

한국IT교육원

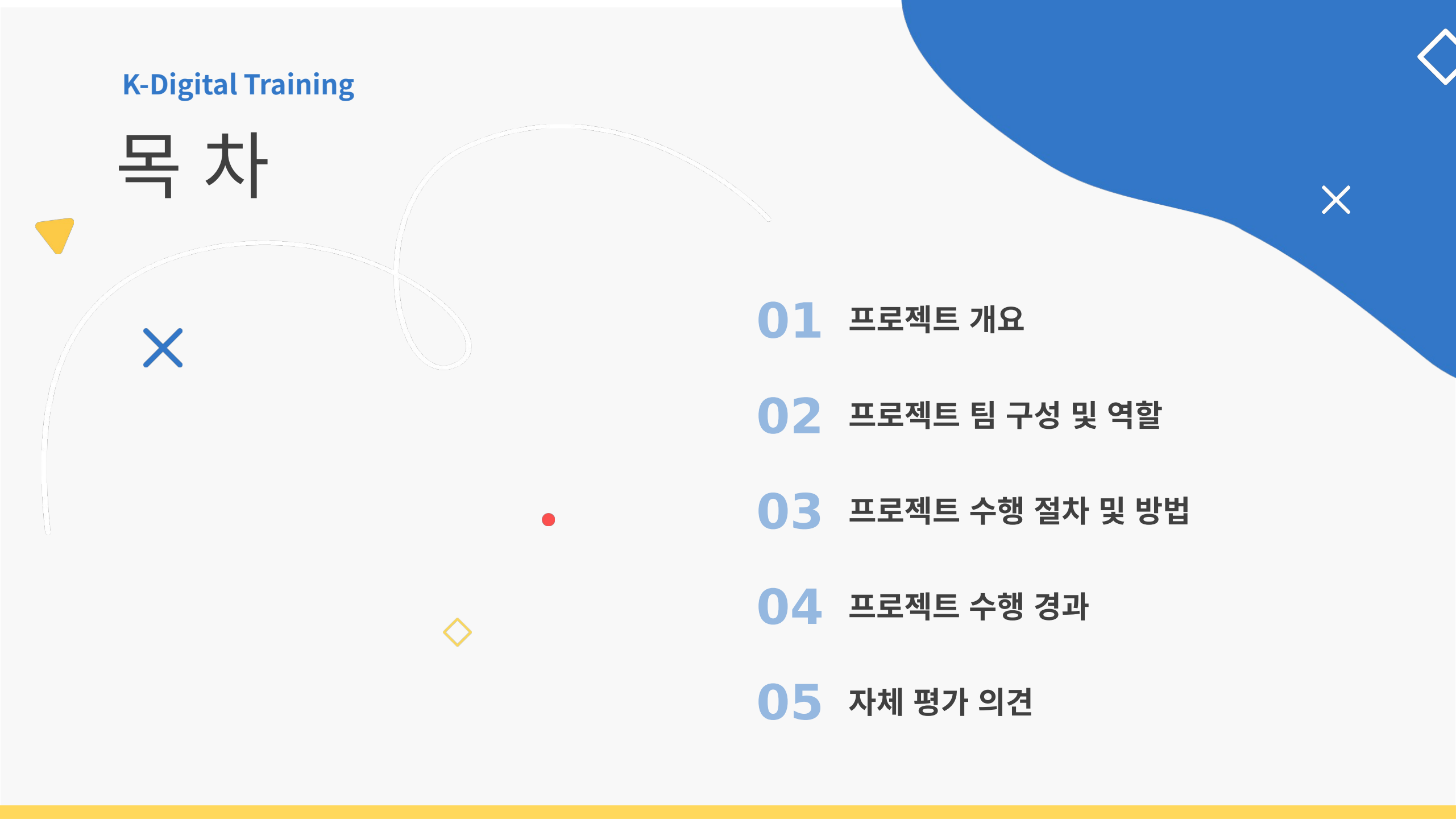
Draco 기술을 이용한 2D SLAM

TEAM 1조 Dr.G

이규호 박태용 이동현 이진석

[멘토] 퓨처드라이브 황종락 본부장

목 차

- 
- 01 프로젝트 개요
 - 02 프로젝트 팀 구성 및 역할
 - 03 프로젝트 수행 절차 및 방법
 - 04 프로젝트 수행 경과
 - 05 자체 평가 의견

