데이터셋

- 수집
  - 기존 1차 라파코드에 카메라 라이브러리 추가해서 사용해보자.
    - 라파에 CAN HAT쓸때 설정해줘야함.
      - 설치 RS485 CAN HAT - Waveshare Wiki
      - ```
        sudo ip link set can0 up type can bitrate 1000000 &&
        sudo ifconfig can0 txqueuelen 65536 &&
        sudo ifconfig can0 up
        ```
    - sudo apt-get install dphys-swapfile 하고 swap메모리 늘려주자. 1차 라파 빌드할때 메모리 초과남.
    - 빌드
      - 1차라파코드의 dkbuild/ssv에서 dockerfile로 이미지 만들어서 컨테이너 만들던가
      - 아님 기존 컨테이너에 빌드해야함
        - 그냥 colcon build --symlink-install하면 의존성 패키지 없어서 에러남
          - rosdep으로 설치하던가
          - 직접 설치 (순서문제인지 직접설치해서 테스트 결과 joystick출력 순서가 바뀌는 문제가있음. docker 권장)
            sudo apt-get install ros-humble-joy
            sudo apt-get install ros-humble-ros2-socketcan
    - dkrun/ssv.sh 로 실행하던가
      ros2 launch skid_steer_vehicle bringup.launch.xml 로 실행하면 컨트롤러 노드 수행됨.
  - 카메라 토픽과 컨트롤러 입력 토픽을 한 노드가 받아서 묶어서 다시 토픽 출력하면 그걸 record 해서 rosbag으로 저장하여 동기화를 수행하자.
    - LIBCAMERA_RPI_TUNING_FILE="/root/ws/cam_test/src/libcamera/src/ipa/rpi/pisp/data/imx219_noir.json" ros2 run camera_ros camera_node --ros-args -p format:=BGR888 -p sensor_mode:="3280:2464" -p width:=320 -p height:=240
      - 표지판 촬영용
      - 이렇게 하니 320 240 안하는 것과 해상도만 차이남.
    - ros2 run camera_ros camera_node --ros-args -p format:=BGR888 -p sensor_mode:="3200:2400" -p width:=320 -p height:=240 -p orientation:=180
      - 만약 camera 바꾸고 /dev/cam0이 안생기고 camera 못찾으면
        ```

        - sudo vim /boot/firmware/config.txt 에 dtoverlay=imx219 추가 및 재부팅.
    - ros2 bag record -o imgchk /sync_republisher_node/image_synced /sync_republisher_node/joy_synced
  - 데이터 파싱
    - bag2img_control.py
- tflite 테스트
  - tensorflow c++ 라이브러리 빌드
    - TensorFlow Lite C++で量子化モデルをRaspberry Pi 4で動かす - 土日の勉強ノート 여기서 버전만 git checkout r2.15쓰면 될 것.
    - git clone https://github.com/tensorflow/tensorflow.git
    - 폴더로 들어가서 git checkout r2.15
    - git submodule update --init --recursive
    - mkdir tflite_build && cd tflite_build
    - cmake ../tensorflow/lite
    - cmake --build . -j $(nproc)
  - 샘플 테스트
    - 참고
      - Install TensorFlow 2 Lite on Raspberry 64 OS - Q-engineering
      - tensorflow/tensorflow/lite/examples/label_image at master · tensorflow/tensorflow
    - 모델 및 데이터 다운 (/tmp폴더에 저장됨)
      - ```
        # Get model
        curl https://storage.googleapis.com/download.tensorflow.org/models/mobilenet_v1_2018
        
        # Get labels
        curl https://storage.googleapis.com/download.tensorflow.org/models/mobilenet_v1_1.0_
        
        mv /tmp/mobilenet_v1_1.0_224/labels.txt /tmp/
        ```
      - root@geunu:/tensorflow/tflite_build# examples/label_image/label_image -m /tmp/mobilenet_v1_1.0_224.tflite -l /tmp/labels.txt -i /tensorflow/tensorflow/lite/examples/label_image/testdata/grace_hopper.bmp 수행
        - ERROR: failed to get XNNPACK profile information. 뜨긴하는데 무시해도 된다고 하긴 함.
  - ros2 노드 만들기
    - CMakeLists.txt
      - ```
        # CMakeLists.txt

        cmake_minimum_required(VERSION 3.8)
        ```

```
project(lkas_test) # <-- (프로젝트 이름이 lkas_test라고 가정)

# --- 1단계: ROS 2 패키지 찾기 ---
find_package(ament_cmake REQUIRED)
find_package(rclcpp REQUIRED)
find_package(std_msgs REQUIRED)
find_package(sensor_msgs REQUIRED)
find_package(message_filters REQUIRED)
find_package(cv_bridge REQUIRED)
find_package(image_transport REQUIRED)
find_package(OpenCV REQUIRED)

# ============================================================
# 👇 TFLite 'find_package' 삭제
# ============================================================
# find_package(tensorflow-lite REQUIRED) # <-- 이 줄을 삭제하거나 주석 처리!


# ============================================================
# 👇 여기에 TFLite를 직접 빌드하는 코드 추가
# ============================================================
# TFLite 소스 코드 경로 (절대 경로 사용)
set(TFLITE_SOURCE_DIR /tensorflow/tensorflow/lite)

# TFLite 빌드 옵션 설정 (설치/예제 등 불필요한 기능 끄기)
option(TFLITE_ENABLE_INSTALL "Enable install rule" OFF)
option(TFLITE_ENABLE_LABEL_IMAGE "Enable label_image example" OFF)
option(TFLITE_ENABLE_BENCHMARK_MODEL "Enable the benchmark_model tool" OFF)
option(TFLITE_ENABLE_XNNPACK "Enable XNNPACK" ON) # <-- 성능을 위해 켜는 것을 권장

# TFLite 소스 폴더를 하위 디렉터리로 추가 (TFLite 빌드 시작)
# TFLite의 빌드 파일은 ROS 2 빌드 폴더 내부에 생성되도록 설정
add_subdirectory(
  ${TFLITE_SOURCE_DIR}
  ${CMAKE_CURRENT_BINARY_DIR}/tflite_build
)
# ============================================================


# --- 2단계: 실행 파일 정의 ---
# (src/lkas_test.cpp 경로에 맞게 수정)
add_executable(lkas_node src/lkas_test.cpp) # <-- (실행 파일 이름이 lkas_node라고 가정)

# --- 3단계: ROS 2 의존성 연결 ---
ament_target_dependencies(lkas_node
  rclcpp
  std_msgs
  sensor_msgs
  message_filters
  cv_bridge
  image_transport
)

# --- 4단계: 외부 라이브러리 링크 ---
target_link_libraries(lkas_node
  ${OpenCV_LIBRARIES}
  # 👇 'tensorflow-lite' 타겟 링크 (이름은 동일함)
  #    find_package 대신 add_subdirectory가 이 타겟을 생성함
```

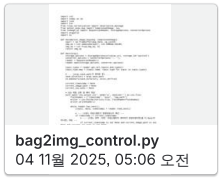
```
  tensorflow-lite
)

# --- 5단계: 설치 및 패키징 ---
install(TARGETS
  lkas_node
  DESTINATION lib/${PROJECT_NAME}
)

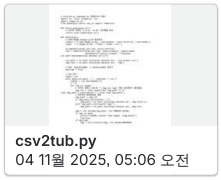
ament_package()
```

- colcon build --packages-select lkas_test (30~40분 걸림)

여기를 클릭하여 펼치기...



bag2img_control.py
04 11월 2025, 05:06 오전



csv2tub.py
04 11월 2025, 05:06 오전