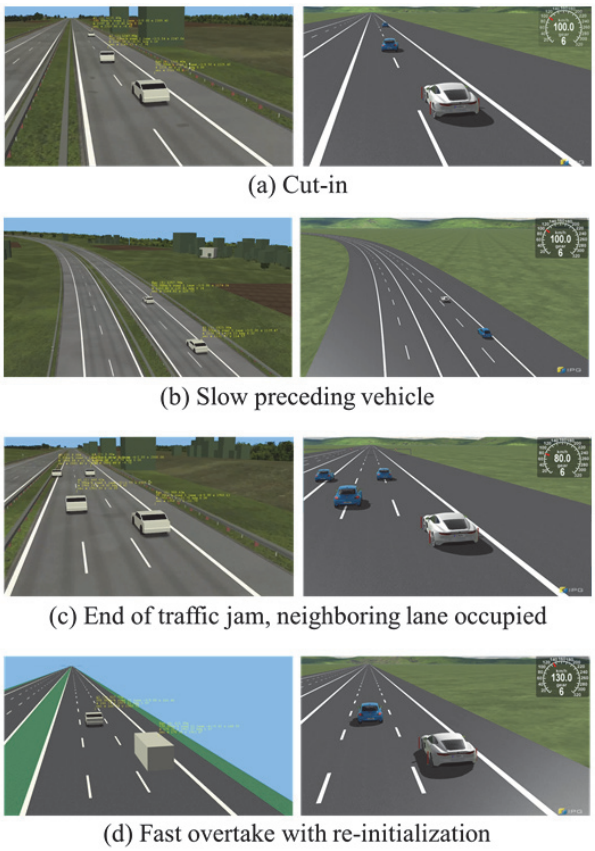

Fig. 9 Simulation visualization (left) OpenSCENARIO by esmini (right) TestRun by CarMaker IPG Movie

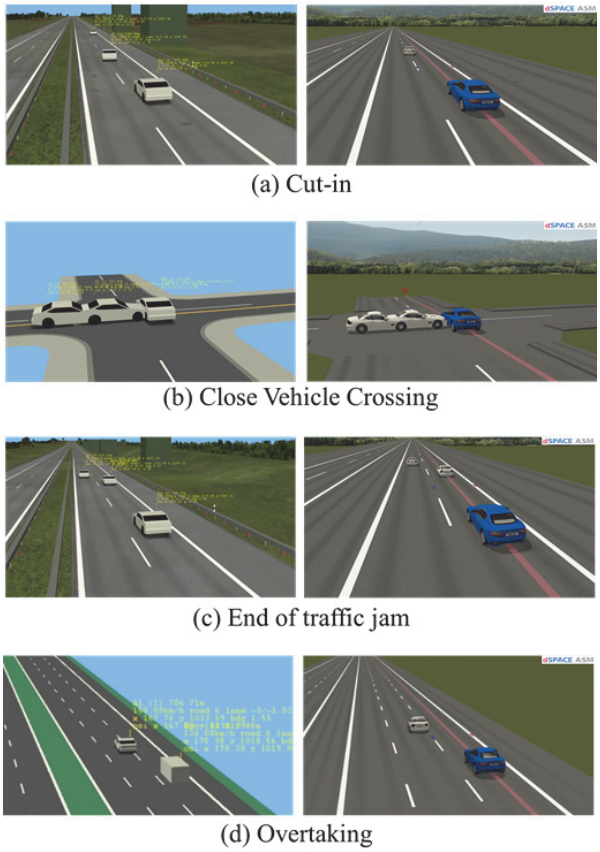

Fig. 10 Simulation visualization (left) OpenSCENARIO by esmini (right) XML Scenario by MotionDesk

감속하는 시나리오이다. (d)는 옆 차선의 차량이 Ego차량을 추월하는 시나리오를 나타낸다. 왼쪽 그림은 변환하기 전 OpenSCENARIO 형식의 시나리오 파일을 전용 뷰어인 Esmini[13]에서 실행한 것이고 오른쪽 그림은 변환 프로그램 결과 생성된 시나리오 파일을 각 시뮬레이터에서 제공하는 뷰어로 실행한 모습이다. 서로 다른 형식을 갖는 시나리오 파일의 객체들이 같은 주행 경로를 그리며 주행하는 것을 확인할 수 있다. Fig. 8(b)은 시간에 따른 속도 그래프로써 확대하여 보면 거동 시작 초기 시점에 차이가 발생하는 것을 알 수 있다. 이는 시간에 따른 x 좌표를 나타낸 Fig. 8(c)의 그림처럼 종방향 위치 오차를 수반하게 되는데, 이는 OpenSCENARIO가 차량 동역학적 특성을 반영하지 않는 것과 달리 시뮬레이터는 동역학적 특성을 반영하기 때문에 발생하는 오차이다. 한편, 차량이 횡방향 거동을 하는 경우에 동작의 시작과 끝 지점은 OpenSCENARIO와 일치하도록 변환 가능하나 과정 중의 거동은 완벽하게 일치시키기 어렵다. 이는 시간에 따른 y좌표를 나타낸 Fig. 8(d)의 마지막 그림을 통해 확인할 수 있다. 그러나, 변환 과정에서 반영되는 이러한 동역학적 특성은 오히려 현실적인 주행 환경을 구현하는 데에 도움이 되므로 자율주행 시뮬레이션 환경의 신뢰성을 높이는 요소로 작용할 수 있다. 이러한 오차를 정량적으로 평가하기 위해 상관성 분석 절차를 수행하였다.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i}^{N}(y_{Open,i}-y_{Sim,i})^2} \quad (2)$$