1

주제 : Draco기술을 활용한 2D SLAM

1. 선정 배경

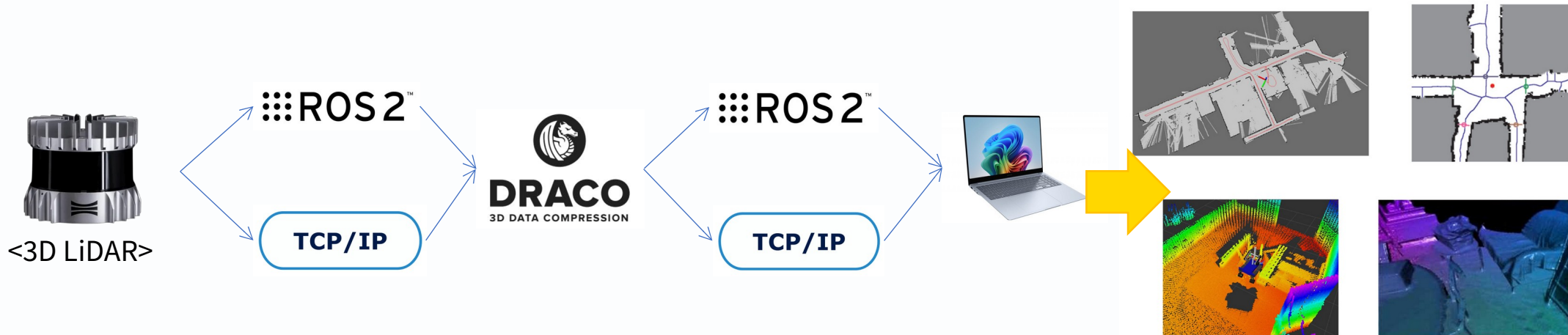
- 3D LiDAR는 대용량의 Pointcloud data를 생성하여 저비용 임베디드 플랫폼에서는 데이터 전송 지연과 연산 부하로 인해 실시간 처리 성능 저하가 발생
- Google의 Draco 압축 기술을 적용하여 저비용 환경에서도 실시간 처리가 가능한 SLAM 구현의 필요성

2. 기획의도

- 3D LiDAR 데이터를 처리하는 과정에서 대용량 Pointcloud data Google Draco 압축 기술을 적용하여 데이터 전송 효율을 높이고, 압축 전·후의 성능(데이터 손실율, 프레임 지연 등)을 비교 분석하고자 함.
- 저비용 임베디드 플랫폼(NVIDIA Jetson 등)에서도 효율적인 SLAM 수행 가능성을 검증함.

1. 프로젝트 내용

- ROS2, TCP/IP 기반 Google Draco 압축 기술을 사용하여 시각화 및 2D SLAM 구현



2. 연관성

- 자율주행 기술 분야에서 가장 핵심적인 인지 기술로써 로봇, 드론과 같은 다양하게 사용되므로 본 과정과 연관성이 깊다.

01

K-Digital Training

프로젝트 개요

3

1. 사용 장비

- Desktop
- 3D LiDAR ouster 32CH
- Laptop

2. 개발환경

- Python, C/C++
- ROS2
- TCP/IP

3. 형상관리

- Github
- Slack

TCP/IP

ROS2™

slack



<3D LiDAR>



1. 프로젝트 진행 일정

월	화	수	목	금
9/29 프로젝트 기획	9/30	10/1 ROS2, TCP/IP기반 Draco통신 구현	10/2	10/3
10/6	10/7	10/8 ROS2, TCP/IP기반 Draco통신 구현	10/9 ROS2, TCP/IP기반 Draco통신 구현	10/10
10/13	10/14	10/15 SLAM 구현	10/16	10/17 현장실습
10/20 프로젝트 PPT	10/21			

1. 활용 방안 및 기대 효과

- 저비용 임베디드 환경에서도 실시간 2D SLAM 구현 가능성 검증
→ Jetson Nano 등 저가 하드웨어에서 3D LiDAR 데이터 처리 효율 향상
- 대용량 포인트 클라우드 데이터의 전송 대역폭 감소
→ Draco 압축 기술 적용을 통해 데이터 전송 효율 향상 및 네트워크 부하 감소
- 지연시간(latency) 단축 및 실시간성 향상
→ 약 100ms 수준으로 전송 지연을 최소화하여 안정적인 자율주행 인지 가능
- SLAM 성능 유지 및 맵 생성 품질 확보
→ 압축 적용 후에도 기존 맵 생성 품질과 유사한 수준 유지 확인
- 자율주행·로봇 분야로의 확장성 확보
→ LiDAR 기반 인지 기술의 경량화로, 향후 로봇/드론/모빌리티 시스템에도 적용 가능

02

K-Digital Training











프로젝트 팀 구성 및 역할

훈련생	역할	담당 업무	
이규호	팀장	 ROS2기반 Draco 통신	 Draco 성능 분석
박태용	팀원	 SLAM	 테스트 수행
이동현	팀원	 TCP/IP기반 Draco 통신	 SLAM
이진석	팀원	 산출물 정리	
홍종락	멘토	 피드백 및 질의응답	

