프로젝트 개발
포트폴리오

CONTENT

자기소개

이재현
개발자

안녕하세요. 코딩을 좋아하며 파이썬을 즐겨 사용하는
5년차 **백앤드 개발자 이재현** 입니다.

다양한 프로젝트 경험

R&D 개발 프로젝트 경험을 많이 쌓았습니다. 연구과제 프로젝트 특성상 신기술을 사용하여 보다 혁신적이고 새로운 서비스를 개발하려고 노력해왔습니다.

협업과 커뮤니케이션

다양한 직군의 협업을 통해 문제를 함께 해결해 나가는 경험이 많습니다 . 원활한 커뮤니케이션과 상호 이해를 바탕으로 프로젝트의 시너지를 높이는데 기여해 왔습니다 .

새로운 기술 연구 및 도입

새로운 기술을 많이 탐색하고 연구하며 프로젝트에 도입할 수 있는지에 대한 고민을 많이 해왔습니다. 기술에 대한 인사이트를 얻고 공유하는 것 을 즐깁니다.

경력

이재현

개발자

CAREER

2020~2025 ㈜로딕스 전임연구원
2015~2021 중부대학교 정보보호학과 전공

EXPERIENCE

2024 (과기부)인공지능 기반 라이다 및 열영상 융합을 통한 야간 상황인지 기술
2023 (해양경찰)선박 모니터링 시스템
2023 (행안부)스마트현장 재난 가이던스 시스템
2022 (울주군)디지털 상황판
2022 (중기청)스마트팜
2021 (산학협력)달빛영상 기반 변화탐지
2021 (KIOST)국가해양영토 광역 감시망 구축

SKILL

- Python – Flask, FastAPI, Django
- ORM, SQLAlchemy, Postgresql, MySQL
- spatial Query, GeoServer, GDAL
- Celery, rabbitmq, Redis
- selenium, scheduler
- Torch, huggingFace
- Swagger / ReDoc
- Postman, Pytest
- Docker, DockeSwarm, K8s, GCP, AWS

LINK

Email: swaa23@gmail.com
Github: https://github.com/JaeHyunL
Blog: https://edudeveloper.tistory.com/
Kaggle: https://www.kaggle.com/jaehyunl

보유기술

프로그래밍 언어

- Java
- Python
- Java/Type SCRIPT

Backend 기술

- FastAPI, Flask, DRF
- RabbitMQ, Redis
- Celery
- Geoserver, GDAL, Spatial Query
- ORM, SQLAlchemy, Postgresql
- Selenium, Scheduler
- Swagger/ Redoc
- Postman, Pytest

ML/DL Model

- Torch, Tensorflow, Sickit-learn
- Hugging face
- GAN (pix2pix)
- GEN (text2Img, text2Video)
- NLP/ LLM (Bert, gpt , Llama 등)
- RVC
- High Resolution
- YOLO

Frontend 기술

- REACT
- Angular
- Pyqt5/6
- Android
- JavaFX, JavaSwing

DevOps

- Docker
- Docker Swarm, K8S
- GCP, AWS
- Gitlab-CI, ArgoCD
- Grafana, Prometheus
- ElasticSearch, Kibna, Logstash
- Gitlab 운용
- NAS 관리

협업

- Slack
- Discord
- Notion
- Gitlab
- JIRA

참여 프로젝트

프로젝트명	개발기간	개발내용	사용기술	구분
인공지능기반라이다및열영상융합을통한 야간상황인지기술	24.03~25.02	▪ 카메라To 모델파이프라인 처리 ▪ GAN 모델 트레이닝 및 Fine-Tuning	▪ OpenCV ▪ Pytorch ▪ Docker ▪ Calibration	▪ ML 파이프라인 구축
선박모니터링시스템	23.02~24.02	▪ AIS (대용량) 데이터 핸들링 (일일 2천 만건) ▪ 외부기관 알고리즘 수행 - 모델서빙 ▪ 데이터 스캐줄링 ▪ API 작업	▪ Flask ▪ Postgresql ▪ Docker-compose + Web Socket	▪ RestAPI 구현 ▪ ML 파이프라인 구축
스마트현장재난가이던스시스템	23.02~23.11	▪ API 작업 ▪ 데이터 파이프라인 구축 ▪ 스케줄링 ▪ 폐쇄망 전세계 지도 구축	▪ FastAPI ▪ Postgresql ▪ Python Multiple Version ▪ OSM-Tile Cashing	▪ RestAPI 구현 ▪ OSM 캐싱 조절
디지털 상황판	22.09~23.01	▪ API 작업 ▪ HTML 파싱을 통한 데이터 수집	▪ Flask ▪ Selenium, Flask	▪ Selenium 자동화를 통한 데이터 수집 알고리즘 구현
스마트팜	21.11~22.11	▪ API 작업 ▪ 커뮤니티 맵 작업 ▪ 데이터 파이프라인 처리 ▪ 드론 이미지 기반 정사영상 제작	▪ Flask ▪ React / OpenLayer ▪ OpenDronMap	▪ React 프레임워크 기반 커뮤니티 맵 웹사이트 구현 ▪ 데이터 파이프라인을 통한 수집 자동화 ▪ 드론 이미지 처리 ▪ RestAPI 구현
달빛영상기반변화탐지	21.05~24.12	▪ 위성영상 수집 데이터 파이프라인 구축 ▪ API 구현 ▪ 변화탐지 알고리즘 연동 작업	▪ AP Scheduler ▪ Geoserver ▪ Flask ▪ GDAL	▪ 데이터 파이프라인 구축 (데이터 수집, 정제, 활용) ▪ RestAPI 구현
국가해양영토광역감시망구축	2020.11~21.04	▪ 데이터 파이프라인 구축, ▪ API 구현 ▪ 외부기관 알고리즘 연동 ▪ 쿼리튜닝	▪ Celery, Rabbit MQ ▪ Flask ▪ Postgresql, SQLAlchemy	▪ RestAPI 구현 ▪ 쿼리 튜닝을 통한 성능 개선 (30초 -> 0.3 초 페이징 처리)

