# SPLAD: 자율주행을 위한 스트림 프로세싱 언어

양범준*1) · 노순현1) · 양석우1) · 안재호1) · 홍성수1)2)
서울대학교 전기정보공학부1) · 서울대학교 융합과학기술대학원 융합과학부2)

# SPLAD: Stream Processing Language for Autonomous Driving

Beomjoon Yang*1) · Soonhyun Noh1) · Seokwoo Yang1) · Jaeho Ahn1) · Seongsoo Hong1)2)

*1) *Department of Electrical and Computer Engineering, SNU, 1, Gwanak-ro, Gwanak-gu, Seoul, Republic of Korea*

2) *Department of Transdisciplinary Studies, GSCST, SNU, 1, Gwanak-ro, Gwanak-gu, Seoul, Republic of Korea*

**Abstract** : Autonomous vehicles must satisfy non-functional requirements such as real-time stream processing and reliability. In order to satisfy these requirements, we propose a new programming language named SPLAD (Stream Processing Language for Autonomous Driving). SPLAD is a graphical programming language that describes the processing of sensor stream data in the form of a data flow graph so that the program flow can be easily grasped. In addition, timing constraints can be specified on the data flow graph to satisfy the real time requirements. Developers can also define an exception and specify its handling in order to guarantee reliability. In this paper, we present the language semantics and language constructs of SPLAD and explain them in detail through an example.

**Key words**: Stream processing(스트림 처리), autonomous driving(자율주행), programming language(프로그래밍 언어), real-time system(실시간 시스템), reliability(신뢰성), sensor fusion(센서 퓨전), exception handling(예외 처리)

## 1. 서 론

현재 시장을 선도하는 자동차/IT 기업들이 자율주행차량 개발에 박차를 가하고 있다. 대표적으로 국외에서는 Audi, Waymo 등이 2020년까지 자율주행차량의 상용화를 목표로 하고 있으며, 국내에서는 현대자동차, 네이버 등이 자율주행 기술개발을 위한 임시운행 허가를 받아 개발을 추진하고 있다 [1]. 하지만 현재까지 개발된 대부분의 자율주행차량들은 기본적인 기능적 요구사항만을 만족시키는데 그치고 있다.

자율주행차량의 상용화를 위해서는 주행 과정에서 요구되는 비기능적 요구사항들을 만족시킬 수 있어야 한다. 대표적인 자율주행의 비기능적 요구사항에는 (1) 실시간성과 (2) 신뢰성이 있다. 실시간성을 만족시키기 위하여 자율주행의 인지, 판단, 제어 과정에서 반드시 시간제약(timing constraint)을 지켜야 한다. 또한 신뢰성을 만족시키기 위하여 자율주행차량은 도로주행 환경에서 발생할 수 있는 여러가지 예외상황에서도 정상 동작을 보장할 수 있어야 한다.

현재 자율주행차량을 개발하고 있는 대부분의 기업과 연구소들은 이러한 비기능적 요구사항의 충족을 세부 알고리즘 개발자들에게 전적으로 맡기고 있다. 각 세부 알고리즘 개발자들은 반복적인 튜닝을 통해 알고리즘을 수정하여 비기능적 요구사항을 만족시킨다. 그러나 이러한 접근방법은 개발 과정의 비효율성을 초래하고, 튜닝 과정에서 간과한 예

* Beomjoon Yang, E-mail: bjyang@redwood.snu.ac.kr

외상황에서 치명적인 인명 피해를 발생시킬 수 있다.

그러므로 자율주행 응용을 개발하기 위해서는 이러한 비기능적 요구사항들을 명시할 수 있는 프로그래밍 언어가 반드시 필요하다. 이 프로그래밍 언어는 기본적으로 센서 스트림 데이터의 처리 과정을 효과적으로 표현할 수 있어야 하고, 동시에 자율주행의 신뢰성과 실시간성 요구사항을 명확하게 기술할 수 있어야 한다.

하지만 자율주행 응용을 위한 stream processing language에 관한 연구는 현재 매우 미비하다. 현존하는 stream processing language들은 빅데이터 분석을 위해 개발되었기 때문에 실시간성과 신뢰성 요구사항을 명시할 수 없다[2][3][4]. 또한, 이런 언어들은 스트림 데이터의 흐름을 직관적으로 파악할 수 있는 필수적인 기능도 제공하지 않는다[3][4].

본 논문에서는 이러한 한계를 극복하기 위해 Stream Processing Language for Autonomous Driving(SPLAD)를 제안한다. SPLAD는 시각적 프로그래밍 언어로, data flow graph를 이용하여 자율주행 응용을 기술할 수 있도록 한다. 이를 통하여 자율주행차량의 센서 데이터 처리를 시각적을 표현하여 데이터의 흐름을 쉽게 파악할 수 있도록 한다.