03

K-Digital Training

프로젝트 수행 절차 및 방법

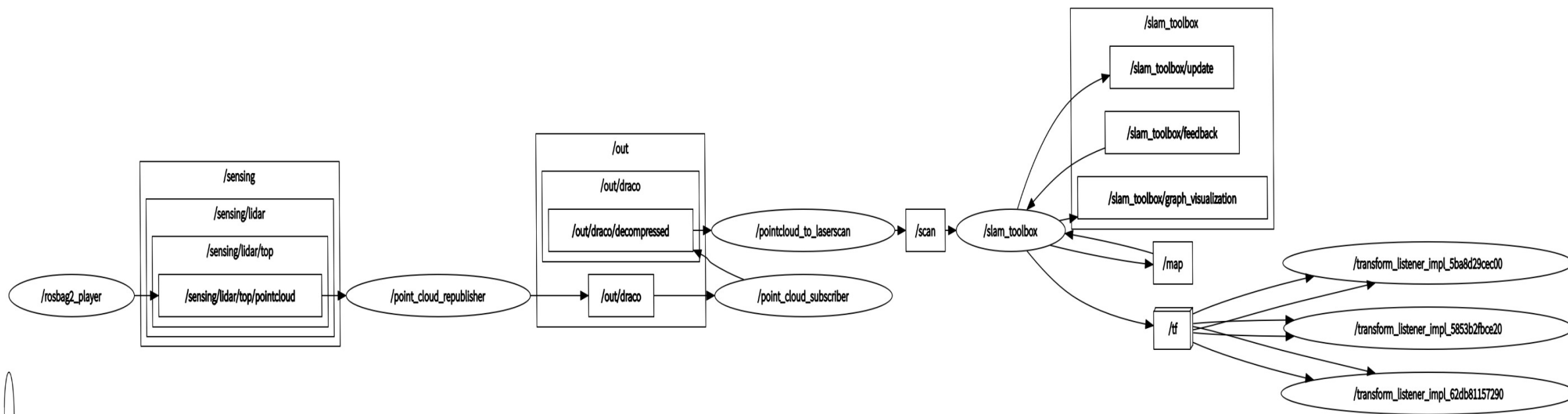
구분	기간	활동		비고
사전 기획	9/29(월) ~ 9/30(금)	 프로젝트 기획 및 주제 선정	 기획안 작성	아이디어 선정
ROS2기반 통신	10/01(월) ~ 10/10(금)	 Pointcloud transport	 Draco plugin	RVIZ2
TCP/IP기반 통신		 Pointcloud to laserscan	 Subscriber	
SLAM	10/13(월) ~ 10/16(목)	 PLY -> DRC		
		 Slamtoolbox	 Kiss icp	맵 생성
총 개발기간	09/29(월) ~ 10/20(월)(총 3주)	 결과보고서		

04

K-Digital Training

프로젝트 수행 경과

▶ Architecture



04

K-Digital Training

프로젝트 수행 경과

▶ 3D LiDAR Data (rosvbag2 file)

※ rosvbag2_bagfile_information

- version : 5
- storage_identifier : sqlite3
- Frequency : 10hz
- duration:
nanoseconds: 43406166756
- starting_time:
nanoseconds_since_epoch: 1727155822969170507
- message_count: 870

항목	/sensing/lidar/top/ pointcloud_raw_ex	/sensing/lidar/top/ pointcloud
메시지 수	435	435
데이터 타입	sensor_msgs/msg/PointCloud2	
QoS-history	KEEP_LAST (depth 100)	KEEP_LAST (depth 10)
QoS-reliability	RELIABLE(1)	BEST_EFFORT(2)
QoS-durability	TRANSIENT_LOCAL	
압축여부	없음	없음
역할 추정	LiDAR 원본	압축 or 변환 후 전송용 데이터

04

K-Digital Training

프로젝트 수행 경과

▶ Point cloud transport

· Point cloud transport pkg

Republish.cpp 실행

-정해진 topic과 통신 방식이 없어 실행 명령어로 지정하여 보냄

```
ros2 run point_cloud_transport republish \  
  --ros-args \  
  -p in_transport:=raw \  
  -p out_transport:=draco \  
  -r in:=/sensing/lidar/top/pointcloud \  
  -r out:=/pct/point_cloud
```

```
#include <memory>  
#include <string>  
#include <utility>  
  
#include <rclcpp/rclcpp.hpp>  
  
#include <pluginlib/class_loader.hpp>  
#include <sensor_msgs/msg/point_cloud2.hpp>  
  
#include "point_cloud_transport/exception.hpp"  
#include "point_cloud_transport/point_cloud_transport.hpp"  
#include "point_cloud_transport/publisher.hpp"  
#include "point_cloud_transport/publisher_plugin.hpp"  
#include "point_cloud_transport/republish.hpp"  
#include "point_cloud_transport/subscriber.hpp"  
  
using namespace std::chrono_literals;  
  
namespace point_cloud_transport  
{  
  Republisher::Republisher(const rclcpp::NodeOptions & options)  
  : Node("point_cloud_republisher", options)  
  {  
    // Initialize Republishercomponent after construction  
    // shared_from_this can't be used in the constructor  
    this->timer_ = create_wall_timer(  
      1ms, [this]() {  
        if (initialized_) {  
          timer_>cancel();  
        } else {  
          this->initialize();  
          initialized_ = true;  
        }  
      });  
  }  
  
  void Republisher::initialize()  
  {  
    std::string in_topic = rclcpp::expand_topic_or_service_name(  
      "in",  
      this->get_name(), this->get_namespace());  
  
    std::string out_topic = rclcpp::expand_topic_or_service_name(  
      "out",  
      this->get_name(), this->get_namespace());  
  
    std::string in_transport = "raw";  
    this->declare_parameter<std::string>("in_transport", in_transport);  
    if (!this->get_parameter(  
      "in_transport", in_transport))  
    {  
      RCLCPP_WARN_STREAM(  