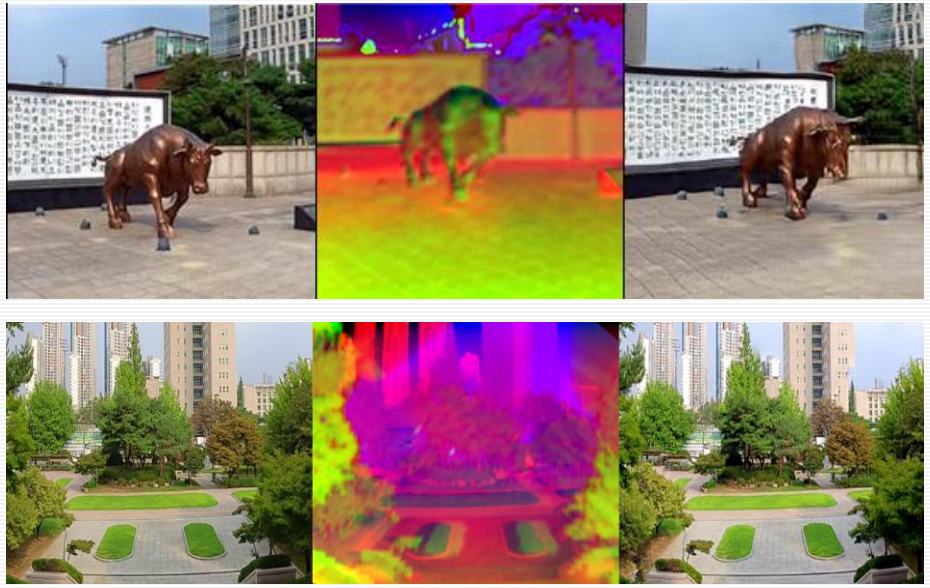
주요 포트폴리오

프로젝트 기본정보

프로젝트명	인공지능기반라이다및열영상융합을통한야간상황인지 기술
프로젝트기간	24.03~25.02
프로젝트인원	5명
설명	라이다+ 열 이미지를 이용한 주·야간 스타일 변화를 활용한 야간 상황인지 기술 개발
담당역할	데이터 파이프라인, 모델 서빙, 모델 개선

개발 주요사항

- 카메라 장비 구성 및 영상 취득 프로그램 개발
- 데이터 품질 향상을 위한 센서간 캘리브레이션
- 열화상, 라이다 이미지 정합을 통한 데이터셋 생성
- GAN (적대적-신경망) 인공지능 학습 및 활용



주간 야간 모의

주·야간 스타일 변화를 통하여 야간에도 실제 낮과 같은 이미지를 얻을 수 있다.