SPLAD는 실시간성을 만족시키기 위하여 data flow graph 상에서 timing constraint를 명시할 수 있도록 한다. 또한 신뢰성을 보장하기 위하여 개발자가 예외상황을 정의하고, 각각의 예외상황에 대한 처리를 명시할 수 있게 한다.

본 논문의 나머지 장은 다음과 같이 구성되어 있다. 2장에서는 설계된 SPLAD의 language constructs와 그 semantics를 설명한다. 3장에서는 SPLAD를 이용하여 자율주행 응용을 작성하는 방법을 예제를 통해 제시하여 SPLAD의 유용성을 보인다. 4장에서는 SPLAD의 추후 개발 사항을 다룬다.

## 2. Language Semantics and Language Constructs of SPLAD

본 장에서는 자율주행차량에서 수행되는 실시간 스트림 처리 응용 개발을 위한 프로그래밍 언어인 SPLAD를 설명한다. 2.1절에서는 자율주행 응용들의 기본적인 동작과 요구사항들을 명확하게 표현하기 위한 language semantics를 정의한다. 그 후 2.2절에서는 정의한 language semantics에 mapping되는 SPLAD의 language construct들을 제안한다. 2.3절에서는 개발자가 정의한 예외상황에서 프로그램의 비정상적 동작을 막기위한 예외 처리(exception handling)에 대해 설명한다.

### 2.1 Language Semantics

자율주행에서 요구되는 기본적인 스트림 처리를 지원하기 위해서는 스트림 처리의 basic operational semantics를 명시할 수 있어야 한다. Basic operational semantics는 (1) data processing semantics, (2) path control semantics, (3) triggering semantics로 분류할 수 있다. Data processing semantics는 연속적으로 입력되는 스트림 데이터에 대해서 개발자가 원하는 연산을 수행할 수 있게 자유롭게 프로그래밍이 가능함을 의미한다. Path control semantics는 data flow graph 상에서 스트림 데이터의 진행 경로를 개발자가 의도한 방향으로 제어할 수 있음을 의미한다. 이때 data flow graph는 스트림 데이터의 흐름을 표현한 directed graph이다. 마지막으로, triggering semantics는 스트림 처리의 시작 시점을 개발자가 의도한 시점으로 결정 가능함을 의미한다.

자율주행 응용들의 실시간 스트림 처리를 위해서는 응용에 요구되는 timing constraint들을 프로그래밍 언어를 이용하여 명시할 수 있어야 한다. 이를 timing semantics라 부른다. 명시해야 하는 timing constraint들에는 (1) deadline, (2) input/output minimum rate constraint, (3) freshness constraint, (4) correlation constraint가 있다. Deadline은 입력 스트림 데이터가 들어올 때부터 최종 결과가 출력될 때까지의 제한시간이다. Input/output minimum rate constraint는 단위시간당 응용이 입력 받아야 하는 스트림 데이터와 출력해야 하는 스트림 데이터의 최소 개수이다. Freshness constraint는 스트림 데이터가 유효한 시간으로, 스트림 데이터가 생성되어 입력으로 들어오는 시점부터 처리를 시작하는 시점까지의 제한시간으로 정의된다. Correlation

constraint는 함께 처리 되어야하는 두 개 이상의 스트림 데이터 간에 생성 시간 차이의 최대값으로 정의된다.

마지막으로, 자율주행차량에서 수행되는 응용들은 주행 중 예외상황(exception)이 발생하였을 때에도 중단 없이 정상 동작을 수행하여야 한다. 우리는 예외상황을 앞서 언급한 basic operational semantics나 timing semantics가 지켜지지 않는 상황으로 정의된다. 예외상황은 (1) timing violation exception, (2) invalid data exception, (3) absent data exception으로 구분된다. 예외상황에서 정상 동작을 보장하기 위해서는 각 예외상황마다 그에 맞는 예외 처리 알고리즘이 수행되어야 하는데, 이를 exception handling이라고 부른다.

첫 째, timing violation exception은 명시한 timing constraint가 위반되는 경우를 말한다. Deadline violation은 스트림 데이터의 처리가 시작되는 시점부터 처리를 완료되는 시점까지의 시간이 deadline을 초과할 경우를 말한다. Input/output minimum rate constraint violation은 단위시간당 입출력 스트림 데이터의 개수가 input/output minimum rate constraint 미만일 경우이다. Freshness constraint violation은 스트림 데이터의 처리를 시작하는 시점에 freshness constraint를 위반한 상황이다. Correlation constraint violation은 함께 처리되는 입력 스트림 데이터들이 correlation constraint 위반하는 경우이다. 둘 째, invalid data exception는 입력된 데이터의 값이 유효한 범위를 벗어나서 알고리즘이 정상 동작 할 수 없는 경우를 말한다. 입력 데이터가 스트림 데이터의 경우 이를 invalid stream data input이라 하고, 이벤트 데이터일 경우 invalid event data input이라 한다. 여기서 이벤트 데이터란 자율주행 응용의 상태를 변환시키는 사건인 이벤트의 발생을 전달하기 위한 단발성의 데이터이다. 셋째, absent data exception은 스트림 처리의 시작 시점에 처리할 입력 스트림 데이터가 없는 경우를 말한다.

