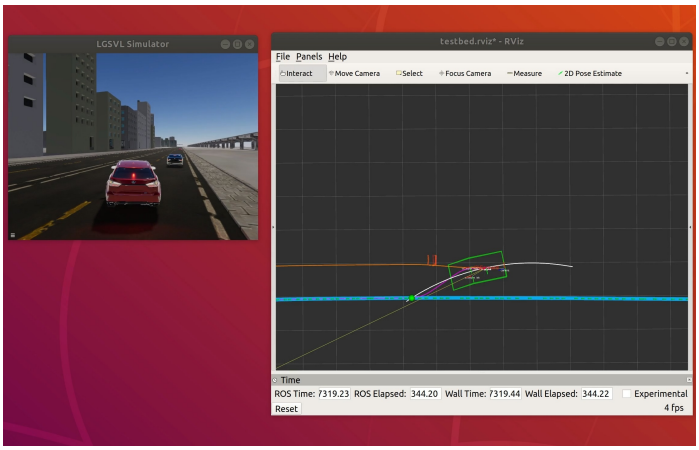
1. Autoware on Embedded Board [Source Code]



자율주행의 상용화를 위해서는 값비싼 workstation이 아닌 성능은 떨어지지만 값싼 embedded board에서의 자율주행 로직 수행이 필요합니다. 이러한 목적을 달성하고자 Nvidia TX2, Nvidia AGX Xavier Board에서 open-source 자율주행 SW인 Autoware를 실행하고자 연구실 사람들과 진행한 프로젝트입니다. Autoware라는 규모가 큰 코드를 분석하며 자율주행의 stack을 이해하고, 필요한 부분을 수정하거나 새로운 ROS 모듈들을 구현했습니다. 최종적으로 Nvidia TX2, Nvidia AGX Xavier Board에서 Autoware를 실행할 수 있었으며 SVL Simulator, 미니카, 실차에서 모두 자율주행에 성공했습니다. Autoware를 분석한 경험을 토대로 **2021 현대 자율주행 챌린지**에도 참여한 경험이 있습니다.

What I do

Interfacing

- TX2, Xavier board에 Ubuntu 포팅
- Velodyne VLP-16 LiDAR와의 interfacing
- Multi LiDAR fusion
- 미니카의 brushless motor와 FocBox ESC 연결
- SVL simulator와 Autoware의 interfacing

