[두산로보틱스] 지능형 로보틱스 엔지니어

디지털 트윈 기반 서비스 로봇 운영 시스템 구성

TEAM F-2 JUNE

이재호, 배재성, 전유진, 정은영

[멘토] 김루진

K-Digital Training

목 차

K-Digital Training

01 프로젝트 개요

1

프로젝트 주제 및 선정 배경, 기획의도

디지털 트윈 기반 서비스 로봇 운영 시스템 바탕으로 산업 현장에서 차량 이동성 확보 또는 경비의 목적으로 활용할 수 있는 로봇 프로그래밍 설계 및 테스트

2

프로젝트 내용

기존 터틀봇3의 기본 패키지 기능의 한계를 보완하기 위하여 자율 주행, 머니퓰레이션, 객체 인식 기능을 개선 및 탑재

3

활용 장비 및 재료

Turtlebot3 Waffle Pi (Jetson Nano, OpenCR), Open Manipulation, Li Batteries, Web Cam, USB Hub, Monitor, Cable

4

프로젝트 구조

기존 Turtlebot3 패키지 기능을 범용적으로 활용할 수 있도록 개선 하기 위하여 5월 9일부터 5월 22일까지 진행 프로젝트 수행
터틀봇3 기본 패키지 분석 및 필요사항 확정, 프로그래밍 개발, 시뮬레이션 및 테스트 수행

5

활용방안 및 기대 효과

산업 현장에서 경비를 수행하고 운송과정에서 떨어진 운송품을 정리하여 산업 환경 개선 도모

02 K-Digital Training

프로젝트 팀 구성 및 역할

훈련생	역할	담당 업무	
이재호	팀장	Lane Detection (Main)	Control Lane (Sub)
배재성	팀원	Manipulation (Main)	ArUco Marker (Sub)
전유진	팀원	Control Lane (Main)	ArUco Marker (Sub)
정은영	팀원	Object Detection (Main)	RQt App (Main)
김루진, 이정우	멘토	프로그래밍 피드백, 프로젝트 질의응답	

03 프로젝트 수행 절차 및 방법

구분	기간	활동	비고
사전 기획	5/9(금) ~ 5/11(일)	시나리오 설정 및 역할 분담 Turtlebot3 Emanual 분석	목표 설정
Turtlebot3 기본 패키지 분석 및 요구사항 설정	5/12(월) ~ 5/14(수)	Turtlebot3 데모 기능 구현 필요 기술 확정	
Turtlebot3 Simulation Implementation	5/15(목) ~ 5/18(일)	Simulation 구현 Turtlebot3 분석 발표	팀별 중간보고 실시
Turtlebot3 실제 구동 및 RQt 구현	5/19(월) ~ 5/21(수)	파라미터 조정 및 테스트 파라미터 조정 및 테스트	최적화, 오류 수정
발표자료 정리 및 발표	5/21(수) ~ 5/22(목)	발표 자료 작성 및 발표	
총 개발기간	5/9(금) ~ 5/22(목)(총 2주)		

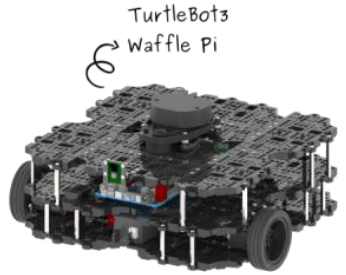
프로젝트 수행 경과

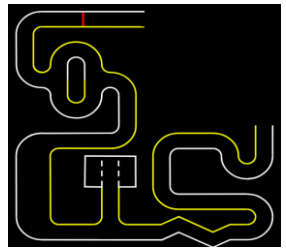
기존 터틀봇3의 기본 패키지 기능 한계

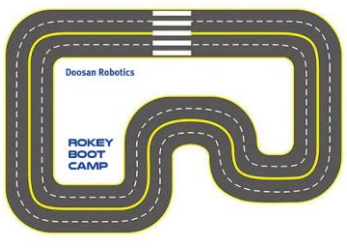
- 색에 따라 차선 구분
  - 차선이 많거나 색이 다양한 경우, 차선을 제대로 인지하지 못함
- 일반적인 객체인식이 어려움
  - 인식할 수 있는 객체가 적으며 다양한 상황을 반영하지 못함
- 로봇팔 제어 기능이 미탑재
  - 물체를 이동시키는 기능을 수행할 수 없음

Table 1. Bill of materials

부품명	Qty (EA)
Waffle Pi	1
Open manipulation	1
Li battery	2
Web cam	1
USB Hub	1
Monitor	1
Cable	1







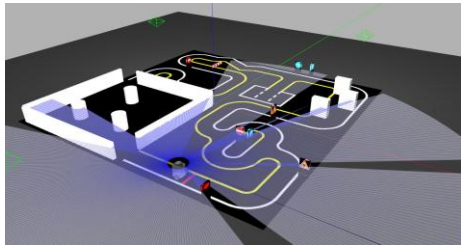


Fig 1. 기존 터틀봇3과 환경

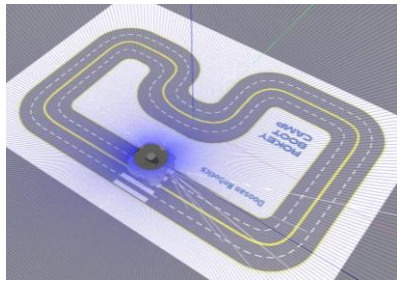


Fig 2. 개선된 터틀봇3 환경

➢ **이러한 점을 보완할 수 있는 기술 개발 및 검증**
- **상황 변수가 다양한 트랙으로 교체**
- **Deep Learning 기반 객체인식**
- **로봇팔 탑재 및 제어 기능 구현**
- **Simulation과 Test를 위하여 터틀봇3 Buger을 Waffle Pi로 교체**