### 2.2 Language Constructs

이 절에서는 SPLAD의 핵심 language construct들을 설명한다. 제안하는 language construct들은 2.1절에서 언급한 자율주행차량의 실시간 스트림 처리과정에서 필요한 semantics를 효과적으로 표현할 수 있도록 설계되었다. 우리가 제안하는 SPLAD의 language construct들을 Table 1에 정리하였다. SPLAD 프로그램은 이 language construct들을 조합한 data flow graph로 표현되는데, 이 때, processing cell, merger, selector는 node가 되고, pipe는 edge가 된다. 각 language construct가 표현할 수 있는 basic operational semantics와 timing semantics는 Table 2에 정리되었다.

#### 2.2.1 Processing Cell

Processing cell은 스트림 처리의 최소 수행 단위이다. Processing cell은 하나의 스트림 데이터를 입력으로 받아 개발자가 정의한 연산을 수행한다. 수행이 끝나면 결과를 스트림 데이터 또는 이벤트 데이터를 출력한다.

Processing cell은 연산을 시작하는 triggering 조건에 따라 (1) data-driven과 (2) time-driven의 두가지 형태로 분류된다. Data-driven processing cell은 스트림 데이터가 processing cell에 입력되는 즉시 triggering된다. Time-driven processing cell은 일정한 시간 주기로 triggering된다. Processing cell의 표기는 Fig. 1와 같다.

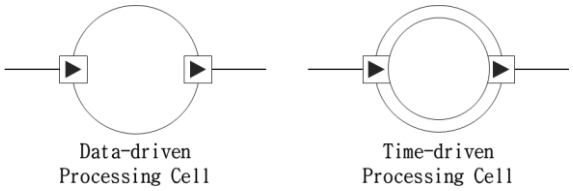



Fig. 1 Processing cell의 표기

또한, 실시간 스트림 처리과정에서 요구되는 시간 제약을 만족시키기 위해 각 processing cell에 deadline을 설정할 수 있다.

#### 2.2.2 Port

Port는 스트림 데이터와 이벤트 데이터를 위한 입출력 인터페이스이다. Processing cell, merger, selector와 factory는 port를 사용하여 스트림 데이터와 이벤트 데이터를 입력 받거나 출력할 수 있다. Port의 종류와 표기를 Table 3에 표시한다. 개

발자는 각 port에 실시간 스트림 데이터의 전달 과정에서 요구되는 input/output minimum rate constraint와 freshness constraint를 명시할 수 있다.

Table 1 Language construct의 정의

| Language construct | 정의 |
|---|---|
| Processing cell | 스트림 처리의 최소 수행 단위 |
| Port | 스트림 데이터와 이벤트 데이터가 입출력 되는 인터페이스 |
| Pipe | 스트림 데이터와 이벤트 데이터의 전달 경로 |
| Merger | 여러 개의 입력 port로 들어온 각각의 스트림 데이터를 하나로 합쳐서 출력 port로 내보내는 operator |
| Selector | 하나의 입력 port로 들어온 스트림 데이터를 선택한 출력 port로 내보내는 operator |
| Factory | 스트림 처리의 수행 단위 |

Table 2 Language construct가 만족시키는 semantics

| Language construct | Basic operational semantics | | | Timing semantics | | | |
|---|---|---|---|---|---|---|---|
| | Data processing semantics | Path control semantics | Triggering semantics | Deadline | Input/output minimum rate constraint | Freshness constraint | Correlation constraint |
| Processing cell | ○ | | ○ | ○ | | | |
| Port | | | | | ○ | ○ | |
| Pipe | | ○ | | | | | |
| Merger | | | | | | | ○ |
| Selector | | ○ | | | | | |
| Factory | | ○ | | ○ | | | |

Table 3 Port의 종류와 표기

| 이름 | 설명 | 표기 | | 연결되는 language construct |
|---|---|---|---|---|
| | | 입력 | 출력 | |
| Data port | Factory 내의 스트림 입출력을 담당 | ▶ | ▶ | Processing cell<br>Pipe<br>Merger<br>Selector |
| Factory port | Factory 간의 스트림 입출력을 담당 | ▷ | ▷ | Factory<br>Pipe |
| Event port | Processing cell의 이벤트 데이터의 출력과 selector의 출력 분기 조건에 사용되는 이벤트 데이터의 입력을 담당 | ∨ | ∧ | Processing cell<br>Pipe<br>Selector |
| Mode change port | Factory의 mode 전환에 사용되는 이벤트 데이터의 입력을 담당 | ∴ | ∧ | Factory<br>Pipe |

### 2.2.3 Pipe

Pipe는 스트림 데이터와 이벤트 데이터의 전달 경로이다. 개발자는 pipe를 사용하여 한 language construct의 출력 port를 다른 language construct의 입력 port와 연결시킬 수 있다. 하나의 pipe는 한 종류의 데이터만 전달할 수 있다. 또, 입력 경로는 하나이지만 출력 경로는 여러 개인 pipe를 그릴 수도 있다. Pipe에는 timing constraint가 명시되지 않으나 하나의 pipe에 연결된 port들의 input/output minimum rate constraint들은 모두 같아야 한다. Pipe의 표기는 Fig. 2와 같다.