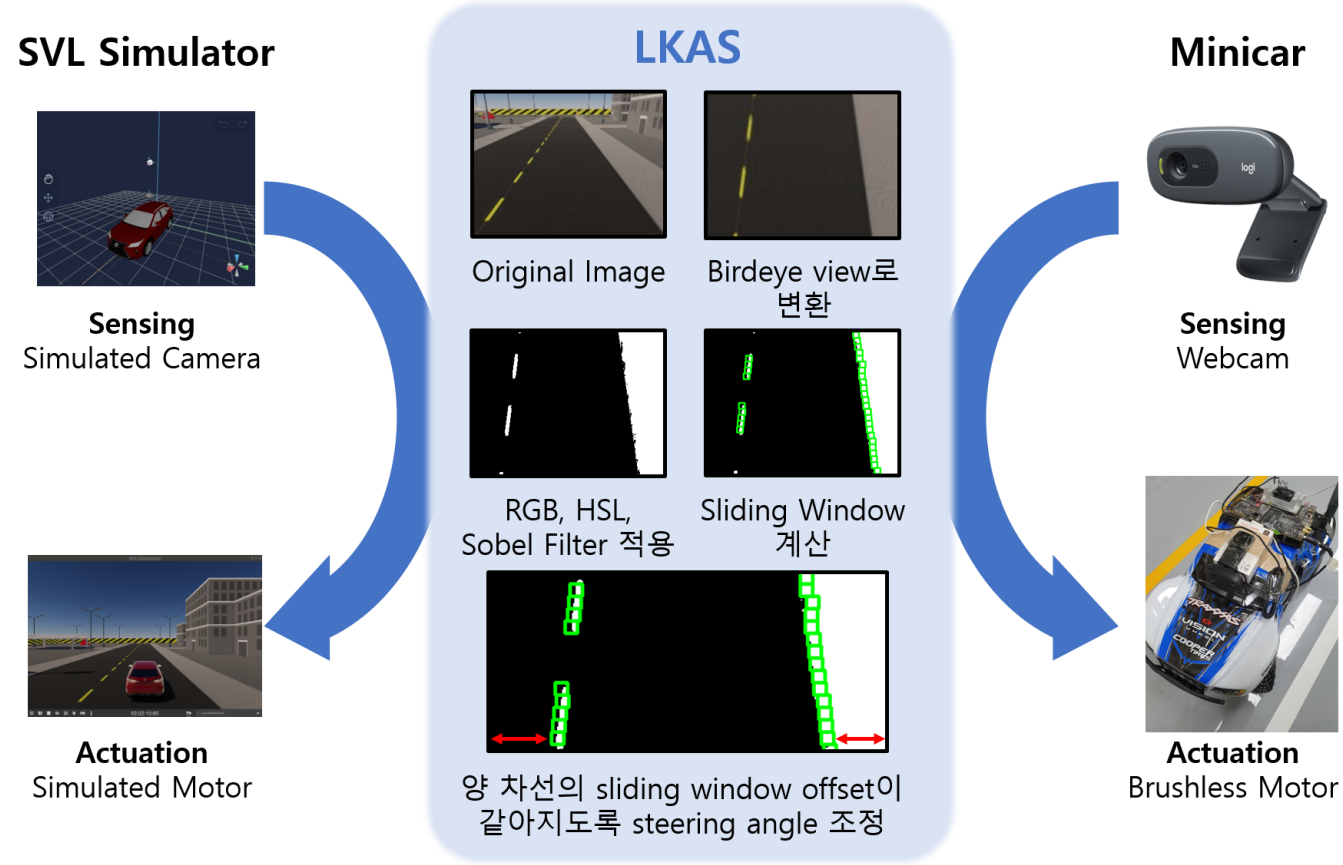
Autoware Logic 외의 자율주행을 위한 task

- `NDT matching` 분석 및 미니카, 실차, simulator에 맞는 localization parameter tuning
- Autoware에서 사용하는 제어 알고리즘인 `pure pursuit`의 parameter tuning
- Point map 및 vector map 제작
- Autoware stack을 순서에 맞게 자동으로 실행하는 ROS node 구현
- ROS, Autoware, dependency가 있는 library들을 한번에 설치하는 shell script 작성
- 키보드를 통한 미니카 주행 ROS node 구현

Autoware Logic 추가 및 수정

- `NDT matching`의 localization 실패 시 GNSS 위치 정보를 바탕으로 복구 알고리즘 추가
- 주변 차량에 따른 회피 알고리즘 개선
- GNSS sensor에서 받아온 정보를 차량의 좌표계로 변환하는 ROS node 구현
- V2X topic을 통해 교통 신호에 맞게 정지하는 알고리즘 추가
- 원형 트랙을 반복해서 주행하게 하는 알고리즘 추가
- 곡률 기반 감속 알고리즘 추가

2. Vision Lane Keeping [Source Code]



Autoware가 localization 실패 등의 에러에 의해 vector map을 따라 자율주행할 수 없는 경우에도 정상적으로 주행할 수 있도록, vision sensor에서 받아온 영상 데이터를 바탕으로 차선을 인식하여 차선의 중앙으로 차량이 계속해서 주행할 수 있도록 하기 위해 진행한 프로젝트입니다. OpenCV를 사용하여 ROS node 형태로 구현하였으며 구현의 모든 부분을 제가 담당했습니다. 구현한 프로그램을 통해 SVL Simulator와 미니카에서 성공적으로 주행할 수 있음을 확인했습니다.

What I do

- OpenCV 이용 birdeye view 변환 및 RGB, HSL, Sobel Filter 구현
- sliding window에 따른 고려한 차선 유지 logic 고안 및 구현
- SVL simulator와 Autoware의 interfacing
- Webcam 및 FocBox ESC와의 interfacing

Autoware + LKAS [Source Code]

프로젝트 1에서 구현한 Autoware 코드에 LKAS를 적용시켜 `NDT matching`에서 localization이 실패하는 경우에 safety-backup인 LKAS를 통해 주행할 수 있도록 했습니다. 이 때 주어진 자율주행 stack에서 `NDT matching`의 최대 수행 시간을 계산하는 analysis를 제안하는 연구를 진행했습니다. 연구를 통해 localization 실패 상황에서도 안전하게 주행할 수 있음을 이론과 실험을 통해 성공적으로 보였으며 이를 바탕으로 작성한 논문이 올해 5월 SCI급 학회인 RTAS 2022(28th IEEE Real-Time and Embedded Technology and Applications Symposium)에 publish될 예정입니다.