```

https://github.com/ros-perception/point_cloud_transport/tree/humble

04

K-Digital Training

프로젝트 수행 경과

▶ Draco point cloud plugin

· Draco point cloud plugin

sudo apt install ros-humble-point-cloud-draco

rqt → configuration → dynamic reconfigure

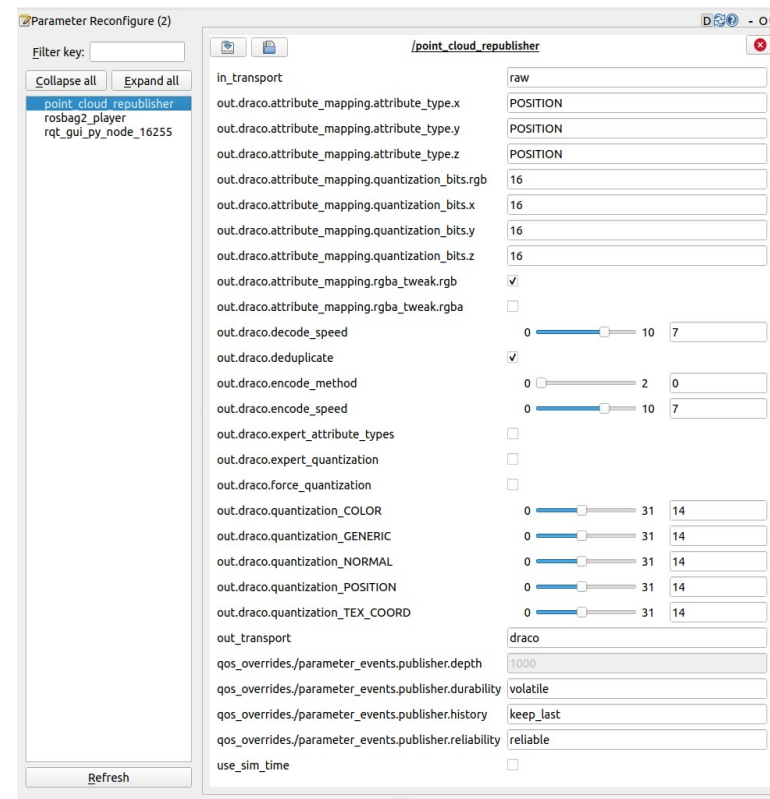
```

GNU nano 6.2 draco_plugins.xml
<library path="draco_point_cloud_transport">
  <class
    name="point_cloud_transport/draco_pub"
    type="draco_point_cloud_transport::DracoPublisher"
    base_class_type="point_cloud_transport::PublisherPlugin">
    <description>
      This plugin publishes a CompressedPointCloud2 using KD tree compression.
    </description>
  </class>

  <class
    name="point_cloud_transport/draco_sub"
    type="draco_point_cloud_transport::DracoSubscriber"
    base_class_type="point_cloud_transport::SubscriberPlugin">
    <description>
      This plugin decompresses a CompressedPointCloud2 topic.
    </description>
  </class>
</library>