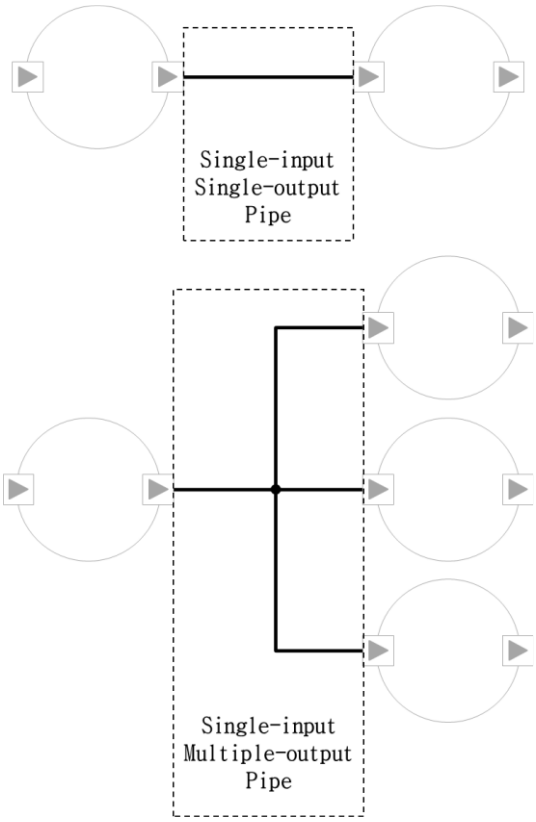



Fig. 2 Pipe의 표기

### 2.2.4 Merger

Merger는 여러 개의 입력 data port로 들어온 스트림 데이터들을 하나로 합쳐서 출력 data port로 내보내는 operator이다. Merger는 두 개 이상의 입력 data port와 한 개의 출력 data port를 가진다. SPLAD 프로그램에서 두 개 이상의 스트림 데이터를 processing cell의 입력으로 사용하고 싶을 때 merger를 사용할 수 있다. Merger의 표기는 Fig. 3과 같다.

Merger를 trigger 시키는 입력 스트림 데이터는 반드시 필요한 mandatory, 선택적으로 필요한 optional로 분류된다. Mandatory 입력의 port는 mandatory part에, optional 입력의 port는 optional part에 위치한다. Merger가 trigger되기 위해서는 다음 두 조건이 동시에 만족되어야 한다.

(1) Mandatory part에 속하는 모든 입력 port에 스트림 데이터가 존재하여 모든 입력 port가 활성화되어야 한다.

(2) Optional part에 속하는 입력 port가 특정 숫자 이상이 활성화되어야 한다.

Merger는 활성화된 입력 port에서 correlation constraint를 만족시키는 입력 스트림 데이터들의 조합을 찾은 경우에 한해 실제로 수행된다. 만약 이 조합이 존재하지 않는다면, 스트림 데이터를 합치는 작업은 다음 triggering 시점까지 연기된다.

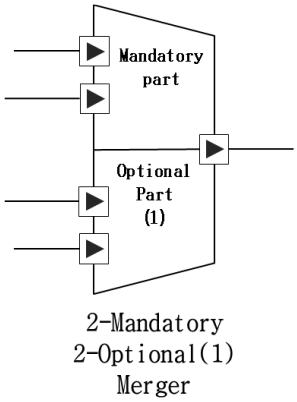
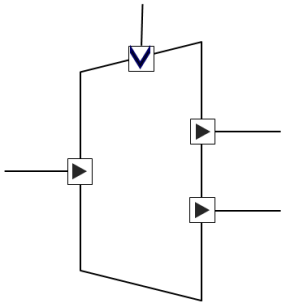



Fig. 3 Merger의 표기

### 2.2.5 Selector

Selector는 Fig. 4와 같이 하나의 입력 data port, 하나의 입력 event port, 다수의 출력 data port로 구성된다. Selector는 입력으로 들어온 스트림 데이터를 선택된 특정 출력 port로 내보내는 operator이다. 이때 출력 port는 입력 이벤트 데이터의 값에 따라 선택된다. 물론 여러 개의 출력 port를 선택될 수도 있다.