프로젝트 수행 경과

Detection, Control, and Manipulation of Turtlebot3

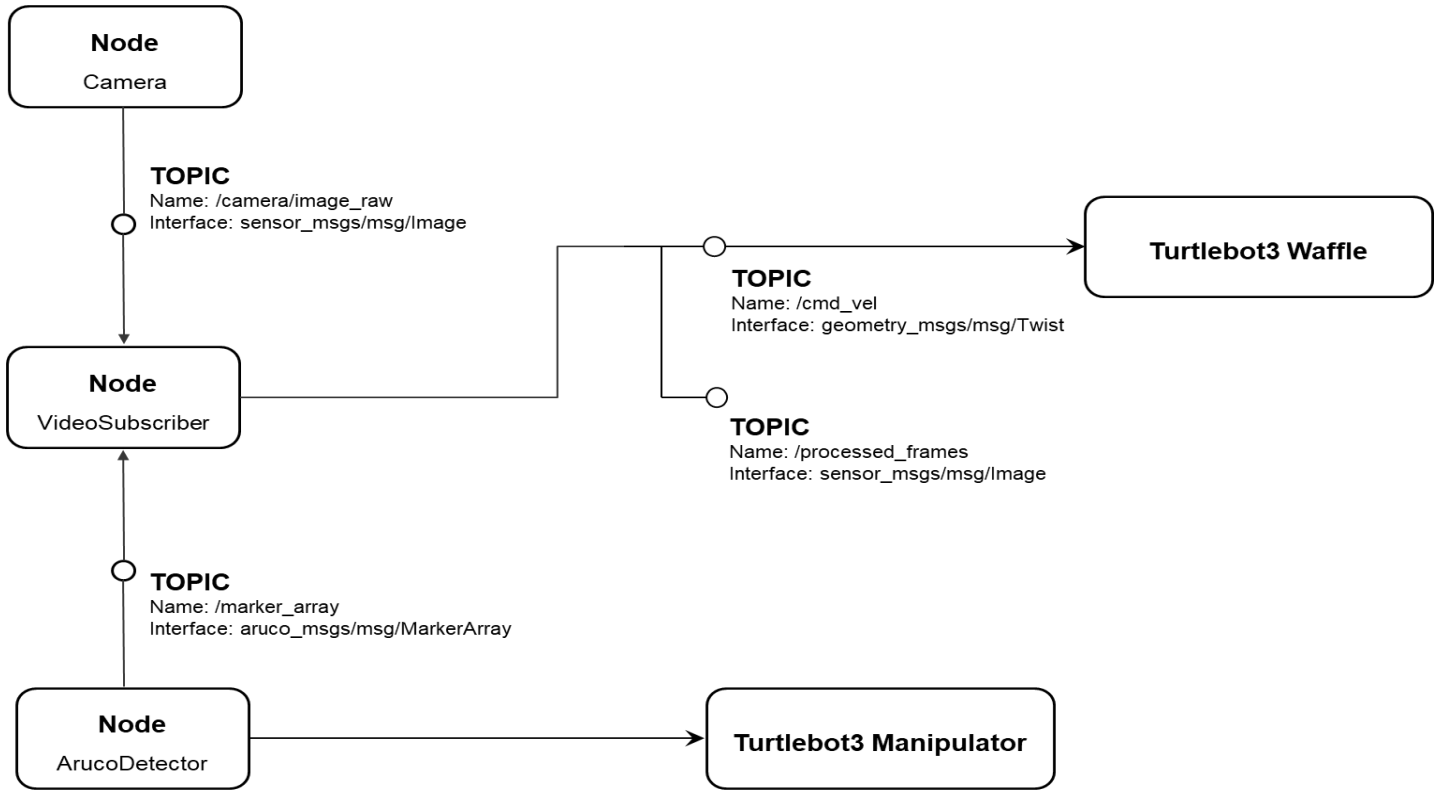


Fig 3. Node architecture of Waffle Pi Main system

Lane Detection

- Block Diagram of Lane Detection Algorithm

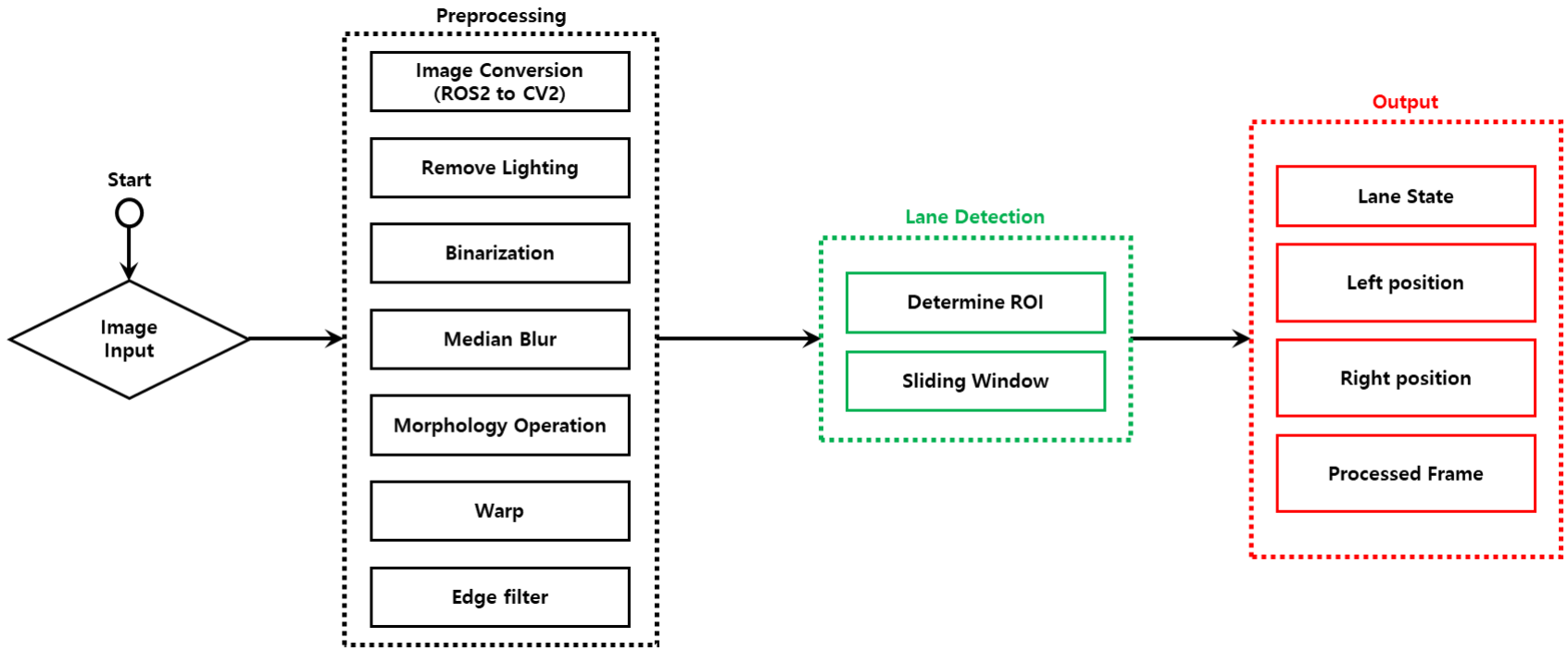


Fig 4. Block diagram of lane detection

프로젝트 수행 경과

Lane Detection

- Preprocessing
  - Image Conversion – 직렬화된 이미지를 CV2에 적합하게 변환
  - Remove Lighting – 차선인식을 위한 광도 및 채도 처리
    - 차선 인식의 성능을 제고하기 위해 **흰색 범위**를 필터링하여 제외
    - 엣지 특성을 살리기 위해 Gaussian Blur, 샤프닝, Normalization 처리
  - Binarization – 노이즈를 제거하고 엣지 특성을 살리기 위해 이진화
    - 중위수를 활용하는 Median Blur으로 노이즈 제거
  - **Morphology Operatioin**으로 작은 노이즈 제거
  - Warp - 수신된 원본 이미지를 투사한 것처럼 변환
  - Edge Filter - Soble Filter로 Edge 추출

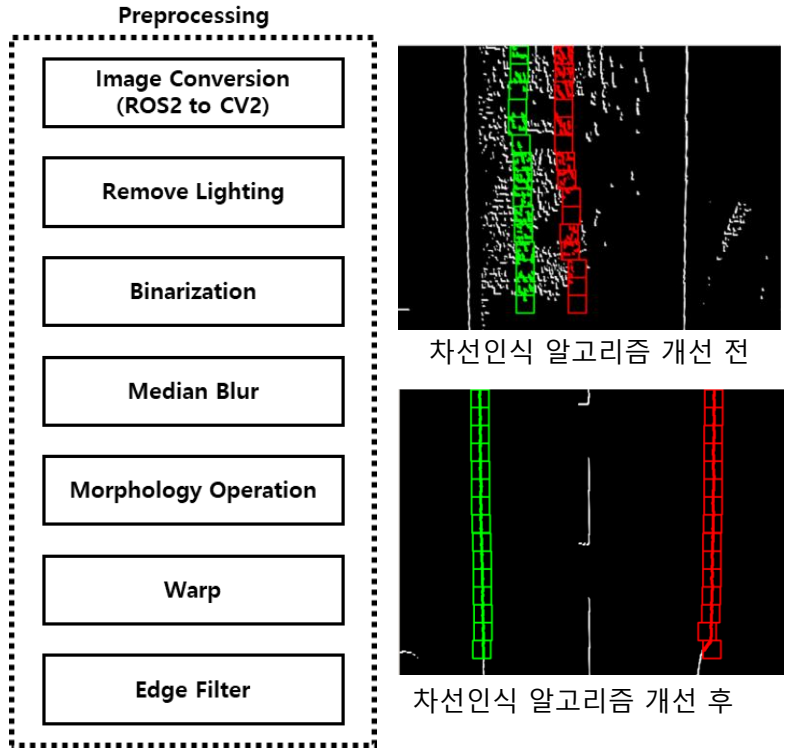


Fig 5. Preprocessing step과 차선인식 개선 전후

프로젝트 수행 경과

Lane Detection

- Lane Detection
  - Determine ROI (Region of Interest)
    - 연산 효율성과 차선 인식 성능 제고를 위해 차선 감지 영역 분할 및 스캔
  - Sliding Window
    - 차선 픽셀을 효과적으로 추출하기 위해 ROI를 분할하고 각 영역을 이동하며 차선 유무 감지

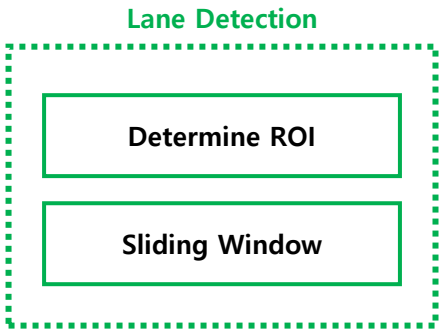


Fig 6. Lane detection step

```
def lane_detect(self, frame):
    frame, _ = self.camera_processor.process_image(frame)
    slide_frame = frame[frame.shape[0] - 200 :frame.shape[0] - 150, :]
    detected, left, right, _ = self.slide_window_processor.slide(slide_frame)
    processed_frame = self.slide_window_processor.lane_visualization(frame, left, right)
    return detected, left, right, processed_frame
```