```

<opt/ros/humble>



<rqt>

▶ Point cloud transport tutorial

· my_subscriber.cpp

-transportHints hints (“draco”) draco plugin 사용

-/out topic 구독할 때

-“draco” transport가 자동으로 압축 해제 처리

- pointcloudcallback 함수에서 sensor_msgs::msg::PointCloud2 타입으로 복원된 데이터 전달

- 포인트 수 계산, 해제된 topic 발행

```
#include <point_cloud_transport/point_cloud_transport.hpp>
#include <rclcpp/rclcpp.hpp>
#include <sensor_msgs/msg/point_cloud2.hpp>
#include <point_cloud_transport/transport_hints.hpp>

// 콜백 함수: PointCloud2를 로그로 확인하고, RViz2용으로 publish
void pointCloudCallback(
  const sensor_msgs::msg::PointCloud2::ConstSharedPtr & msg,
  rclcpp::Logger logger,
  rclcpp::Publisher<sensor_msgs::msg::PointCloud2>::SharedPtr pub)
{
  // 포인트 수 로그 출력
  RCLCPP_INFO_STREAM(logger, "Message received, number of points is: " << msg->width * msg->height);

  // RViz2에서 구독할 수 있는 새 토픽으로 publish
  pub->publish(*msg);
}

int main(int argc, char ** argv)
{
  rclcpp::init(argc, argv);
  auto node = std::make_shared<rclcpp::Node>("point_cloud_subscriber");

  // PointCloudTransport 초기화
  point_cloud_transport::PointCloudTransport pct(node);

  // Draco transport 명시
  const point_cloud_transport::TransportHints hints("draco");

  // RViz2용 publisher 추가
  auto pub = node->create_publisher<sensor_msgs::msg::PointCloud2>("/out/draco/decompressed", 5);

  // subscriber 생성, 콜백에서 publisher 전달
  point_cloud_transport::Subscriber pct_sub = pct.subscribe(
    "out", 10,
    std::bind(pointCloudCallback, std::placeholders::_1, node->get_logger(), pub),
    nullptr,
    hints);

  RCLCPP_INFO_STREAM(node->get_logger(), "Waiting for point_cloud message...");

  rclcpp::spin(node);
  rclcpp::shutdown();

  return 0;
}
```

```
hkit@hkit-desktop:~/futuredrive_ws$ ros2 run point_cloud_transport_tutorial subscriber_test
[INFO] [1760932083.845890375] [point_cloud_subscriber]: Subscribing to: /out/draco

[INFO] [1760932083.845955592] [point_cloud_subscriber]: Waiting for point_cloud message...
[INFO] [1760932103.377157332] [point_cloud_subscriber]: Message received, number of points is: 130323
[INFO] [1760932103.450598044] [point_cloud_subscriber]: Message received, number of points is: 130267
[INFO] [1760932103.524472059] [point_cloud_subscriber]: Message received, number of points is: 130372
[INFO] [1760932103.597526820] [point_cloud_subscriber]: Message received, number of points is: 130335
[INFO] [1760932103.799013778] [point_cloud_subscriber]: Message received, number of points is: 130350
[INFO] [1760932103.901073638] [point_cloud_subscriber]: Message received, number of points is: 130339
[INFO] [1760932104.001219738] [point_cloud_subscriber]: Message received, number of points is: 130306
[INFO] [1760932104.095890473] [point_cloud_subscriber]: Message received, number of points is: 130373
[INFO] [1760932104.199310093] [point_cloud_subscriber]: Message received, number of points is: 130343
[INFO] [1760932104.294898437] [point_cloud_subscriber]: Message received, number of points is: 130277
[INFO] [1760932104.389892417] [point_cloud_subscriber]: Message received, number of points is: 130284
[INFO] [1760932104.492215688] [point_cloud_subscriber]: Message received, number of points is: 130415
[INFO] [1760932104.589354147] [point_cloud_subscriber]: Message received, number of points is: 130317
[INFO] [1760932104.692474995] [point_cloud_subscriber]: Message received, number of points is: 130391
[INFO] [1760932104.790207644] [point_cloud_subscriber]: Message received, number of points is: 130333
[INFO] [1760932104.888117101] [point_cloud_subscriber]: Message received, number of points is: 130304
```