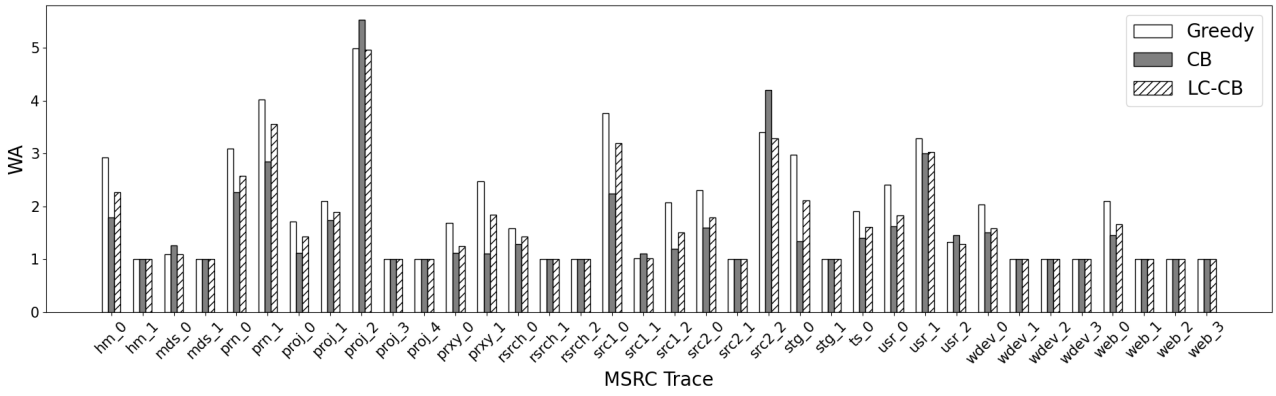
3. WASim [Source Code]

대학원 과목인 '고급 운영체제' 과목의 프로젝트에서 진행한 SSD 관련 연구를 위해 SSD Simulator를 찾던 중 기존의 SimpleSSD, MQSim 등이 재현이 잘 되지 않는 문제가 있어 연구에 필요한 새로운 SSD Simulator를 구현했습니다. Python으로 Flash Translation Layer (FTL)의 동작을 구현하여 Garbage Collection (GC)을 진행할 때 policy 별로 WA (Write Amplification)을 비교할 수 있는 simulator를 구현했습니다. 모든 구현과 실험을 제가 담당했습니다.

What I do

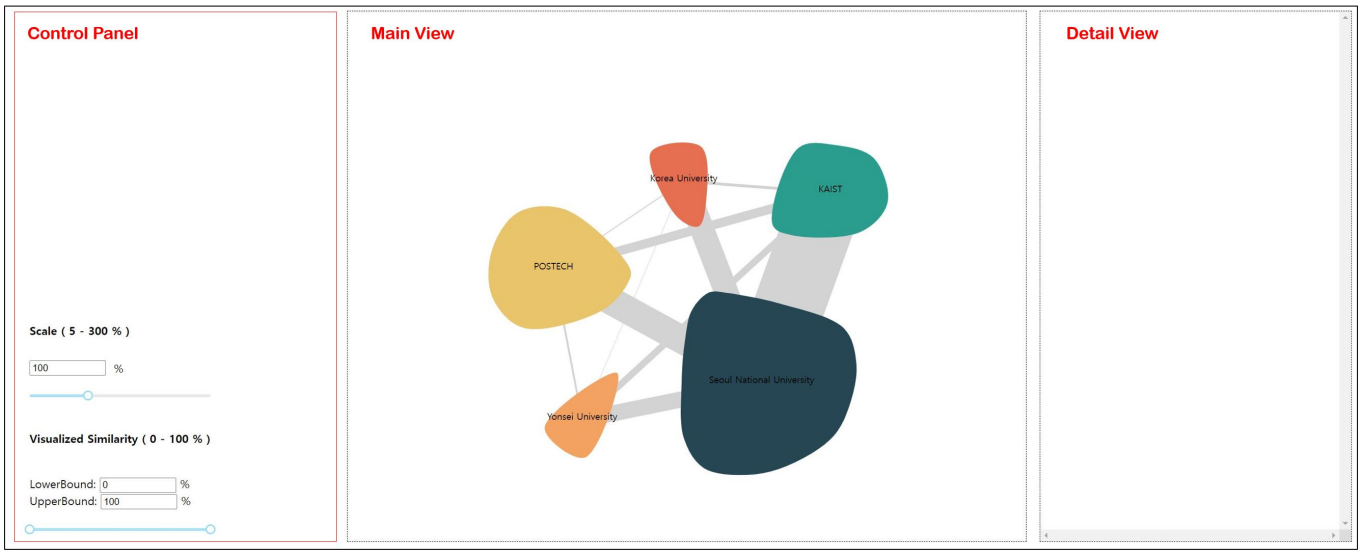
- FTL 동작의 구현
- Greedy, Cost-Benefit policy를 통한 GC 동작 구현
- 새로운 victim selection policy인 Low Computational Cost-Benefit (LC-CB) policy 고안 및 구현
- synthetic workload를 통한 반복 실험 구현
- MicroSoft Research Cambridge (MSRC) trace를 이용한 실험 구현
- 실험 결과 시각화 코드 구현