Fig 7. Sliding window code

Lane Control

- Block Diagram of Lane Control Algorithm

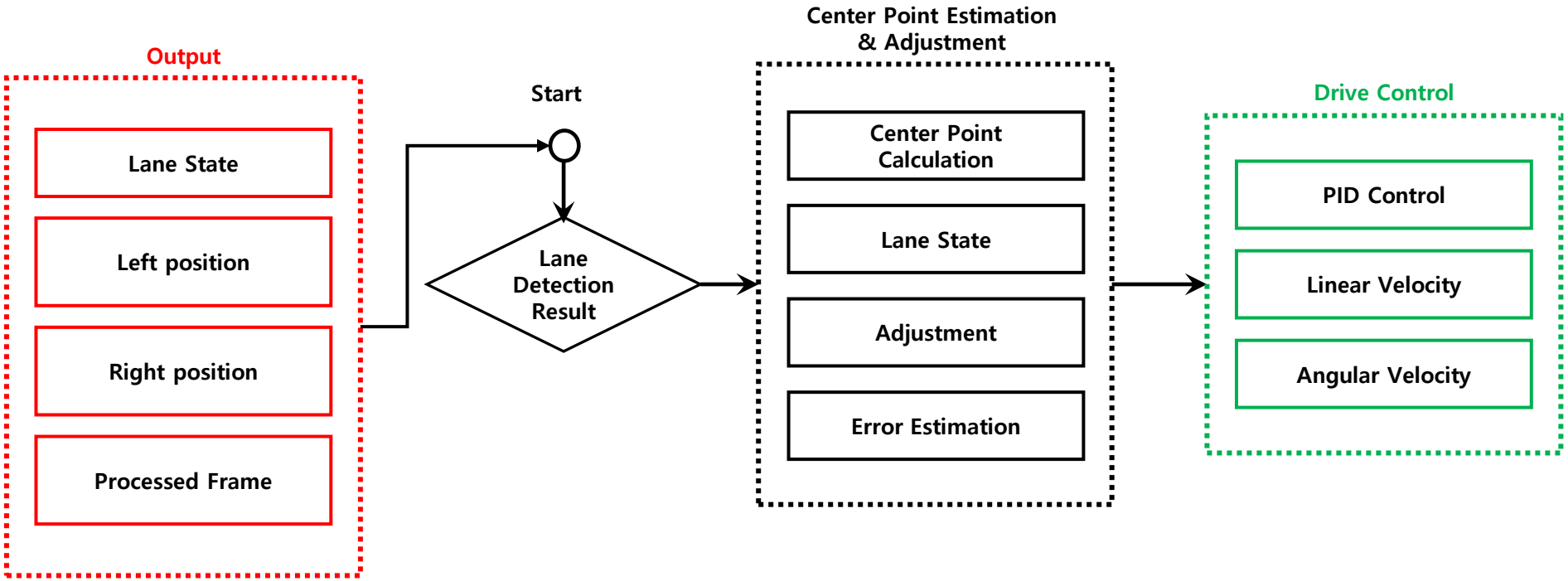


Fig 8. Block diagram of lane control

프로젝트 수행 경과

Lane Control

- Center Point Estimation & Adjustment
  - Center Point Calculation
    - Lane detection에서 산출한 양 차선의 위치를 바탕으로 차선 중앙점 계산
  - Lane State
    - 산출한 차선 위치에 의해 차선 상태 판정 (left / right)
  - Adjustment
    - 차선 상태에 따른 보정값 부여
  - Error Estimation
    - 산출된 차선 중앙값과 이미지 중앙값의 차를 Error로 설정

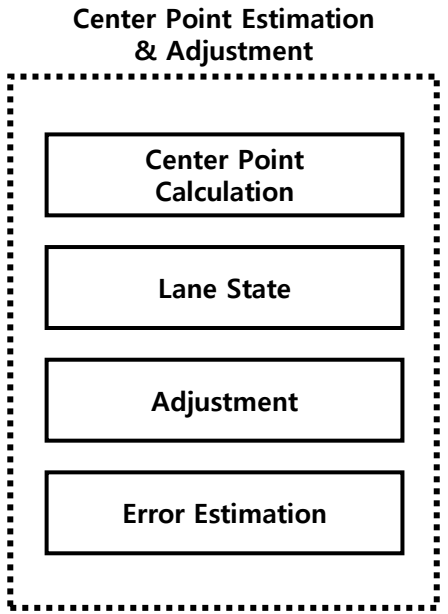


Fig 9. Center point estimation and adjustment step

```
frame = self.bridge.imgmsg_to_cv2(msg, desired_encoding='bgr8')
detected, left, right, processed = self.lane_detect(frame)

center = float((left + right) / 2)
if detected == 'left':
    center += 0.125 * 320
elif detected == 'right':
    center -= 0.125 * 320

error = center - 320.0
```

Fig 10. Center point estimation and adjustment code

프로젝트 수행 경과

Lane Control

- Center Point Estimation & Adjustment
  - PID Control
    - PID Control을 활용하여 선형속도와 각속도를 산출

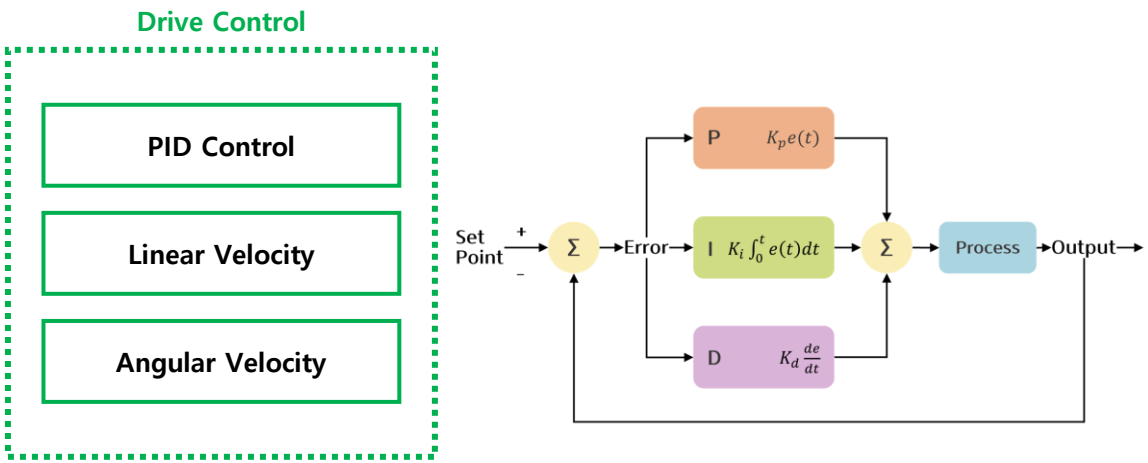


Fig 11. Drive control components and diagram

```python
# PID 제어 계산
self.integral += error
self.integral = max(min(self.integral, self.integral_limit), -self.integral_limit)
derivative = error - self.last_error
self.last_error = error

angular_z = (
    self.Kp * error +
    self.Ki * self.integral +
    self.Kd * derivative
)

twist = Twist()

# 구간에 따라 속도 조절
twist.linear.x = self.MAX_VEL if detected == 'both' else self.MIN_VEL
twist.angular.z = -max(min(angular_z, 2.0), -2.0)  # 조향 제한

# 토픽 발행
self.pub_cmd_vel.publish(twist)

# 결과 이미지 발행
processed_msg = self.bridge.cv2_to_imgmsg(processed, encoding='bgr8')
self.image_publisher.publish(processed_msg)
```

Fig 12. Center point estimation and adjustment code

프로젝트 수행 경과

ArUco Marker 인식

- OpenCV에서 제공하는 오픈소스 라이브러리
- N x N 행렬의 흑백 정사각형 패턴
- 로봇 제어, 증강 현실, 자동화 공정 등에서 사용

- 인식 및 메시지 출력
  - Topic : /marker_array
  - Aruco_msgs/msg/MarkerArray
    - 마커 ID
    - 위치: position. x, y, z
    - 회전: orientation. z (roll), y (pitch), x (yaw)

Fig 13. ArUco Marker 인식

ArUco Marker 인식

- 실행 과정

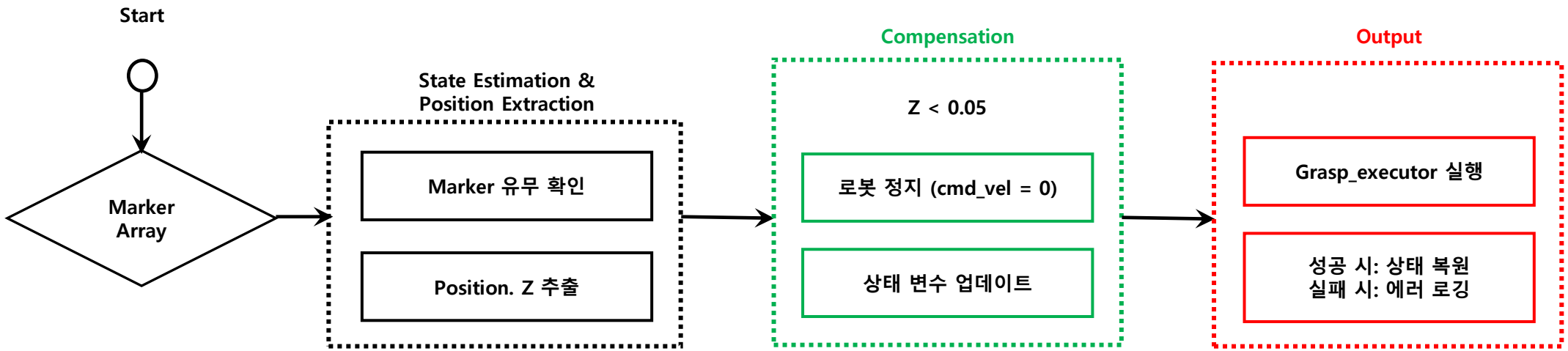


Fig 14. Block diagram of marker array

프로젝트 수행 경과

ArUco Marker 코드 설명

- ArUco Marker 인식 후 주행 정지 및 외부 노드 실행

```
def marker_callback(self, marker_array_msg):
    z0 = marker_array_msg.markers[0].pose.pose.position.z

    if z0 < 0.05 and not self.stop_requested and not self.stopped:
        twist = Twist()
        self.pub_cmd_vel.publish(twist)
        self.stop_requested = True
        self.stopped = True

        subprocess.run(['ros2', 'run', 'grasp_executor_cpp', 'grasp_executor'], check=True)

        self.stop_requested = False
        self.stopped = False
```

Fig 15. ArUco Maker code

- Z값 (로봇과의 거리) < 0.05m :
  - 정지 명령 (cmd_vel = 0)
  - Grasp_executor 실행 -> gripper 작동
  - 작업 완료 후 주행 재개