y_{Sim} : $Simulator$

y_{Open} : $OpenSCENARIO$

OpenSCENARIO와 변환 시나리오 간 시간에 따른 객체 별 위치 및 속도에 대한 두 데이터의 평균 차이를 측정하기 위해 RMSE를 계산하였고, 변환 프로그램이 거동 모사를 적절하게 수행하였는지 평가하기 위하여 종/횡방향 거동을 수행하는 객체의 위치 및 속도 데이터가 변하는 경우, 두 데이터 간 선형 상관 관계를 Pearson 상관 계수[14]로 측정하였다. 식 (2)는 RMSE로 OpenSCENARIO와 변환 시나리오에서 각각 얻은 데이터 간의 차이를 평균한다. 결과는 Table 3에 나타내었다. Table 3에서 시나리오 별로 객체의 위치에 대해 RMSE를 계산한 결과가 평균 2 ~ 5 cm인 것을 확인할 수 있다. 이러한 오차는 국토지리정보원에서 고시하는 정밀도로지도의 위치 정확도[15] 기준인 10 cm 이내의 값으로 허용 오차 범위에 속한다. 속도에 대해서는 평균 0.01 m/s의 오차가 발생하는데 실제 속도 센서의 측정 오차 범위[16]를 고려하였을 때 해당 속도 오차는 매우 작은 값으로 판단된다. 원본 시나리오 파일과 형식이 변환된 시나리오 파일 간의 차이가 허용 오차 범위 내의 작은 값이므로 두 시나리오 파일이 같은 주행 상황을 설명하는 것으로 볼 수 있다. 식 (3)의 r은 Pearson 상관 계수로 두 데이터의 공분산을 각 데이터의 표준 편차의 곱으로 나누어 구하며 -1부터 1사이의 값을 가질 수 있는데 절댓값이 1에 가까울수록 두 데이터 간에 강한 상관 관계가 있음을 의미한다.

$$r = \frac{\sum(y_{Sim,i}-\hat{y}_{Sim,i})(y_{Open,i}-\hat{y}_{Open,i})}{\sqrt{\sum(y_{Sim,i}-\hat{y}_{Sim,i})^2(y_{Open,i}-\hat{y}_{Open,i})^2}} \quad (3)$$

y_{Sim} : $Simulator$

y_{Open} : $OpenSCENARIO$

Table 3 RMSE between OpenSCENARIO and TestRun to evaluate the error

Scenario type	Actor	ObjX[m]	ObjY[m]	ObjV[m/s]
Cut-in	Ego	0.0227	0.0001	0.0063
	A1	0.0049	0	0
	A2	0.0056	0.0006	0.0001
Slow preceding vehicle	Ego	0.0313	0.3199	0.0019
	A1	0.0418	0.2541	0.0000
End of traffic jam	Ego	0.0404	0.0000	0.0126
	A1	0.0555	0.0000	0.0036
	A2	0.0466	0.0000	0.0034
End of traffic jam, Neighboring lane occupied	Ego	0.0410	0.0000	0.0037
	A1	0.0621	0.0075	0.0127
	A2	0.0000	0.0000	0.0000
	A3	0.0000	0.0000	0.0000
Double lane changer	Ego	0.0003	0.0764	0.0235
	A1	0.0057	0.0161	0.0001
	A2	0.0000	0.0113	0.0000
Fast overtake with re-initialization	Ego	0.0011	0.0700	0.0199
	A1	0.0000	0.0555	0.0056
	A2	0.0000	0.1566	0.0000
Overtaking	Ego	0.0074	0.0651	0.0173
	A1	0.0058	0.0116	0.0000
Traffic jam	Ego	0.0100	0.0766	0.0236
	A1~A6	0.0000	0.0000	0.0000
Synchronized arrival at intersection	Ego	0.0130	0.0720	0.0130
	A1	0.0747	0.0123	0.0126
Close Vehicle Crossing	Ego	0.0937	0.0197	0.0092
	A1	0.0299	0.2135	0.0990
	A2	0.0264	0.1470	0.1124
Average		0.0194	0.0496	0.0119