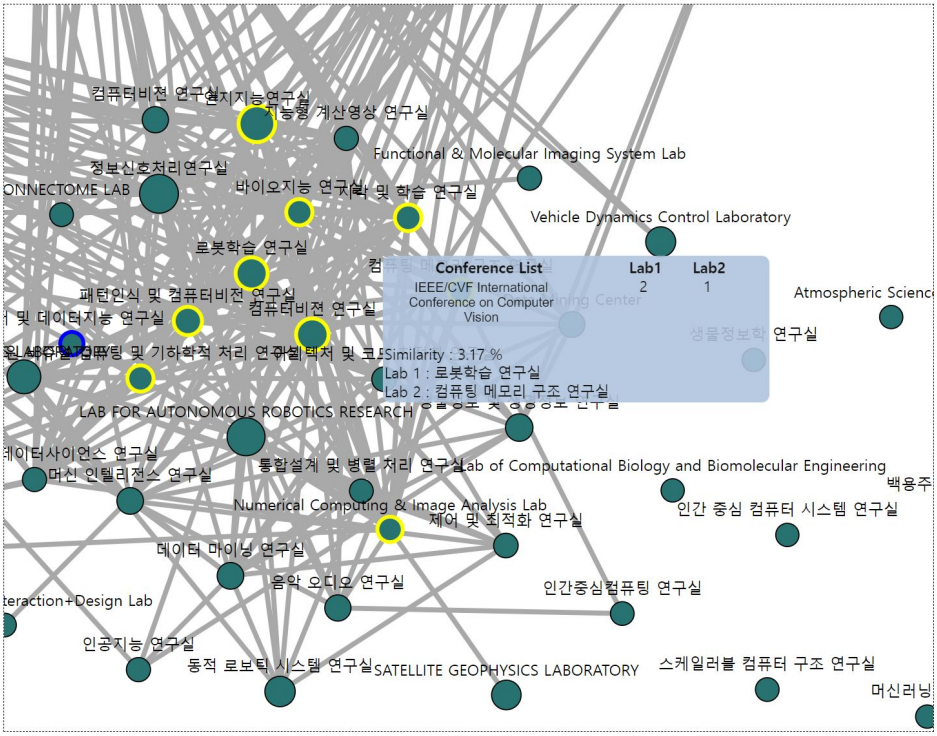
Evaluation with MSRC trace



36개의 MSRC trace에 대해 WA를 비교할 수 있습니다.