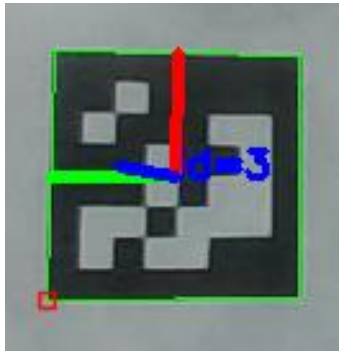


Fig 16. ArUco Maker recognition

프로젝트 수행 경과

Manipulator

- Aruco marker 추가 작업
  - camera intrinsic calibration
    - 9(raw) x 7(column) checker board사용

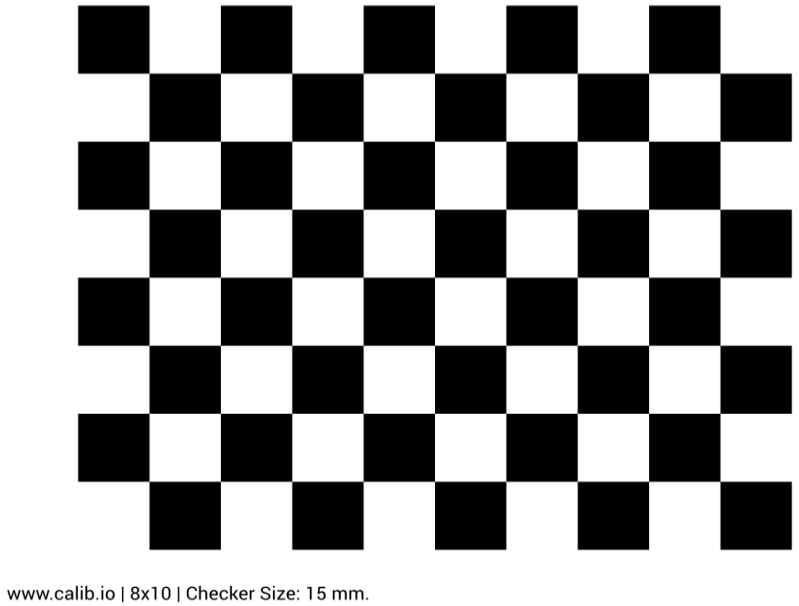


Fig 17. Checker board

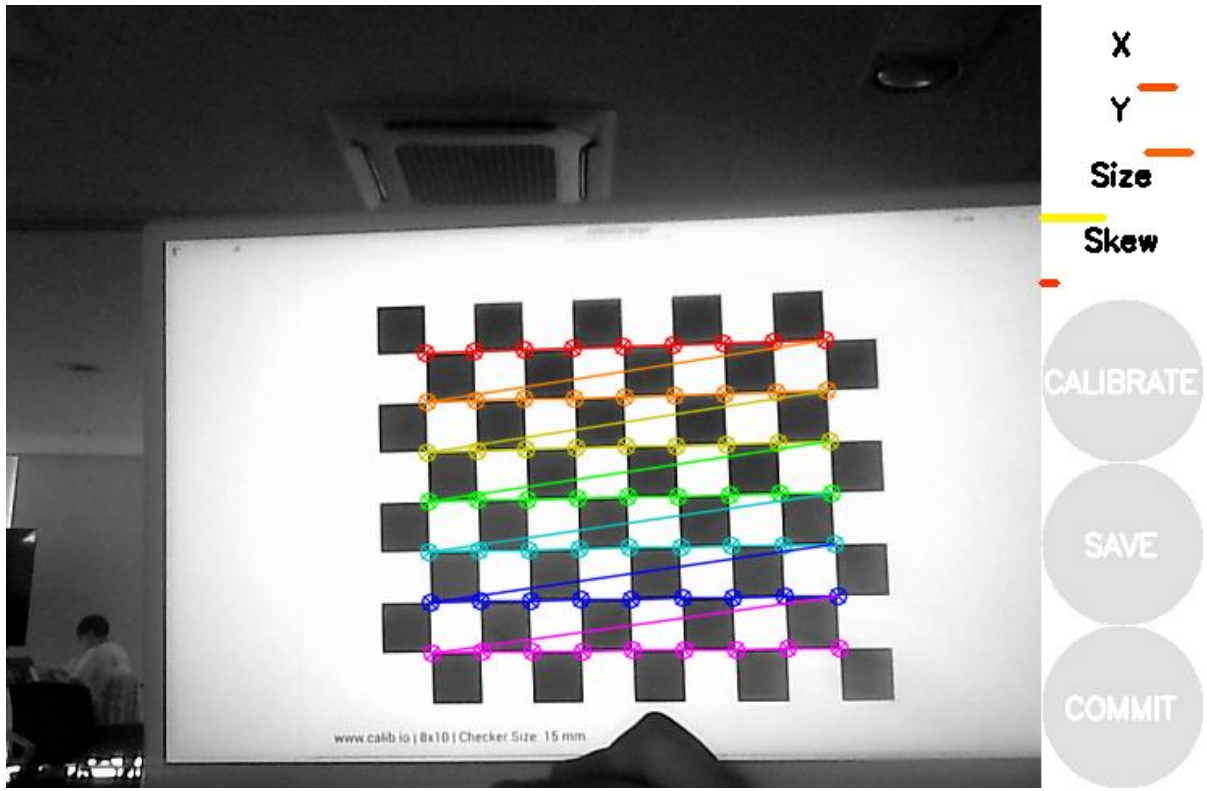


Fig 18. Checker board 인식

Manipulator(main.cpp)

- ArUco marker
  - 카메라로 ArUco Marker 인식
  - 시스템 간략 Flow

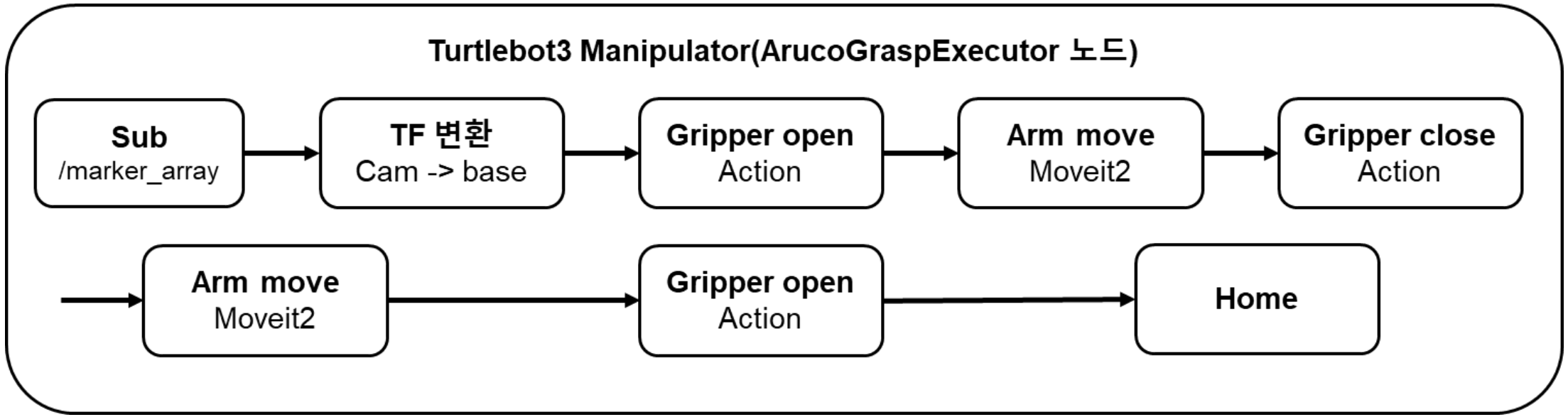


Fig 19. Turtlebot3 Manipulator flow chart

프로젝트 수행 경과

Manipulator(main.cpp)

- 문제점
  - Moveit2가 ArUco marker 좌표를 토대로 좌표 변환 및 경로
    - ArUco marker가 제대로 탐지되지 않거나, 좌표 탐지가 명확하지 않으면, 경로 계획 에러 발생
  - 온전히 ArUco Marker의 좌표 정보에 의존
    - 좌표 탐지 알고리즘 고도화 필요