https://github.com/ros-perception/point_cloud_transport_tutorial

04

K-Digital Training

프로젝트 수행 경과

▶ Draco point cloud plugin

· Draco point cloud plugin

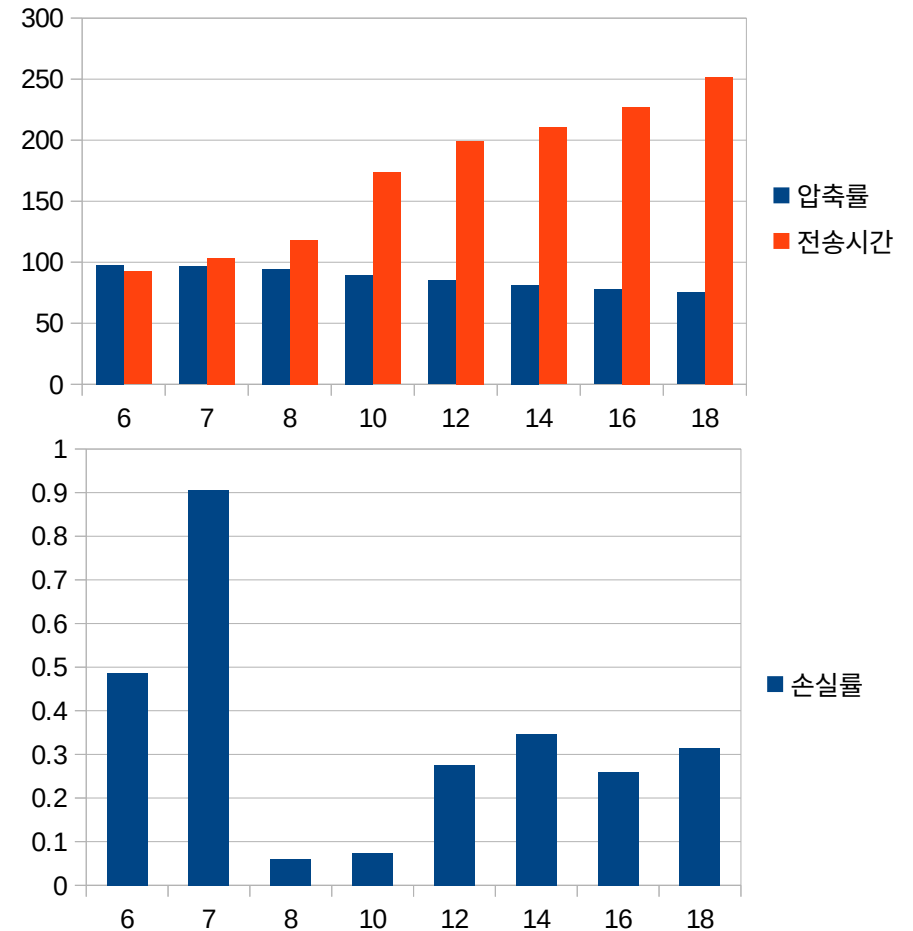
encode_speed:1 (고정) 압축률 최대, 압축 속도 느림

decode_speed:10 (고정) 디코딩 속도 최대, 실시간 처리 우선

quantization_POSITION:6~18

-point의 좌표(x,y,z) 실수라서 정수로 변환하여 비트 수(2^x)로 나누는 단계

양자화 비트수		FPS		대역폭(MB/s)		압축률(%)		손실률(%)		지연시간(ms)	
		평균	표준편차	평균	표준편차	평균	표준편차	평균	표준편차	평균	표준편차
6	Uncompressed	9.9275	0.6461	19.2644	1.3566	97.8284	0.6533	0.4874	2.2317	92.476	14.869
	Compressed	9.9764	0.471	0.4118	0.0715						
	Decompressed	9.6804	1.3983	18.6964	2.7752						
7	Uncompressed	9.9422	0.5638	19.3548	1.1801	96.4103	0.9692	0.9058	3.0064	103.567	71.296
	Compressed	9.9754	0.481	0.6869	0.1177						
	Decompressed	9.4878	1.6776	18.2982	3.2896						
8	Uncompressed	9.9211	0.6659	19.1477	1.4251	94.4062	1.4127	0.0603	0.7129	118.272	71.732
	Compressed	9.9783	0.4528	1.0573	0.1565						
	Decompressed	9.9783	0.4528	19.2448	1.0604						
10	Uncompressed	9.943	0.5823	19.2379	1.2582	89.7221	2.0813	0.074	0.9196	173.5	78.034
	Compressed	9.8176	0.9907	1.9647	0.2745						
	Decompressed	9.9775	0.4605	19.2892	1.0505						
12	Uncompressed	9.9252	0.6562	19.3029	1.3598	84.9564	3.7893	0.2759	1.575	199.305	83.534
	Compressed	9.5067	1.6231	2.8763	0.5144						
	Decompressed	9.8296	0.9663	19.0639	1.9517						
14	Uncompressed	9.9218	0.6481	19.3513	1.3364	81.3899	5.2819	0.3465	2.3055	210.467	34.366
	Compressed	8.9582	2.1797	3.5755	0.872						
	Decompressed	9.4433	1.6127	18.3457	3.1524						
16	Uncompressed	9.9167	0.669	19.3169	1.3732	77.654	7.6602	0.2602	2.4304	227.001	35.164
	Compressed	8.7014	2.4277	4.2654	1.1954						
	Decompressed	9.0460	1.9996	17.5664	3.8981						
18	Uncompressed	9.9278	0.6290	19.3099	1.2908	75.1058	8.4165	0.3143	2.5596	251.783	99.233
	Compressed	8.2181	2.597	4.7839	1.5157						