4. Research-Similarity Community Graph [Source Code]



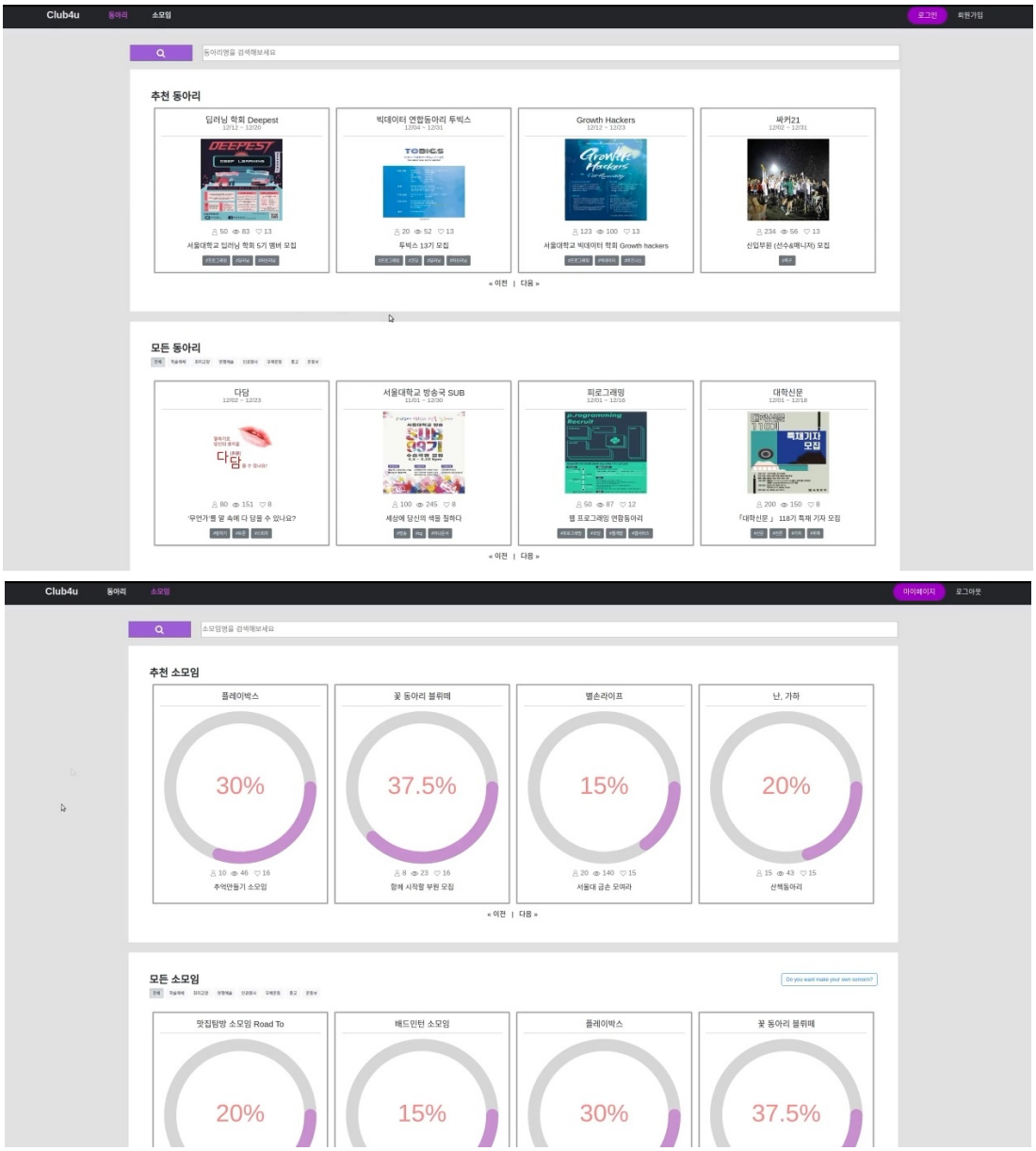


대학원 과목인 '정보시각화와 시각적분석' 과목에서 진행한 AI 연구실 간의 연관성을 표현하는 community graph를 시각화하는 프로젝트입니다. `selenium`과 `BeatifulSoup` 라이브러리를 통해 연구실 정보를 크롤링했으며, React와 D3를 사용하여 웹 페이지에 시각화했습니다. 프로젝트 총괄과 발표를 담당했으며, 크롤링과 시각화 코드에 대해서도 일정 부분 기여했습니다.

What I do

- 논문의 bibtex 크롤링 코드 작성
- 크롤링 데이터 포맷 정의 및 파싱
- 각 panel (control panel, main view, detail view)의 skeleton code 작성
- 그래프 확대, 이동, 연관도 필터링 기능 구현

5. Club4U [Source Code]



'소프트웨어 개발의 원리와 실습' 수업에서 진행한 프로젝트로, 동아리 및 소모임을 위한 커뮤니티 서비스입니다. React로 프론트엔드를 구현하였으며, Django로 백엔드를 구현했습니다. 프로젝트의 총괄과 발표를 맡았으며 코드 작성 역시 조원 중 가장 많은 부분을 담당했습니다.

What I do

- 프론트엔드 및 백엔드 skeleton code 작성
- Unit test 코드 작성
- AWS machine을 이용한 server deploy
- 회원가입, 로그인, 로그아웃 기능 구현
- 동아리/소모임 지원 페이지 구현
- 동이라/소모임 관리 페이지 구현
- 동아리 태그 기능 구현