2-Selector

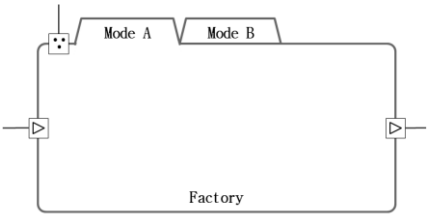
Fig. 4 Selector의 표기

#### 2.2.6 Factory

Factory는 스트림 처리의 가장 큰 구성단위이다. Factory 내부에는 SPLAD language construct들을 조합하여 만들어진 data flow graph가 위치하는데, 이것이 응용의 동작을 기술한다. Factory의 표기는 Fig. 5와 같다.

Factory는 조건에 따른 선택적 알고리즘 수행을 지원하기 위해 mode change 기능을 지원한다. Mode란 하나의 factory 안에서 배타적으로 수행되는 data flow graph이다. 한 factory의 mode는 mode change port의 입력 이벤트 데이터의 값에 따라 결정된다.

Factory에는 processing cell과 마찬가지로 deadline이 부여될 수 있다. Processing cell이 deadline 안에 입력을 처리하여 그 결과를 스트림 데이터로 출력하여야 하듯이, Factory도 deadline 안에 입력을 내부의 processing cell들에 의해 순차적으로 처리하여 출력으로 만들어야 한다.




2-Mode Factory

Fig. 5 Factory의 표기

### 2.3 Exception Handling

이 장에서는 SPLAD의 exception handling에 대해 설명한다. Exception handling을 위해서는 2.1절에서 명시한 예외상황들을 처리할 수 있도록 각각의 exception handler가 필요하다. Exception handler는 SPLAD에서 미리 정의된 것과 개발자에 의하여 작성되는 것으로 나눌 수 있다. 전자에는 correlation constraint violation과 freshness violation에 대한 exception handling이 포함된다. 후자는, exception handler 함수의 argument로 예외 상황을 발생시킨 스트림 데이터 혹은 이벤트 데이터가 주어지며, 응용 개발자가 함수 내부의 코드를 작성해야 한다.

Table 4은 각 예외상황을 처리하는 language construct들을 정리한 것이다.

## 3. SPLAD Program Example: Sensor Fusion 기반 Object Detection 알고리즘

본 장에서는 SPLAD를 사용하여 자율주행차량에서 수행되는 실시간 스트림 처리 응용을 작성하는 예제를 소개한다. 우리는 자율주행의 상황 인지의 대표적인 알고리즘인 sensor-fusion 기반 object detection 알고리즘을 예제로 선택하였다[5]. 이 알고리즘은 RGB 카메라 데이터와 LiDAR 센서 데이터를 동시에 사용하여 차량 주위에 존재하는 물체들의 종류와 위치, 크기를 인식한다. SPLAD language construct들을 모두 보이기 위해, 우리는 [5]에서 RGB 카메라 이미지를 처리하는 알고리즘을 일부 수정하였다. 3.1절에서는 자율주행 알고리즘의 기본동작을 SPLAD로 표현하기 위한 방법과 생성된 예제 SPLAD 프로그램을 설명한다. 3.2절에서는 timing constraint, exception handling을 표현하는 방법과 예제 SPLAD 프로그램에 이를 적용한 것을 설명한다.

Table 4 Exception handling을 담당하는 language construct

| Exception | Factory | Processing Cell | Merger | Selector | Port |
|---|---|---|---|---|---|
| Deadline | ○ | ○ | | | |
| Input/output minimum rate constraint | | | | | ○ |
| Freshness constraint | | | | | ○ |
| Correlation constraint | | | ○ | | |
| Invalid stream data input | | ○ | | | |
| Invalid event data input | ○ | | | ○ | |
| Absent data exception | | ○ | | | |

### 3.1 Language Construct의 생성

이 절에서는 자율주행 알고리즘의 전체 동작을 language construct로 표현하는 방법을 설명한다. 먼저, 알고리즘의 명세를 위해 intermediate representation graph를 그린다. 이때 intermediate representation graph는 알고리즘의 세부 함수를 node로, 함수들이 스트림 데이터를 주고 받는 관계를 edge로 표현한 directed graph 이다. 다음으로, 작성된 intermediate representation graph를 기반으로 SPLAD의 language construct들로 이루어진 data flow graph를 그린다. 세부 과정을 아래에 단계별로 표시하였다. (1)~(4)는 intermediate representation graph를 그리는 과정을 설명하고, (5)~(7)은 이 결과를 기반으로 SPLAD의 data flow graph를 그리는 과정을 설명한다.

전체 알고리즘의 입력 스트림 데이터와 출력 스트림 데이터를 명시한다.