04

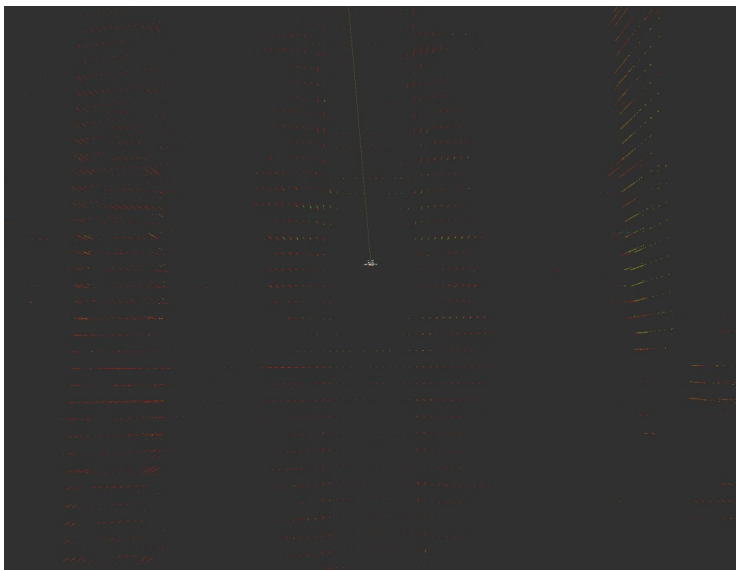
K-Digital Training

프로젝트 수행 경과

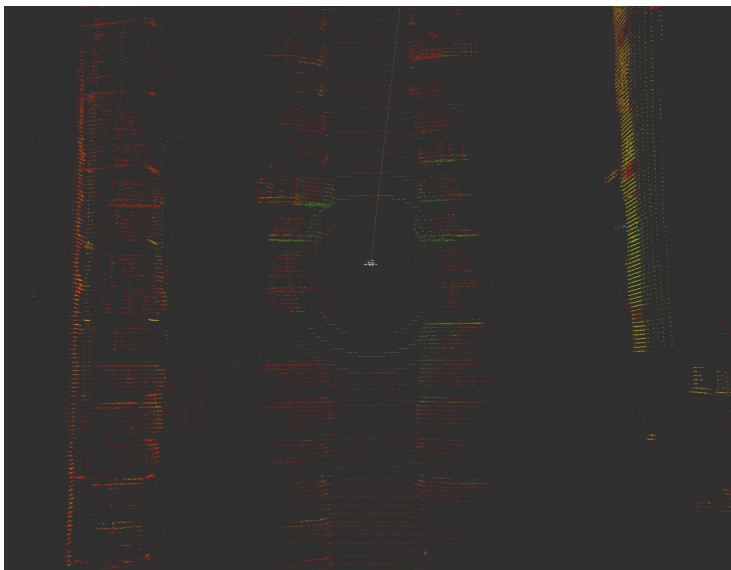
▶ RVIZ2

· 시각화(Visualization)

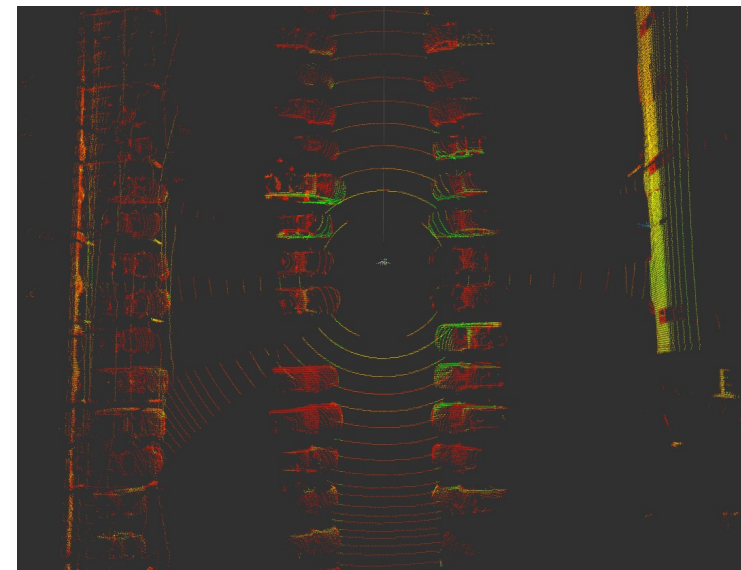
-/out/draco/decompressed



<quantization.POSITION = 8>



<quantization.POSITION = 10>



<quantization.POSITION = 12>

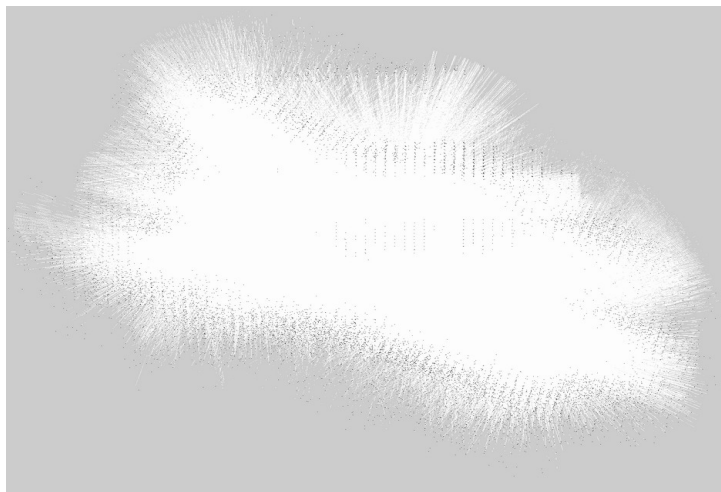
▶ SLAM(Simultaneous Localization And Mapping)