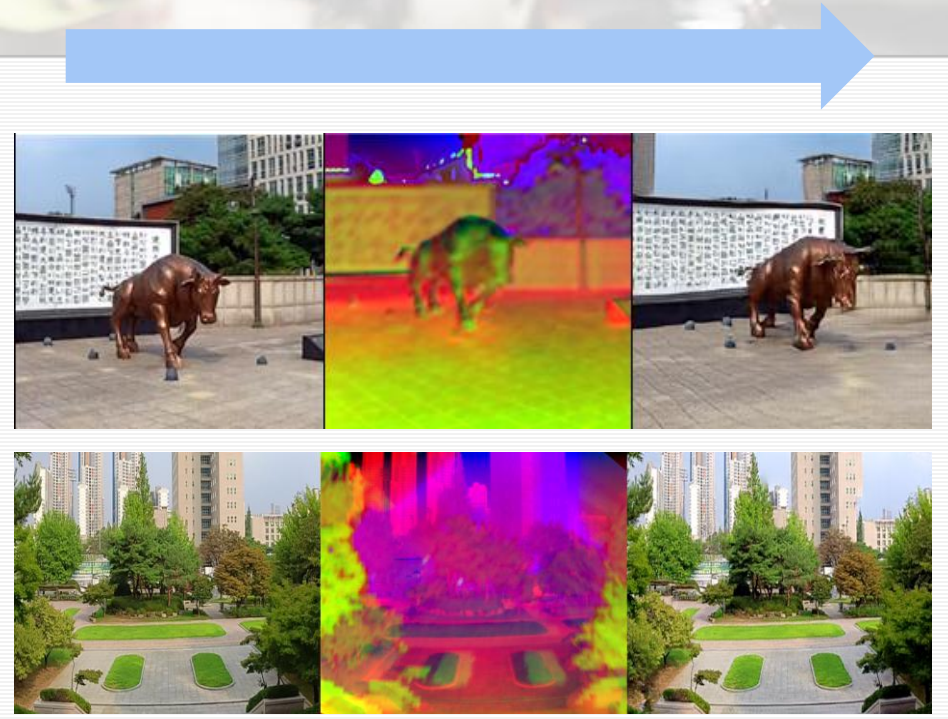
주요 포트폴리오

사용 기술

- Python
- OpenCV
- LightGlue (특징점 매칭)
- GAN Model (CycleGAN, UNSB, CUT)

설명

이미지to이미지 연구가 활발해지며 또한 이미지간 Style변환하는 연구모델이 많이 연구되어 있는 중 라이다와 열영상은 일반 카메라와 다르게 야간에도 낮과 유사하게 데이터를 얻을 수 있다는 관점에서 접근하였습니다.
이를 GAN 모델에게 데이터를 학습시켜 스타일 변화를 시킨 프로젝트입니다.

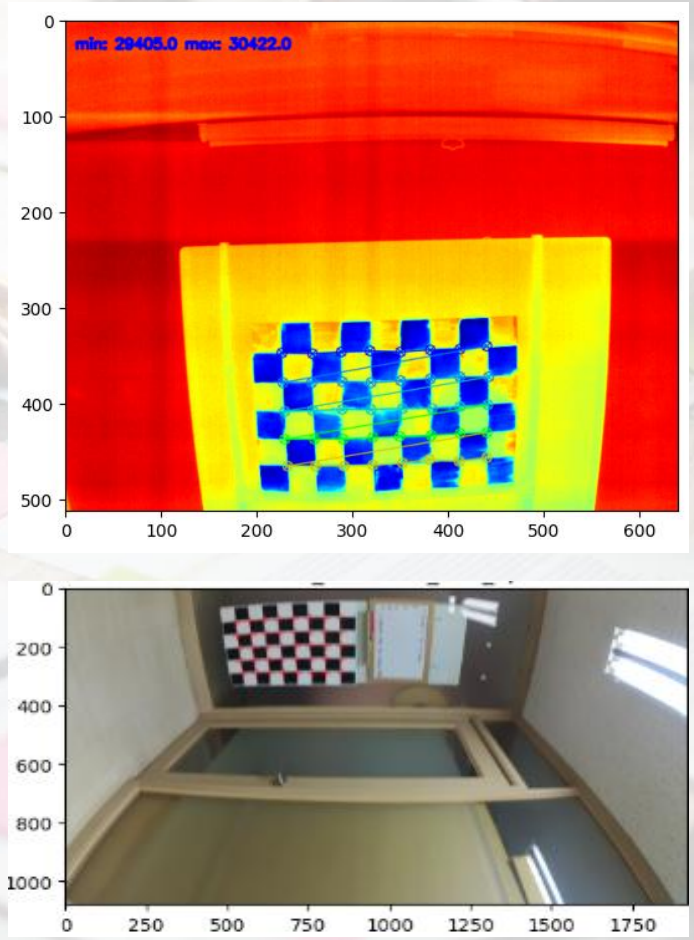


주·야간 스타일 변화를 통하여 야간에도 실제 낮과 같은 이미지를 얻을 수 있다.