(1) 알고리즘을 단일 기능을 수행하는 함수들로 나눈다.
(2) 나누어진 각 함수 별로 스트림 데이터의 입출력을 명시한다.
(3) Intermediate representation graph를 그린다. (2)의 함수들이 node가 되며, 함수 간 스트림 데이터의 입출력 관계가 edge에 대응된다.
(4) 하나의 factory를 그리고 (1)에서 명시한 입출력 스트림 데이터들을 받을 수 있는 factory port를 그린다.
(5) Intermediate representation graph의 node들을 각각 하나의 processing cell에 mapping한다. 이때, 해당 함수의 triggering 방식에 따라 processing cell을 그린다.
(6) Intermediate representation graph의 edge들을 pipe로 mapping한다. Processing cell의 입력이 2개 이상이 된다면, 하나로 합치는 merger를 추가한다. 조건부 출력 분기가 있다면 selector를 추가한다.

예제 알고리즘에 대한 (1), (2), (3) 과정의 결과를 Table 5로, (4)의 결과를 Fig. 6 정리하였다. Fig. 7은 (5), (6), (7) 과정의 결과이다.

### 3.2 Timing Constraints의 명세

이 절에서는 앞서 생성한 language construct들의 timing constraints을 명시하는 방법을 설명한다. 이를 위해 (1) data flow graph의 factory에 전체 알고리즘의 timing constraint를 부여하고, (2) 이를 만족시키기 위해 factory 내부의 각 language construct에 대한 timing constraint을 도출하고 부여한다.

본 예제에서는 factory에 두 개의 timing constraints를 부여한다. 첫째, 전체 알고리즘 수행의 end-to-end latency의 최대 허용치를 정하기 위해 factory의 deadline을 기술한다. 둘째, 전체 알고리즘의 단위 시간당 입출력 스트림 데이터의 최소 개수를 정하기 위해 각 factory port에 input/output minimum rate constraints 부여한다. 이 timing constraints는 [5]를 우리가 분석하여 추가적으로 도출한 것이다. 먼저, 최대 허용 end-to-end latency는 factory의 deadline으로 명시한다. 다음으로, 단위시간당 입출력 스트림 데이터의 최소 개수 제약을 입출력 factory port의 input/output minimum rate constraint로 부여한다.

다음으로, 개별 processing cell의 deadline은 입력 센서 스트림 데이터를 받아 출력 스트림 데이터를 내보내는데 소요되는 전체 시간이 factory의 deadline을 넘지 않도록 조정하여 부여한다. 입출력 data port의 input/output minimum rate constraint는 출력 factory port의 output minimum rate constraint를 맞출 수 있도록 부여한다. 각 language construct 단에서 알고리즘의 특성 상 freshness constraint와 correlation constraint가 필요하다고 판단될 경우 이를 부여한다. 우리가 SPLAD 프로그램에 timing constraints를 기술한 결과를 Fig. 8에 표시한다.

Table 5 Camera-LiDAR sensor fusion 기반 detection 알고리즘의 세부 함수 분석

| 함수 이름 | 설명 | 입력 스트림 데이터 | 출력 스트림 데이터 |
|---|---|---|---|
| 2D object detection | RGB 카메라의 이미지 데이터를 받아 그 이미지 내의 object를 탐지하여 bounding box가 그려진 image를 출력한다. | 2D image (알고리즘의 입력) | Object validation 의 입력 |
| Object interpola-tion | object가 탐지된 image와 raw image를 입력받아 이미 탐지된 object들을 raw image에서 추적하여, bounding box가 그려진 image를 출력한다. | 2D image (알고리즘의 입력), Object updator의 출력 | Object updator의 입력 |
| Object validation | 최신의 2D object detection 결과를 가지고 interpolation에서 탐지된 object들을 갱신한다. | 2D object detection 의 출력, Object updator의 출력 | Object updator의 입력 |
| Object update | Interpolation의 결과와 validation의 결과 중 최신의 것을 받아 현재 주변 object 탐지 결과를 update한다. | Object interpolation 의 출력, Object validation 의 출력 | Object fusion의 입력 |
| Points projection | 평면상에서 point cloud를 projection 한다. | Point cloud (알고리즘의 입력) | Distance estimation 의 입력 |
| Distance estimation | Object가 탐지된 image와 point cloud가 projection된 결과를 통해 object의 depth를 판단한다. | Points projection 의 출력, Object update의 출력 | Object projection 의 입력 |
| Object projection | Distance estimation의 input point projection 상에서의 area of interest를 표시해준다. | Distance estimation 의 출력 | Object fusion의 입력 |
| Point cloud clustering | LiDAR의 pointcloud를 input으로 받아 인접한 point들을 clustering 한다. | Pointcloud (알고리즘의 입력) | Object fusion의 입력 |
| Object fusion | Merge 된 Point cloud lustering 과 object projection output 을 받아 3D point cloud cluster 에게 image 와 distance 를 부여하여 3D object map을 그린다. | Object projection 의 출력, Pointcloud clustering 의 출력 | 3D object map (알고리즘의 출력) |