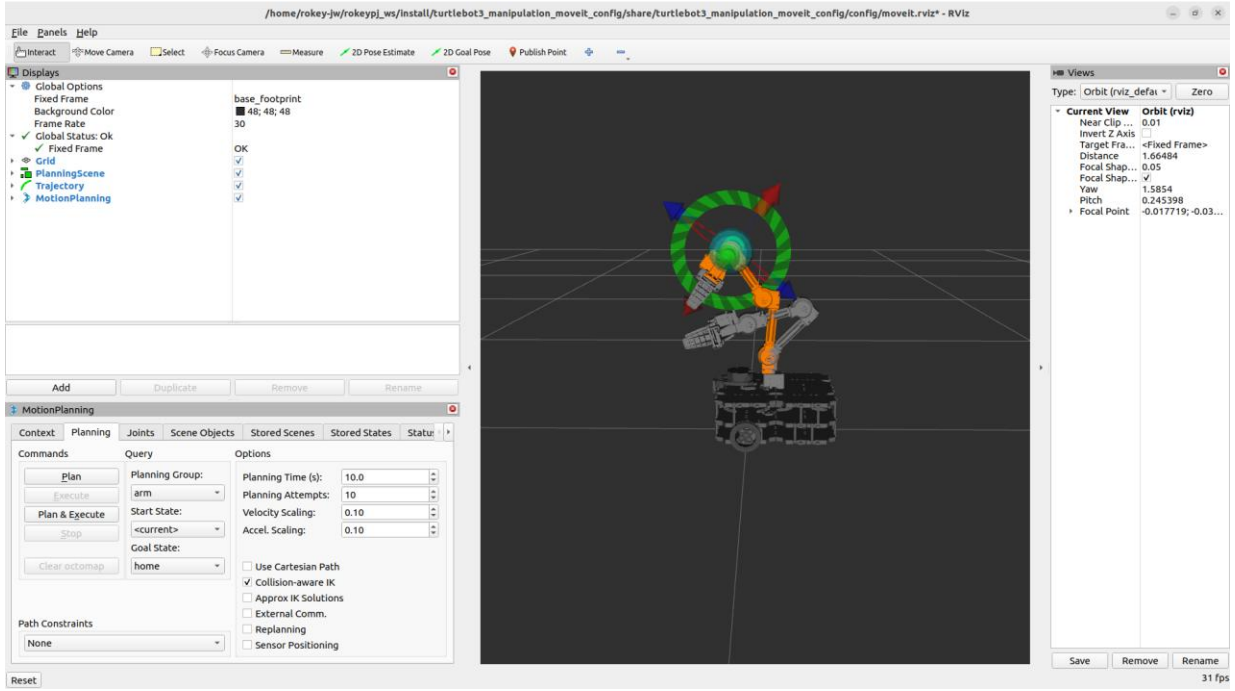


Fig 20. Turtlebot3 Manipulator with rviz

프로젝트 수행 경과

Manipulator(환경 설정 구성 내용)

- End effector link의 **control group 변경** : gripper → arm (moveit_config/config/turtlebot3_manipulation.srdf)
  - Moveit2은 **end effector link가 원하는 좌표로 이동하기 위한 계산**하는 것이라 판단

```
<!--END EFFECTOR: Purpose: Represent information about an end effector.-->
<!-- <end_effector name="end_effector_link" parent_link="link5" group="gripper"/> -->
<!-- <end_effector name="end_effector_link" parent_link="link5" group="arm"/> -->
<end_effector name="end_effector_link" parent_link="link5" group="arm"/>
```

Fig 21. End effector code1

- Camera link를 link5와 결합
  - camera link로 부터 base_link로 tf시, 계산의 편의성을 위해 최대한 가까운 link와 결합
    - 혹시 모를 노이즈에 대비

- End effector link의 collision error가 빈번하게 발생
  - 빈번하게 발생하는 error로 인해 적용

```
<link name="${prefix}end_effector_link">
  <visual>
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <geometry>
      <box size="0.01 0.01 0.01" />
    </geometry>
    <material name="red"/>
  </visual>
  <!-- 추가 -->
  <collision>
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <geometry>
      <box size="0.01 0.01 0.01"/>
    </geometry>
  </collision>
</link>
```

Fig 22. End effector code2

프로젝트 수행 경과

YOLO Object Detection

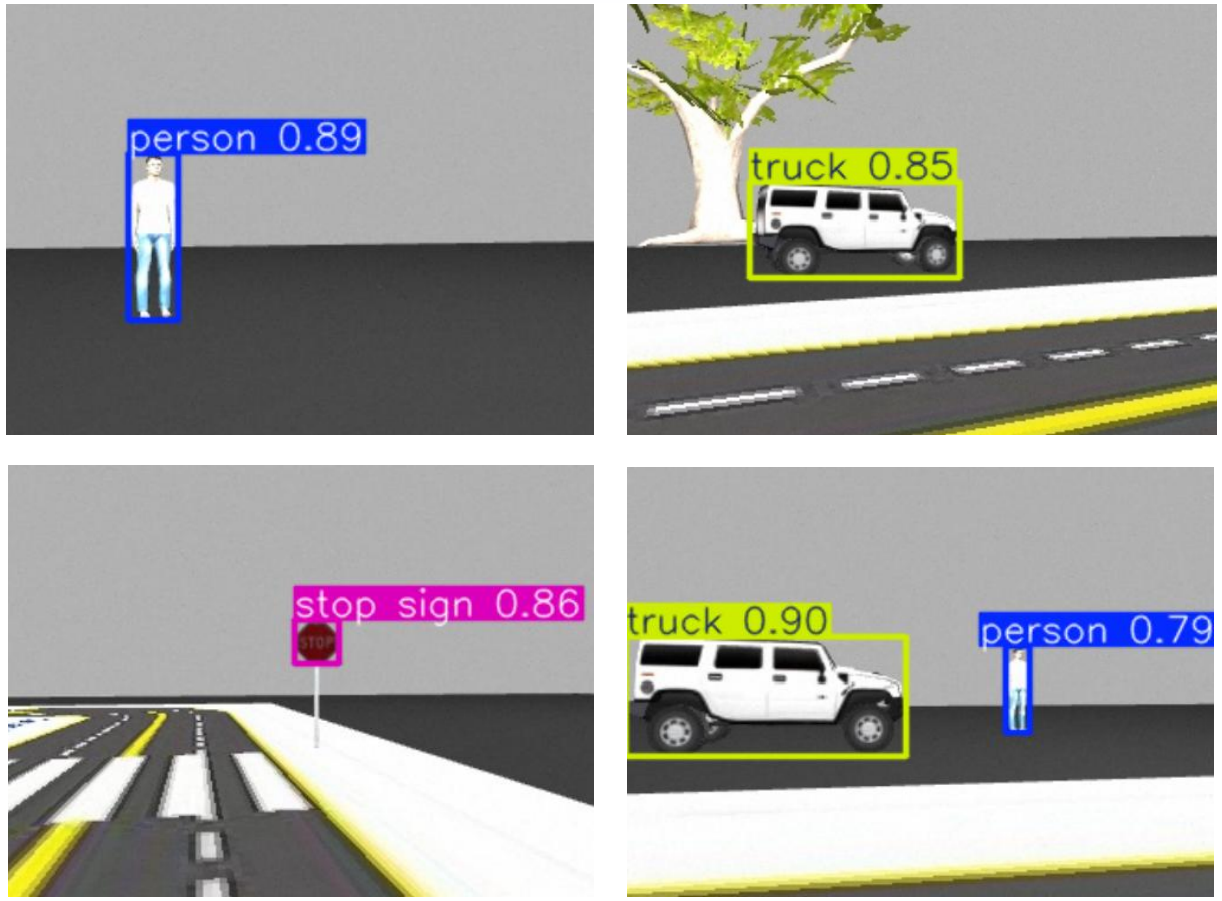


Fig 23 Object detection in Gazebo simulator

YOLO Object Detection

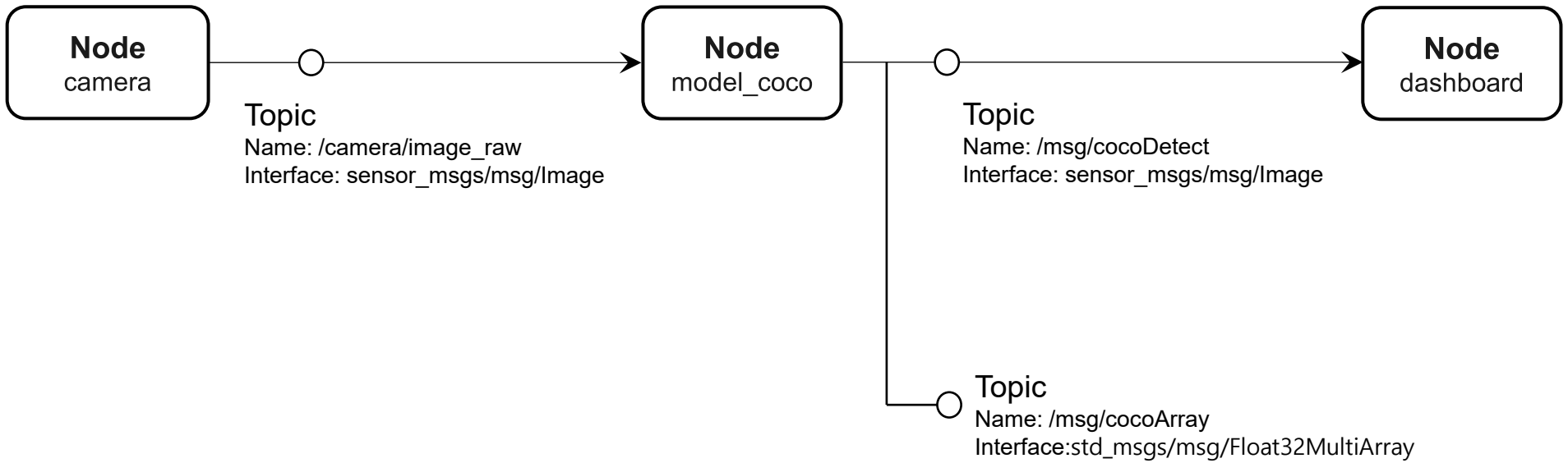


Fig 24. Node architecture of YOLO image detect

YOLO Object Detection

Sub: [/image_raw]
Pub: [/msg/cocoDetect] [/msg/cocoArray]