Table 4 Pearson correlation coefficient between OpenSCENARIO and TestRun to evaluate the consistecy

LK=LaneKeeping
LC=LaneChange

Scenario type	Actor	Maneuver		ObjX	ObjY	ObjV
		Longitudinal	Lateral			
Cut-in	A2	Constant	LK → LC	1.0000	1.0000	0.0000
End of traffic jam	A1	Deceleration	LK	1.0000	1.0000	0.9984
	A2	Deceleration	LK	1.0000	1.0000	0.9991
End of traffic jam, Neighboring lane occupied	A1	Deceleration	LK → LC	1.0000	0.9953	0.9995
Double lane changer	A1	Constant	LK → LC	0.9956	1.0000	0.0000
Fast overtake with re-initialization	A1	Deceleration	LK	0.0000	1.0000	0.9997
Overtaking	A1	Constant	LK → LC	0.9964	1.0000	0.0000
Synchronized arrival at intersection	A1	Deceleration	LK	0.9999	0.9999	0.9994
Close Vehicle Crossing	A1	Deceleration	LK	0.9998	1.0000	0.9708
	A2	Deceleration	LK	0.9998	0.9999	0.9631

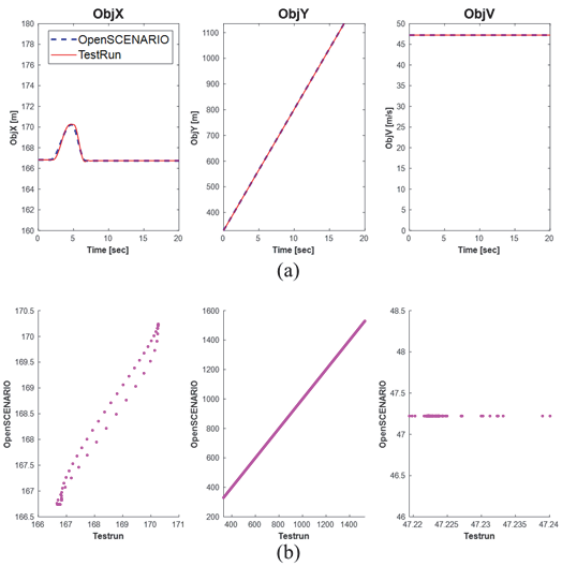

Fig. 11 Double lane changer scenario (a) actor's x,y position and velocity, (b) the correlation between OpenSCENARIO and TestRun

이 때, Pearson 상관 계수는 선형 상관 관계만을 계산하기 때문에 두 데이터가 수평 또는 곡선 관계를 갖는 경우는 0이 될 수 있다. 자세한 설명을 위해 Fig. 11에 'Double Lane Changer'시나리오의 변환결과를 나타내었다. 해당 시나리오에서 객체는 같은 속도로 차선 변경을 두 번 수행한다. 때문에, Fig. 11(a)과 같이 횡방향인 x의 값이 달라지고 시뮬레이션 시간에 따라 y좌표는 일정하게 증가하며 속도는 유지된다. 두 데이터의 상관 관계는 Fig. 11(b)에 나타내었다. 위치 좌표는 선형적인 상관 관계를 가지는 반면 속도에 대해서는 수평 관계를 가지는 것을 확인할 수 있고 이렇게 수평 관계를 갖는 경우 Pearson 상관계수는 0이 된다. 또한, 차량이 정차되어 있어 속도가 변동 없이 0인 경우에 Pearson 식의 분모가 0이 되어 계산되지 않는다. 이러한 점들을 반영하여 Table 4에 형식 변환 전후 시나리오 파일 간 Pearson 상관 계수 계산 결과를 제시하였다. 시나리오 별로 종방향 또는 횡방향 거동을 하는 객체의 위치 및 속도 변화에 대해 Pearson 상관 계수를 계산한 결과, 모든 대상 시나리오에서 Pearson 상관계수가 0.95이상의 값을 갖는다. 이때, Pearson 상관 계수가 0.7보다 큰 경우는 강한 양의 상관 관계를 가진 것으로 분류[17]되므로 두 데이터는 강한 양의 상관 관계를 갖는 것으로 볼 수 있다. 이는 본 논문에서 제안하는 변환 방법이 원본 시나리오의 객체 거동 모사에 대한 내용을 훼손하지 않으면서 형식만을 적절히 변환하였음을 의미한다.