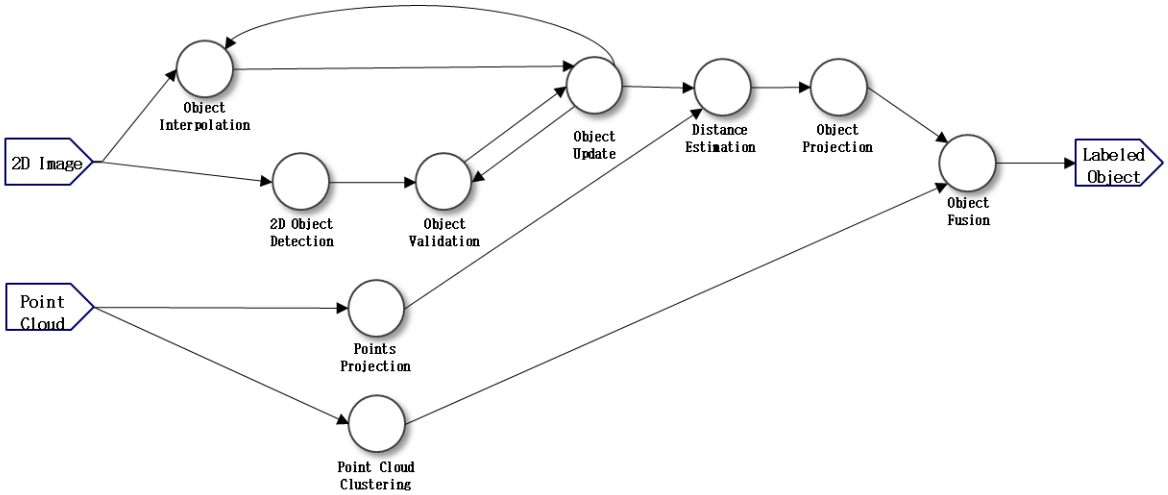



Fig. 6 예제 알고리즘의 intermediate representation

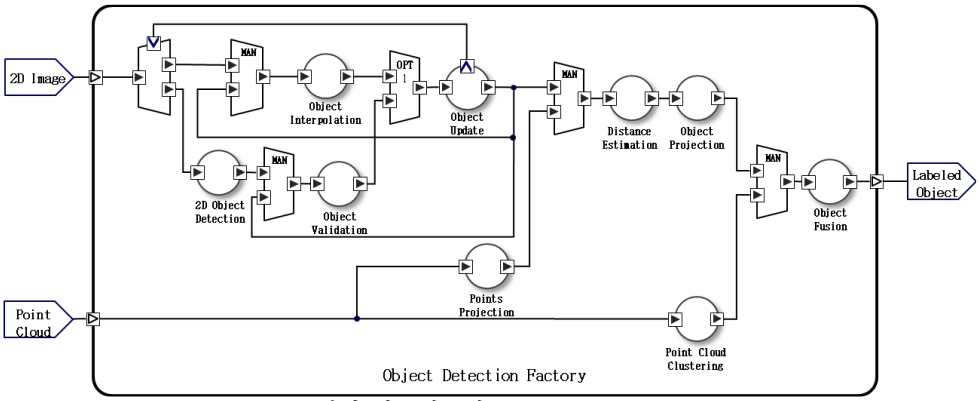



Fig. 7 예제 알고리즘의 SPLAD data flow graph

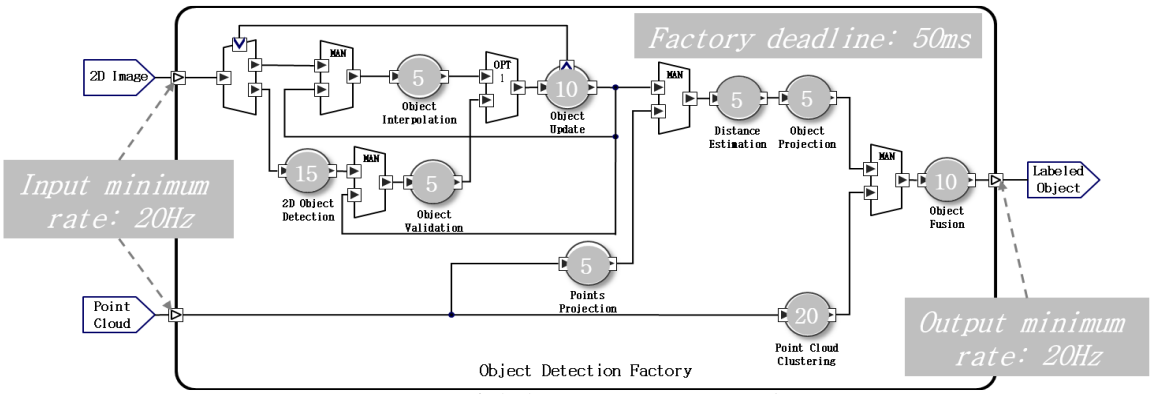



Fig. 8 SPLAD 응용의 timing semantics 부여

### 3.3 Exception Handling의 명세

이 절에서는 알고리즘의 신뢰성 요구사항을 만족시키기 위해 exception handling을 명시하는 방법을 설명한다.