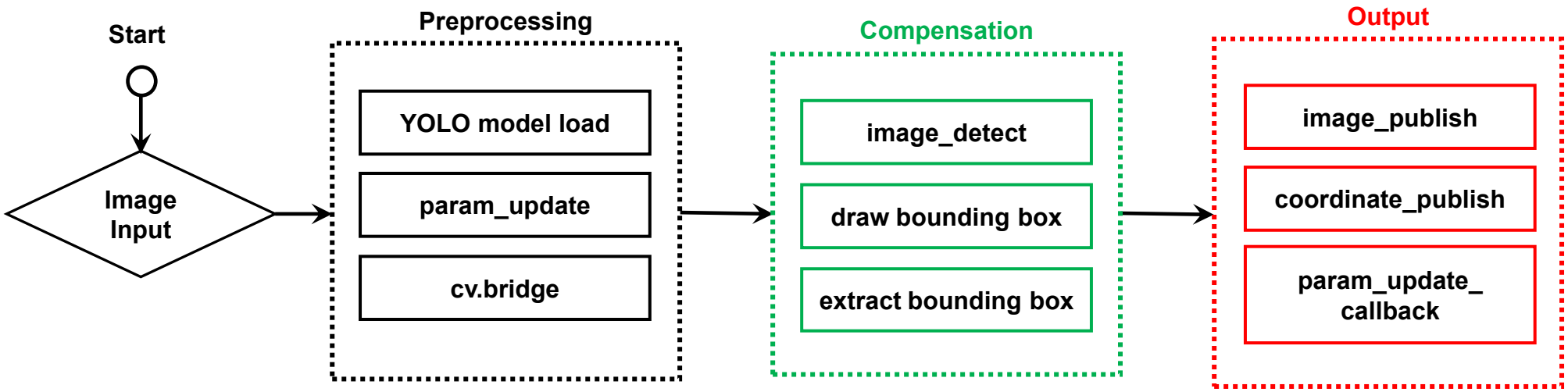


Fig 25. Block diagram of YOLO object detection

프로젝트 수행 경과

Multiplexer cmd_vel

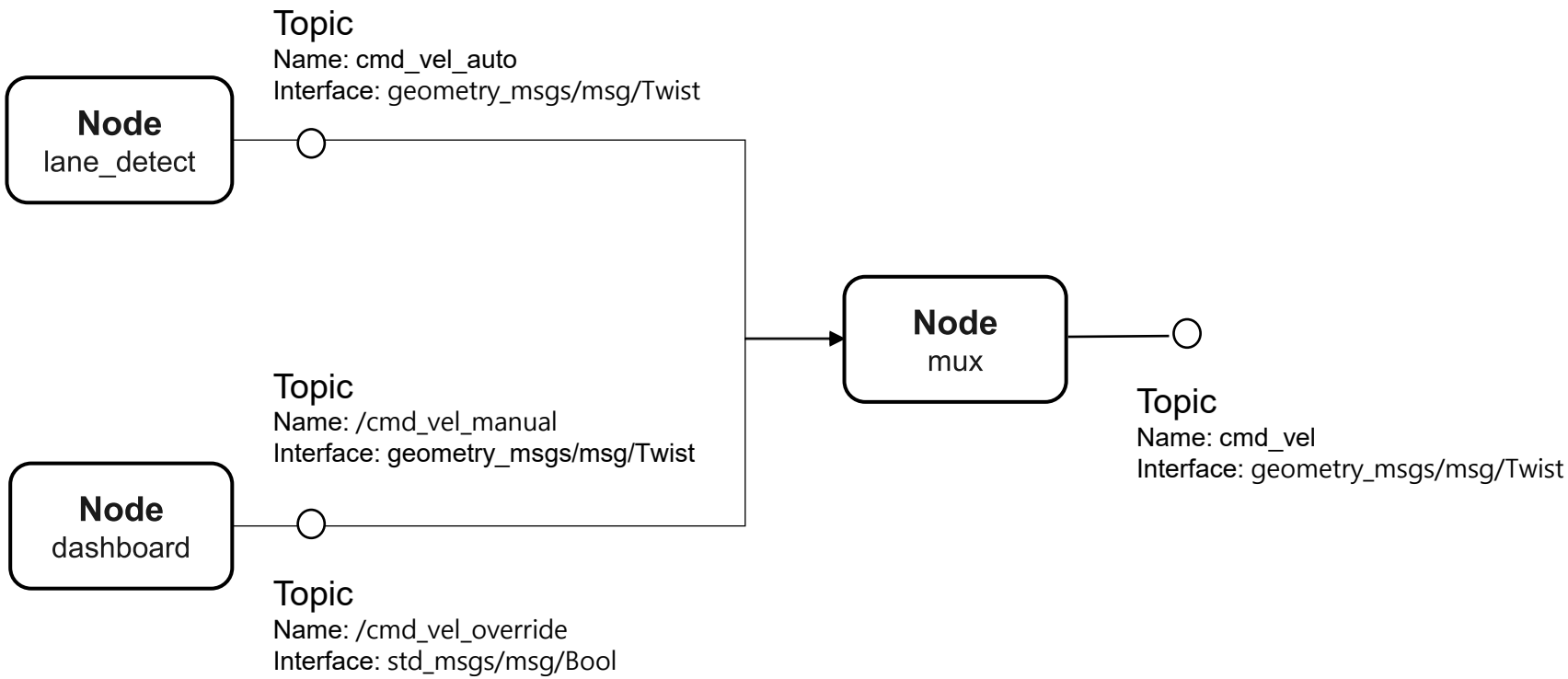


Fig 26. Node architecture of multiflux cmd_vel

프로젝트 수행 경과

Multiplexer cmd_vel

/turtlebot3_ws/src/lane_detect/launch/subscriber_node.launch.py

```python
def generate_launch_description():
    subscriber_node = Node(
            package='lane_detect',
            executable='subscriber_node',
            name='subscriber_node',
            output='screen',
            remappings=[
                ('/control/lane', '/detect/lane'),
                ('/cmd_vel', '/cmd_vel_auto')
            ]
        )
    return LaunchDescription([
        subscriber_node
    ])
```

Fig 27. Remapping code for cmd_vel_mux

/turtlebot3_ws/src/visual_dashboard/visual_dashboard/dashboard_node.py

```python
self.cmd_pub = self.create_publisher(Twist, '/cmd_vel_manual', 10)
self.override_pub = self.create_publisher(Bool, '/cmd_vel_override', 10)
self.cmd_sub = None
```

Fig 28. Publish msg in dashboard node

프로젝트 수행 경과

Multiplexer cmd_vel

sub: [/cmd_vel_auto] [/cmd_vel_manual] [/cmd_vel_override]
pub: [/cmd_vel]

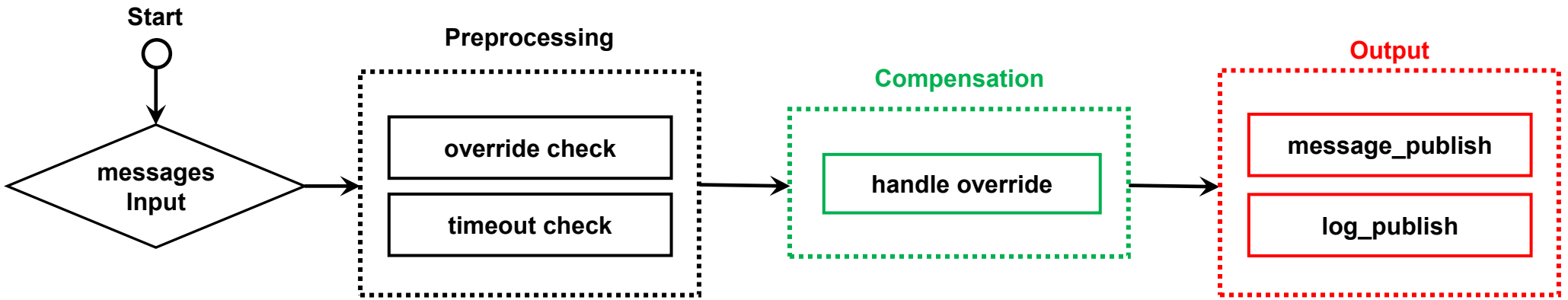


Fig 29. Block diagram of cmd_vel_mux

프로젝트 수행 경과

RQt User Interface

Sub: [/image_raw] [/image_projected] [/msg/cocoDetect] [/processed_frames] [/odom]

Pub: [/cmd_vel_manual] [/cmd_vel_override]

