

6월 17일 중간 점검 발표

▼ 목차

1. 진행상황 보고
2. 중간 점검
 - 2-1. 특징점 추출 방식의 선택적 가동 기능
 - 2-2. 특징점 추출 부분에 Unit Test 추가하기
 - 2-3. CI/CD: Build 결과 보이기
3. 향후 계획
 - 3-1. 고객사 요구사항 관련
 - 3-2. 개발사 관련

1. 진행상황 보고



프로젝트 최종 목표

→ 다양한 실험이 가능한 프레임워크를 만들고, 그 성능을 평가하기

- 바탕이 될 오픈소스 SLAM 선정: ProSLAM
- 중간 및 최종 개발 목표 설정 완료
- 일부 Local에서 개별적으로 이뤄지던 개발 환경을 통일 (Docker Image 및 Container 이용)
- **Architecture**
 - Docker Image 제작 및 사용법 공유 ✓
 - GitHub Repository 설정 및 CI 구현(GitHub Actions) ✓
 - Unit Test 구현 중 👤
- **Image Process**: Feature & Descriptor Extraction 알고리즘 선택 가동 기능 ✓
- **Frontend**: 변경 가능한 알고리즘 탐색 중 👤
- **Backend**
 - Loop Closure On/Off 기능 ✓
 - Optimization 등 변경 가능한 알고리즘 탐색 중 👤

2. 중간 점검

2-1. 특징점 추출 방식의 선택적 가동 기능

프로그램 가동에 필요한 **설정 파일에서 특징점 추출 방식을 선택**할 수 있습니다. 해당 부분의 텍스트를 변경한다면 선택한 알고리즘으로 특징점 추출을 진행합니다.

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/1ac00d6d-d854-4042-b39e-fa0a97172e8e/feature_detector_ORB.mp4

ORB

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/22dab02c-401b-4c19-8218-f88e8140c5de/feature_detector_AKAZE.mp4

AKAZE

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/e3127066-7c51-4af9-a040-23219eb6a2c7/feature_detector_FAST.mp4

FAST

2-2. 특징점 추출 부분에 Unit Test 추가하기

전체 프로세스에 대한 오류가 없음을 테스트하는 Unit Test는 존재하지만, 특징점 추출에 대한 Test는 아직 구현 중에 있습니다. 코드 구조를 다시 조정한 뒤 추가할 예정입니다.

```
[=====] Running 1 test from 1 test case.
[-----] Global test environment set-up.
[-----] 1 test from Test_Name_Major
[ RUN     ] Test_Name_Major.Test_Name_Minor
[      OK ] Test_Name_Major.Test_Name_Minor (0 ms)
[-----] 1 test from Test_Name_Major (0 ms total)

[-----] Global test environment tear-down
[=====] 1 test from 1 test case ran. (0 ms total)
[ PASSED ] 1 test.
```

2-3. CI/CD: Build 결과 보이기

ProSLAM은 Catkin Build 방식을 택하고 있으며, 이를 위한 dockerfile을 새로 만들어 진행하였습니다. 레포지토리에 PUSH를 할 때마다 빌드가 진행되도록 하였습니다.

```

CI
succeeded 4 hours ago in 1m 18s

> Set up job 2s
> Checkout source code 1s
▼ Build - Build SLAM with pre-built 3rdParty libs 1m 9s

1 ▶ Run eval $GET_REPO
13 === Build start ===
14
15 Sending build context to Docker daemon 6.656kB
16
17 Step 1/6 : FROM proslam:base
18 ---> 0dbde0326c3d
19 Step 2/6 : ARG BRANCH=development
20 ---> Running in f03b0b4c7e88
21 Removing intermediate container f03b0b4c7e88
22 ---> ed22801d0533
23 Step 3/6 : ARG DEBIAN_FRONTEND=noninteractive
24 ---> Running in cc759310ed45
25 Removing intermediate container cc759310ed45
26 ---> ad3a0ed9cbcb
27 Step 4/6 : RUN apt-get update -y && apt-get upgrade -y

```

빌드 시작

```

1149 [build - 44.3] [6/7 complete] [1/4 jobs] [ queued] [srrg_proslam:make(96%) - 4...
1150 [build - 44.4] [6/7 complete] [1/4 jobs] [ queued] [srrg_proslam:make(96%) - 4...
1151 [build - 44.5] [6/7 complete] [1/4 jobs] [ queued] [srrg_proslam:make(96%) - 4...
1152 [build - 44.6] [6/7 complete] [1/4 jobs] [ queued] [srrg_proslam:make(100%) - 4...
1153 [build - 44.7] [6/7 complete] [1/4 jobs] [ queued] [srrg_proslam:make(100%) - 4...
1154 [build - 44.8] [6/7 complete] [1/4 jobs] [ queued] [srrg_proslam:make(100%) - 4...
1155 [build - 44.9] [6/7 complete] [1/4 jobs] [ queued] [srrg_proslam:make(100%) - 4...
1156 [build - 45.0] [6/7 complete] [1/4 jobs] [ queued] [srrg_proslam:make(100%) - 4...
1157 [build - 45.1] [6/7 complete] [1/4 jobs] [ queued] [srrg_proslam:make(100%) - 4...
1158 [build - 45.2] [6/7 complete] [1/4 jobs] [ queued] [srrg_proslam:make(100%) - 4...
1159 [build - 45.3] [6/7 complete] [1/4 jobs] [ queued] [srrg_proslam:make(100%) - 4...
1160 [build - 45.4] [6/7 complete] [1/4 jobs] [ queued] [srrg_proslam:make(100%) - 4...
1161 [build - 45.5] [6/7 complete] [1/4 jobs] [ queued] [srrg_proslam:make(100%) - 4...
1162 <<<srrg_proslam [ 43.8 seconds ]
1163 [build] Summary:All 7packages succeeded!