• 3D LiDAR Data → 2D laserscan

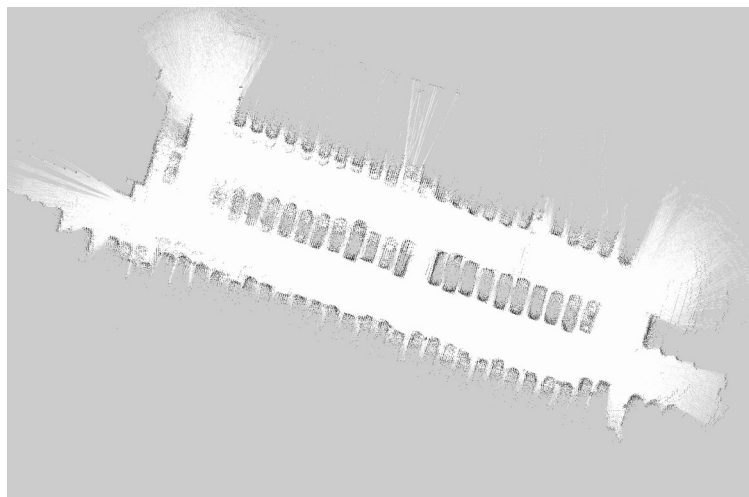
```
-sudo apt install ros-humble-pointcloud-to-laserscan (pkg)
```

```
-ros2 run pointcloud_to_laserscan pointcloud_to_laserscan_node -ros-args -r cloud_in:=/out/draco/decompressed -r scan:=/scan -p target_frame:=base_link -p transform_tolerance:=0.01 -p min_height:=0.0 -p max_height:=2.0 -p angle_min:=3.14 -p angle_max:=3.14 -p angle_increment:=0.0087 -p range_min:=0.1 -p range_max:=50.0 -p qos_overrides/cloud_in.subscription.reliability:=best_effort -p qos_overrides/scan.publisher.reliability:=best_effort
```

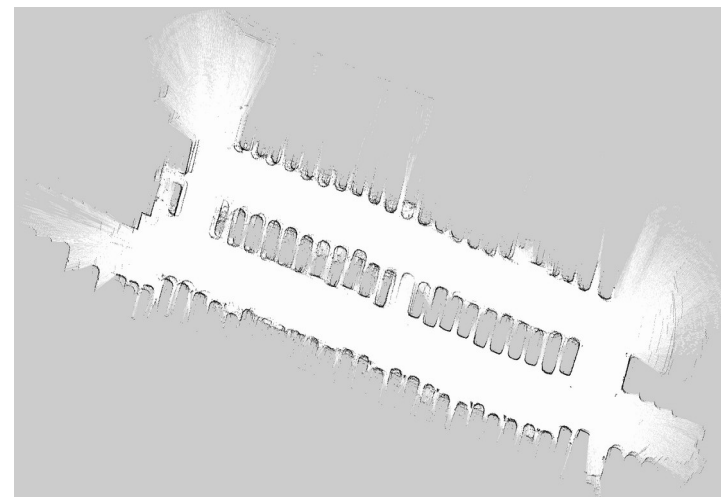
• 2D SLAM



<quantization.POSITION = 8>



<quantization.POSITION = 10>



<quantization.POSITION = 12>

▶ Result

본 프로젝트는 실시간 전송 성능 향상을 목표로 양자화 비트 수(Quantization Bits)를 조정하여 Draco 기반 압축 성능을 측정하였다.

- 실험 결과 8bit, 10bit, 12bit 구간에서 성능 차이가 뚜렷하게 나타났다.
- 8bit: 전송 속도는 가장 빠르나, 포인트 손실로 인해 시각적 품질 저하가 발생하였다.
- 10bit: 전송시간 약 170ms으로 품질과 속도 간 균형이 가장 적절하였다.
- 12bit: 가장 높은 품질을 보였으나, 전송시간이 200ms 이상으로 증가하여 실시간성 측면에서는 다소 부적합하였다.
- 결론적으로 실시간성을 중시하는 시스템에서는 8bit 설정이 가장 적합한 것으로 판단된다.

04

K-Digital Training

프로젝트 수행 경과

▶ 시연영상



05

K-Digital Training

자체 평가 의견

- ▶ 프로젝트 결과물에 대한 프로젝트 기획 의도와 의 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)
9점

개인 또는 우리 팀이 **잘한 부분과 아쉬운 점**

예 성능 평가 결과 분석이 체계적이지 못한 것
전송시간과 압축률, 대역폭, FPS를 한 번에
측정하였으면 더 정확한 결과가 나왔을 것
같습니다.

프로젝트 결과물의
추후 개선점이나 보완할 점 등 내용 정리
Draco parameter 조정에 따른 성능 분석을 통해 더 정확한 분석을
하여 기술을 발전할 필요가 있다고 생각합니다.

프로젝트를 수행하면서
느낀 점이나 경험한 성과(경력 계획 등과 연관)
처음으로 센서 Data를 이용하여 새로운 기술을 개발해보면서 기술적
역량을 쌓을 수 있었습니다.