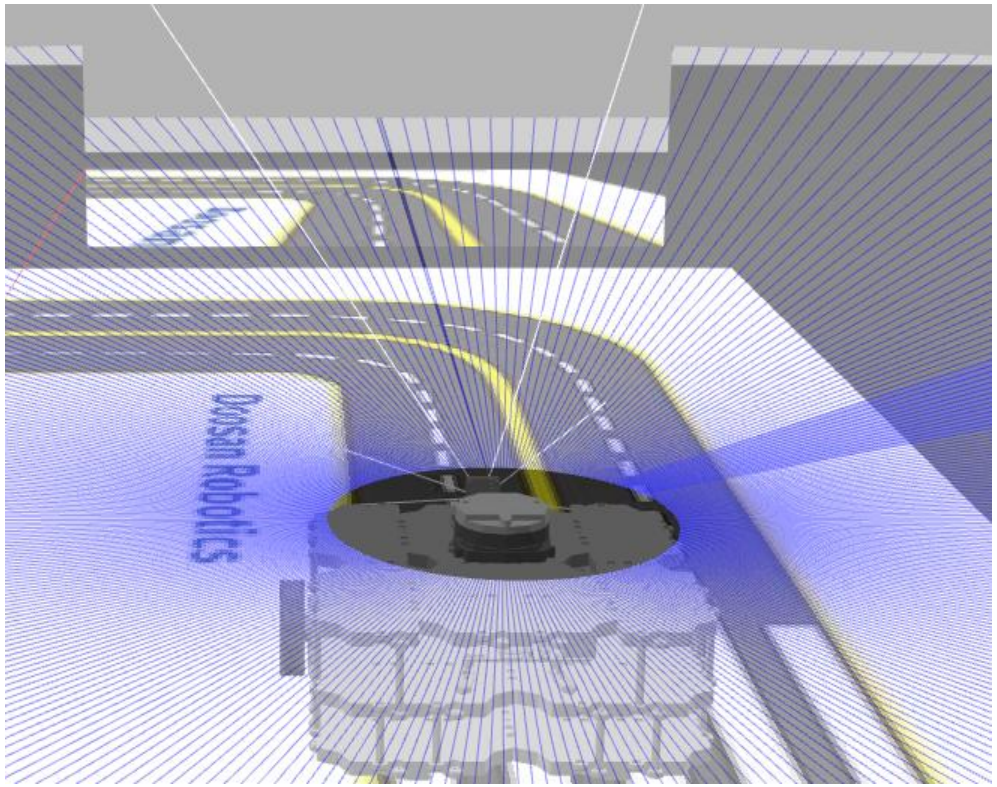


Fig 30. Gazebo simulator

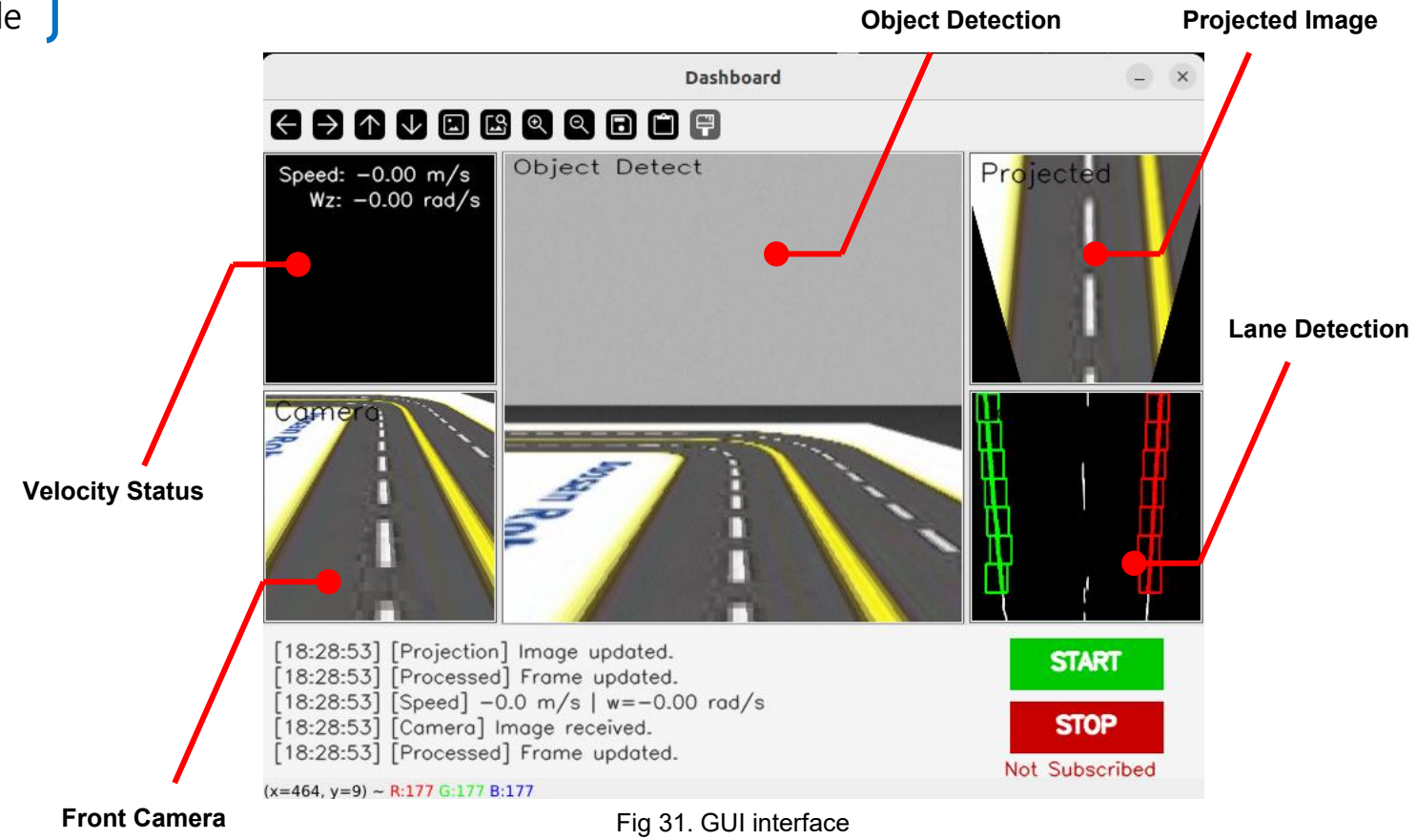


Fig 31. GUI interface

프로젝트 수행 경과

RQt User Interface

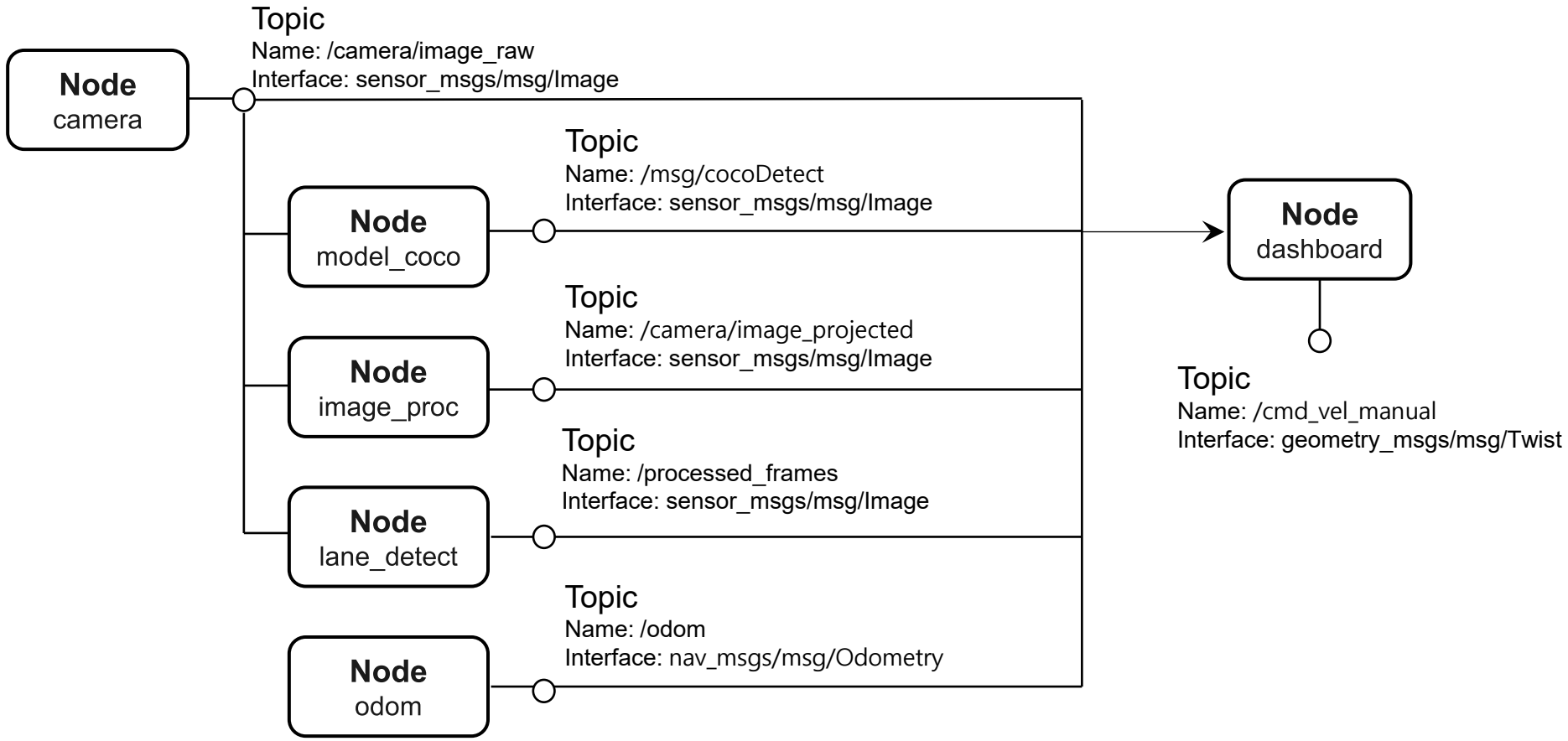
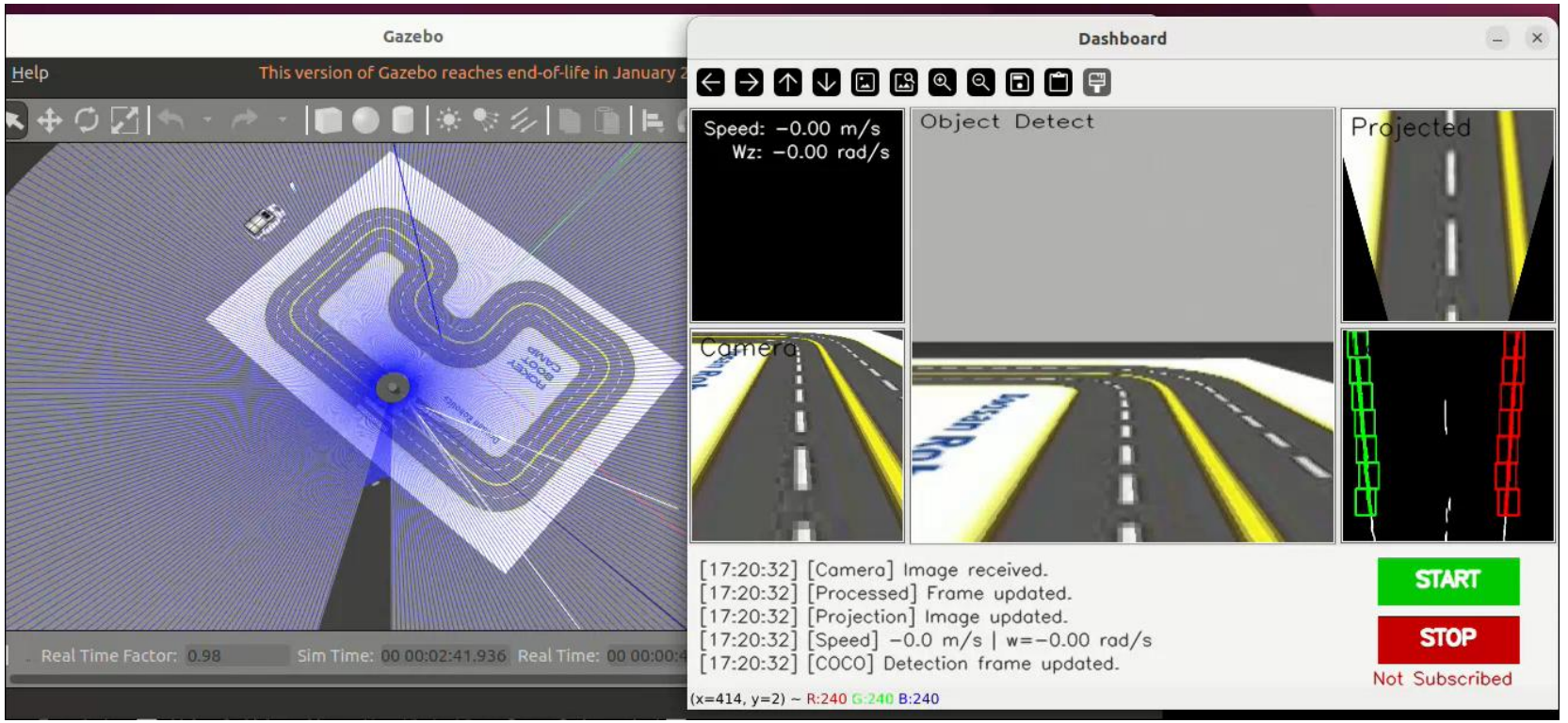