4. 결 론

본 논문에서는 표준 시나리오 형식인 OpenSCENARIO를 시뮬레이션 소프트웨어에 적용하는 방법론을 제안하고 방법의 적절성을 평가하기 위해 정합성을 검증을 수행하였다. 차량 동역학적 요소가 고려된 시뮬레이터에 표준 문서를 적용하는 통합 검증 시스템은 보다 현실적인 자율주행 검증 환경을 구현할 수 있고 표준 문서를 활

용함으로써 자율주행 시뮬레이션 시스템의 범용성을 확보할 수 있다. 이러한 범용적인 자율주행 시뮬레이션 인터페이스는 자율주행 기술 개발 및 검증에 참여하는 기업/기관들의 협업 과정을 효율적으로 개선할 수 있다. 향후 추가 연구를 통해 OpenSCENARIO의 구성 인자 중 운전자 특성 및 기상 상황 등을 반영한 고도화된 변환 프로그램을 개발한다면 실제 주행 상황에 더욱 가까운 시뮬레이션 검증 환경을 구축할 수 있을 것으로 기대된다.

후 기

이 논문은 2024년도 정부(산업통상자원부)의 재원으로 한국산업기술진흥원의 지원을 받아 수행된 연구임 (P0020536, 2024년 산업혁신인재성장지원사업).

References

1) Society of Automotive Engineers(SAE), Taxonomy and Definitions for Terms Related to On-Road Motor Vehicle Automated Driving Systems, J3016_202104, 2021.
2) M. J. Kim, S. H. Yu, T. H. Kim, J. U. Kim and Y. M. Kim, “On the Development of Autonomous Vehicle Safety Distance by an RSS Model Based on a Variable Focus Function Camera,” Sensors, Vol.21, No.20, Paper No.6733, 2021.
3) P. Kaur, S. Taghavi, Z. Tian and W. Shi, “A Survey on Simulators for Testing Self-Driving,” IEEE Fourth International Conference on Connected and Autonomous Driving(MetroCAD), pp.62-70, 2021.
4) C. Pilz, G. Steinbauer, M. Schratter and D. Watzenig, “Development of a Scenario Simulation Platform to Support Autonomous Driving Verification,” IEEE International Conference on Connected Vehicles and Expo(ICCVE), pp.1-7, 2019.
5) ASAM, ASAM Standardization, https://www.asam.net/standards/, 2021.
6) H. Cho, “Development and Validation of Autonomous Driving Vehicle Technology Based on MiLS,” Auto Journal, KSAE, Vol.37, No.10, pp.33-40, 2015.
7) J. Lee, H. Lee and J. Son, “SiLS-Based AUTOSAR SW Validation Technology by Utilizing Virtual ECU,” Auto Journal, KSAE, Vol.37, No.10, pp.41-45, 2015.
8) D. S. Kim, “ADAS(Advanced Driver Assistance Systems) Validation Trend,” Auto Journal, KSAE, Vol.43, No.6, pp.35-38, 2021.
9) W. Son, Y. Ha, T. Oh, S. Woo, S. Cho and J. Yoo, “PG-Based Vehicle-In-the-Loop Simulation for System Development and Consistency Validation,” Electronics, Vol.11, No.24, Paper No.4073, 2022.
10) H. Ren, H. Gao, H. Chen and G. Liu, “A Survey of Autonomous Driving Scenarios and Scenario Databases,” IEEE 9th International Conference on Dependable Systems and Their Applications(DSA), pp.754-762, 2022.
11) Y. Zhang, M. Wang, B. Zhou, X. Hu and D. Zhang, “Design and Implementation of DSpace Using OpenSCENARIO to Construct Scenarios in Vehicle Simulation Testing,” International Conference on Artificial Intelligence, Information Processing and Cloud Computing(AIIPCC), pp.20-23, 2022.
12) E. R. Harold and W. S. Means, XML in a Nutshell: A Desktop Quick Reference, O’Reilly Media, California, 2004.
13) esmini, Environment Simulator Minimalistic, https://github.com/esmini, 2019.
14) R. Donà, S. Vass, K. Mattas, M. C. Galassi and B. Ciuffo, “Virtual Testing in Automated Driving Systems Certification. A Longitudinal Dynamics Validation Example,” IEEE Access, Vol.10, pp.47661-47672, 2022.
15) National Geographic Information Institute, Quality Verification Manual for High-Definition Road Map, 2020.
16) C. S Kim and K. S. Huh, “Development of a Pose and Position Estimation Algorithm with Considering the Error and Latency from the GPS and In-Vehicle Sensors,” Transactions of KSAE, Vol.26, No.5, pp620-629, 2018.
17) D. Nettleton, Commercial Data Mining: Processing, Analysis and Modeling for Predictive Analytics Projects, Elsevier, Amsterdam, 2014.