우리는 두 개의 예외상황을 다룬다. 첫째는 LiDAR 센서 데이터의 sensing range 이외의 입력값에 대한 예외상황이다. 이 예제에서는 object projection processing cell에서 입력 데이터 스트림으로 object의 거리 정보를 받는데, 이 값이 LiDAR의 sensing range인 150m 벗어나면 invalid stream data input으로 정의한다[6]. Exception handling을 위해 해당 스트림 데이터를 logging하는 exception handler 함수를 구현한다.

둘째는 응용의 이벤트 데이터가 corrupt되어 전달되는 예외상황이다. 이 예제에서는 하나의 selector의 invalid event input에 대한 exception handling 방법을 설명한다. 2D image 입력을 받는 selector의 경우 2개의 출력 분기 경로가 정의되어 있고, 이를 결정하는 이벤트 데이터는 object update processing cell에서 출력한다. 이 selector의 출력 분기를 결정하는 이벤트 데이터의 값을 0, 1으로 정의하면, 이를 벗어나는 입력 이벤트 데이터의 값은 invalid event data input이 된다. Exception handling을 위해 invalid event data input을 무시하고 selector의 기존 출력 분기를 계속 유지하는 exception handler 함수를 구현한다.

## 4. Future Work

SPLAD runtime은 SPLAD 응용이 수행되는 환경이다. SPLAD runtime은 SPLAD에서 정의한 language construct들이 설정한 semantics에 따라 수행 가능하도록 하는 execution model과 이를 지원하기 위한 기반 시스템으로 구성된다. 추후 연구에서 이 두 가지를 설계하고 구현한다.

Execution model을 설계하기 위해서는 language construct를 실제 수행 가능 객체로 만들기 위해 task로 mapping 해야 한다. Task란 각 language construct에서 주어진 작업을 수행하는 주체이며, runtime의 scheduling entity이다. Task는 SPLAD에서 정의한 기능을 수행하는 task와 응용 개발자가 정의한 연산을 수행하는 task로 구분된다. 전자는 language construct의 basic operational semantics와 timing semantics에서 명시한 작업을 수행하고, 후자는 응용 개발자가 processing cell에 정의한 연산과 exception handling 함수를 수행한다.

SPLAD runtime이 구현되는 기반 시스템은 timing semantics에서 명시되는 input/output minimum rate constraint, freshness constraint, correlation constraint를 보장하는 실시간 통신 메커니즘을 지원해야 한다. 또한 deadline을 보장하기 위한 실시간 처리 메커니즘을 제공해야 한다.

## 5. 결 론

본 논문에서는 자율주행 자동차 응용의 대표적 비기능적 요구사항인 실시간, 신뢰성을 만족시킬 수 있는 stream processing language인 SPLAD (Stream Processing Language for Autonomous Driving)를 제안하였다. SPLAD는 알고리즘의 basic operation semantics, timing semantics, exception handing를 명시할 수 있도록 한다. SPLAD로 작성한 응용을 수행하기 위해서는 정의한 language construct에 부여된 semantics에 따라 수행 가능한 메커니즘을 제공하는 SPLAD runtime이 필요하다. SPLAD runtime의 설계와 구현은 후속 연구에서 다룬다.

## Acknowledgement

이 논문은 2017년도 정부(미래창조과학부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No. 2016-0-00170, (광역주행-총괄/1세부) 차량 통신 기반의 광역 주행환경인지 및 협조 주행기술개발)

## References

1) Brenner W., Herrmann A., An Overview of Technology, Benefits and Impact of Automated and Autonomous Driving on the Automotive Industry. In: Linnhoff-Popien C., Schneider

R., Zaddach M. (eds) Digital Marketplaces Unleashed. Springer, Berlin, Heidelberg, 2018.
2) Hirzel, Martin, et al., "Spl stream processing language specification." NewYork: IBM Research Division TJ. Watson Research Center, IBM Research Report: RC24897 (W0911 044), 2009.
3) Matei Zaharia, et al., “Discretized Streams: An Efficient and Fault-Tolerant Model for Stream Processing on Large Clusters,” Proceedings of USENIX conference on Hot topics in cloud computing, 2012.
4) Ankit Toshniwal, et al., “Storm@ twitter,” Proceedings of the 2014 ACM SIGMOD international conference on Management of data, p. 147-156, 2014.
5) Kato, Shinpei, et al., An open approach to autonomous vehicles. IEEE Micro, 35.6, p. 60-68, 2015.
6) Quanergy Systems, “360° 3D LIDAR M8-1 SENSOR”, datasheet, 2015.