Fig 32. Node architecture of GUI interface

프로젝트 수행 경과

Simulation 결과

- 다양한 차선 색상과 다수의 차선 상황에서 차선 인식 성능 제고
- YOLO V5을 활용한 객체 인식 기술 구현
- Status 확인을 위한 RQt 대쉬보드 설계



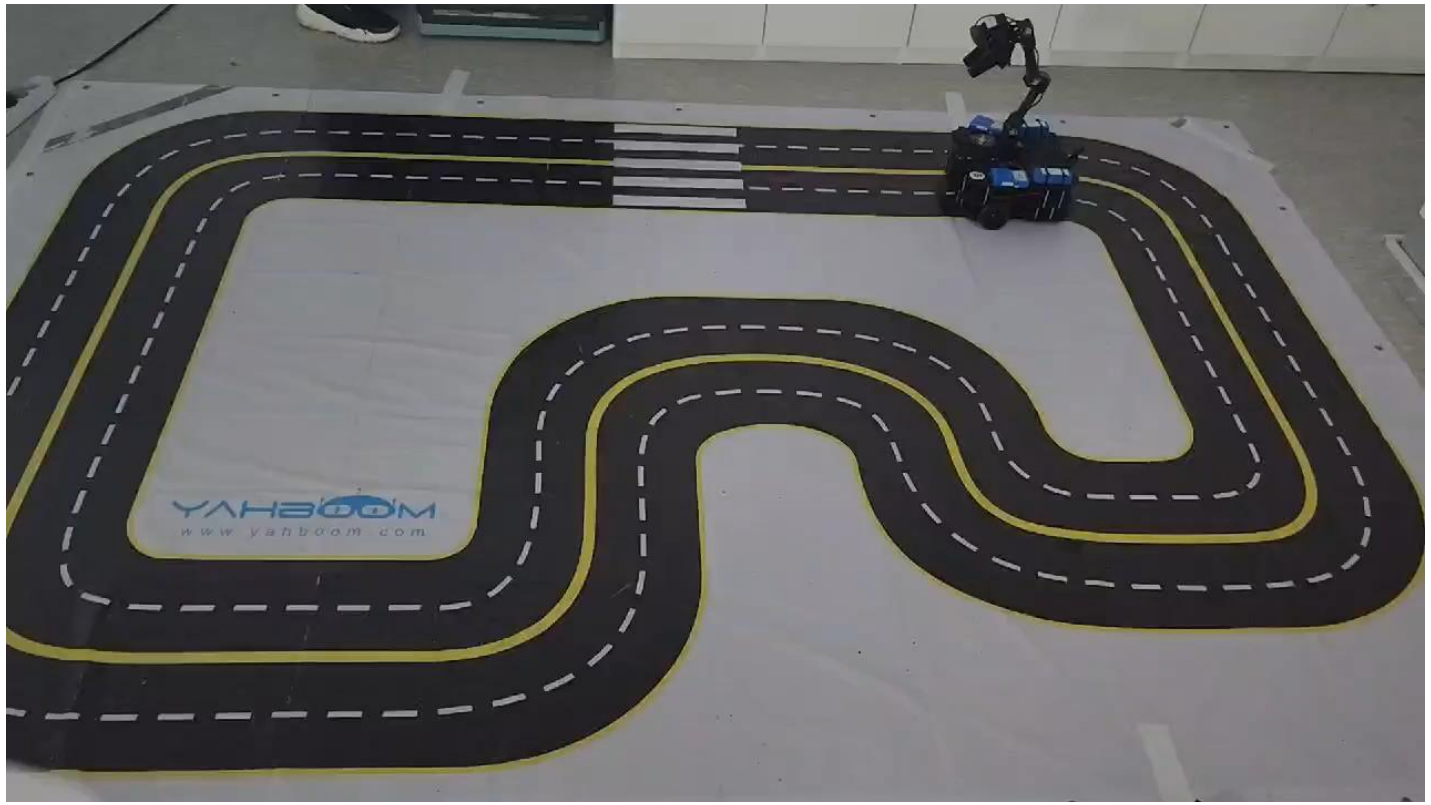
Video 1. Simulation of the improved model

프로젝트 수행 경과

실제 로봇 구동 결과

- 실상황에서 차선 인식 후 한 바퀴 주행 성공
- ArUco marker로 사물 인식 성공
- 사물 인식 시 일시 정지
- 자동으로 Manipulation 움직임 성공

- 노이즈나 광도에 따라 성능이 달라지는 경향이 있으므로 이를 보완할 수 있는 고도화 방안 필요



Video 2. Test of the improved model

05 K-Digital Training 자체 평가 의견

▶ 프로젝트 결과물에 대한 프로젝트 **기획 의도와의 부합 정도** 및 **실무 활용 가능 정도**, **달성도**, **완성도** 등 **훈련생의 자체적인 평가 의견과 느낀 점**을 작성한다.

사전 기획의 관점에서

프로젝트 결과물에 대한 완성도 평가(10점 만점)

- 기존에 기획했던 것만큼의 결과는 나오지 않았지만 더 진행한다면 사전 기획 이상의 결과가 나올 것이라고 확신함
- 10점 만점

개인 또는 우리 팀이 **잘한 부분과 아쉬운 점**

- 팀원들과 업무를 분담해 체계적으로 업무를 수행한 점
- 시간 부족으로 마무리 단계가 아쉬웠던 점

프로젝트 결과물의

추후 개선점이나 보완할 점 등 내용 정리

- 딥러닝을 통한 차선 인식 고도화
- Manipulation 시 ArUco 마커 인식의
- 정확도를 높이고 움직임을 최적화하기

프로젝트를 수행하면서

느낀 점이나 경험한 성과(경력 계획 등과 연관)

- ROS2 프로젝트를 진행하면서 전체적인 진행 과정에 대해 이해할 수 있었음
- Waffle Pi 조작, 차선 인식 등의 다양한 기능을 익힐 수 있었음