```

빌드 성공 시

```

1104 Failed <<srrg_proslam:make [ Exited with code 2 ]
1105 Failed<<srrg_proslam [ 34.1 seconds ]
1106 [build] Summary:6of 7packages succeeded.
1107 [build]
1108 [build]
1109 [build]
1110 [build] Failed:1packages failed.
1111 [build] Runtime:35.5 seconds total.
1112 Removing intermediate container 47db849c5ebe
1113 [sudo] 717lumos의 암호: The command '/bin/sh -c echo "" Start Debug build == " && cd /root/catkin_ws/src/srrg_proslam && git remote update && git fetch --all && git checkout $(BRANCH) && git pull && git branch && cd ~/catkin_ws/src && catkin build --make-args tests -- srrg_proslam' returned a non-zero code: 1
1114 Error: Process completed with exit code 1.

```

빌드 실패 시

추가적으로 CI 시 Unit Test 결과 역시 볼 수 있도록 해당 기능을 추가했습니다.

```

1285 [=====] Running 1 test from 1 test case.
1286 [-----] Global test environment set-up.
1287 [-----] 1 test from AssemblyTest
1288 [ RUN ] AssemblyTest.PlaybackTest

```

```

1364 [ OK ] AssemblyTest.PlaybackTest (16 ms)
1365 [-----] 1 test from AssemblyTest (16 ms total)
1366
1367 [-----] Global test environment tear-down
1368 [=====] 1 test from 1 test case ran. (16 ms total)
1369 [ PASSED ] 1 test.

```

3. 향후 계획

3-1. 고객사 요구사항 관련

★ 알고리즘을 변경해 테스트 할 수 있는 프레임워크 개발

- Feature 추출 방식과 같이 설정 파일에서 선택해 가동할 수 있도록 구현할 예정
- (完) Image Processing
 - 특징점 추출 방식 선택적 구동 (3개 필수)
: ORB, AKAZE, FAST 구현 완료
- Frontend
 - Stereo Camera를 이용함에 있어 좋은 Point 선정 방식
: Standard RANSAC, LMedS(Least Median of Squares) RANSAC, 기존 ProSLAM 방식 (Epipolar Line 이용)
 - 현재 것처럼 3개 구현을 계획하고 있으며, 목표치였던 5개를 충족시킬 수 있는 부분은 고민 중
- Backend
 - Optimization 방식을 전환해보기
: Gauss-Newton, Levenberg, DOG LEG(Powell's dog leg method) (3가지)
 - Linear Solver Type
: CHOLMOD(Cholesky factorization), CSPARSE, DENSE (3가지)
 - 현재 Solver Library를 g2o를 사용중이나 ceres-solver도 사용할 수 있는 방안을 구상 중
- Loop Closure
 - On/Off 기능
 - Aligner
: classic ICP, Fast-robust ICP(2022)
 - 현재 것처럼 구현 혹은 계획 중인 기능이 Backend 6개 이상이므로, Loop Closure 목표치를 낮추는 것을 제안

★ 타 오픈소스 SLAM과 성능 비교

- ORB-SLAM2 외 1개까지는 가능, 그 이상은 시간 관계 상 힘들 듯 합니다. (기존 목표: ORB-SLAM 외 2개)

+ 보너스 포인트 현황

- ☒ KITTI 데이터셋에서 돌 수 있게 프레임워크 개량(+50)

- ☐ 정확도 개선(+50)
- ☐ 속도 개선(+50)
- ☐ 실시간 시각화 가능(+100)
- ☐ **(Ongoing)** 아키텍처/알고리즘 재사용성 개선(+100)
- ☐ **(Ongoing)** 안정성 확보: CI/CD 유닛테스트(+100)
- ☒ 다른 팀에게도 도움이 될 수 있는 자료 정리 및 공유(+50)
- ☐ **(Ongoing)** 오픈소스를 참고해 직접 VSLAM 파이프라인을 설계 및 구현(최소 2 모듈 이상 변경) (+150)
- ☐ **(Ongoing)** 고객의 갑작스러운 요구사항1 달성 (+100)

3-2. 개발사 관련

- GitHub 관리: main 브랜치로의 Push가 아니라 기능 별로 새 브랜치를 만들고 이에 Push한 뒤 Merge를 하는 방식으로 전환할 예정