주요 포트폴리오

데이터 전처리 (Calibration)

```python
class ChessBoardDetector:
    """체스보드 탐지"""

    def __init__(self, dir_path:str, ext_name:str, recursive:bool=False, pattern_size=[8,5]):
        """
        기본 생성자

        Args:
            dir_path: 이미지 파일들이 포함되어 있는 디렉터리 경로(예: /mnt/e/flat/data/DC)
            ext_name: 이미지 파일의 확장자명(예: .jpg)
            recursive: 디렉터리 경로에 대한 재귀적 탐색 여부
        """
        self.dir_path = dir_path
        self.ext_name = ext_name
        self.recursive = recursive
        self.__img_name_and_detection_info_dict = {}

        # 체커보드의 차원 정의
        self.pattern_size = pattern_size
        self.criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 30, 0.001)

        # 실세계 3D 좌표 정의
        self.objp = np.zeros((1, self.pattern_size[0] * self.pattern_size[1], 3), np.float32)
        #self.objp[0,:,:2] = np.mgrid[0:self.pattern_size[0], 0:self.pattern_size[1]].T.reshape(-1, 2)
        # self.objp[0,:,:2] = np.mgrid[0:self.pattern_size[0], 0:self.pattern_size[1]].T.reshape(-1, 2) * 100
        self.objp[0,:,:2] = np.mgrid[self.pattern_size[0]-1:-1:-1, self.pattern_size[1]-1:-1:-1].T.reshape(-1, 2) * 100

    def __len__(self):
        """
        체커보드 탐지 결과의 성공 개수를 반환
        """
        return len(self.__img_name_and_detection_info_dict)

    def __getitem__(self, index:int):
        """
        지정하는 인덱스의 이미지에 대한 체커보드 탐지 결과를 반환
        """
        return list(self.__img_name_and_detection_info_dict.values())[index]
```

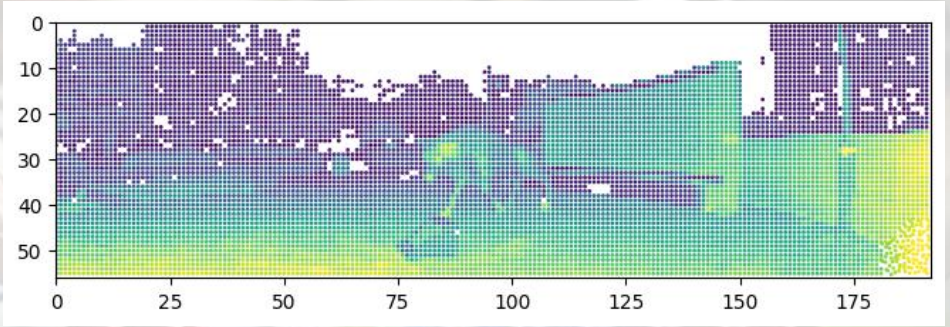


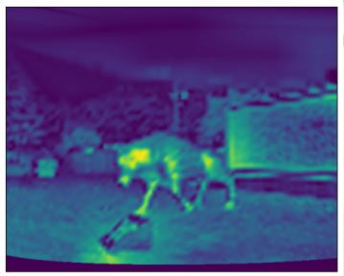
요 약 캘리브레이션을 통하여 채커보드에 점 위치를 추출하고 점 간 Rvec, Tvec을 구하여 이미지를 일치시킴

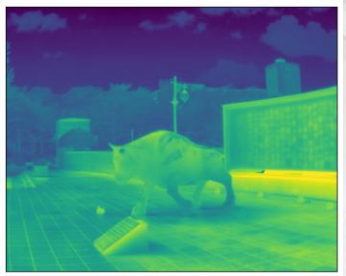
주요 포트폴리오

데이터 전처리 (정합)

```
def project(index, scene: ldx.CameraScene):
    id = index + 1
    print(f'id: {id} sensor: project lidar to optical')

    rgb = ndimage.rotate(cv2.cvtColor(cv2.imread(f'{data_prefix}_{id}_optical.png'), cv2.COLOR_BGR2RGB), optical_rotation90_ntimes*90)
    lidar_scene =ldx.LidarScene(f'{data_prefix}_{id}_lidar.csv')

    projected, inside_mask = scene.project(lidar_scene.point_data[:,:3])

    crop_top = 500
    crop_bottom = 720

    rgb = rgb[crop_top:-crop_bottom,:]
    projected[:,1] -= crop_top

    dpi = matplotlib.rcParams['figure.dpi']
    fig_w = rgb.shape[1] / dpi
    fig_h = rgb.shape[0] / dpi
    plt.figure(figsize=(fig_w,fig_h))
    plt.imshow(rgb)

    plt.scatter(
        projected[lidar_scene.inliers_mask & inside_mask,0],
        projected[lidar_scene.inliers_mask & inside_mask,1],
        s=10,
        marker='.',
        c=lidar_scene.point_data[lidar_scene.inliers_mask & inside_mask, 4],
        cmap='jet')
    plt.xlim([0,rgb.shape[1]])
    plt.ylim([rgb.shape[0], 0])
    plt.show()
```





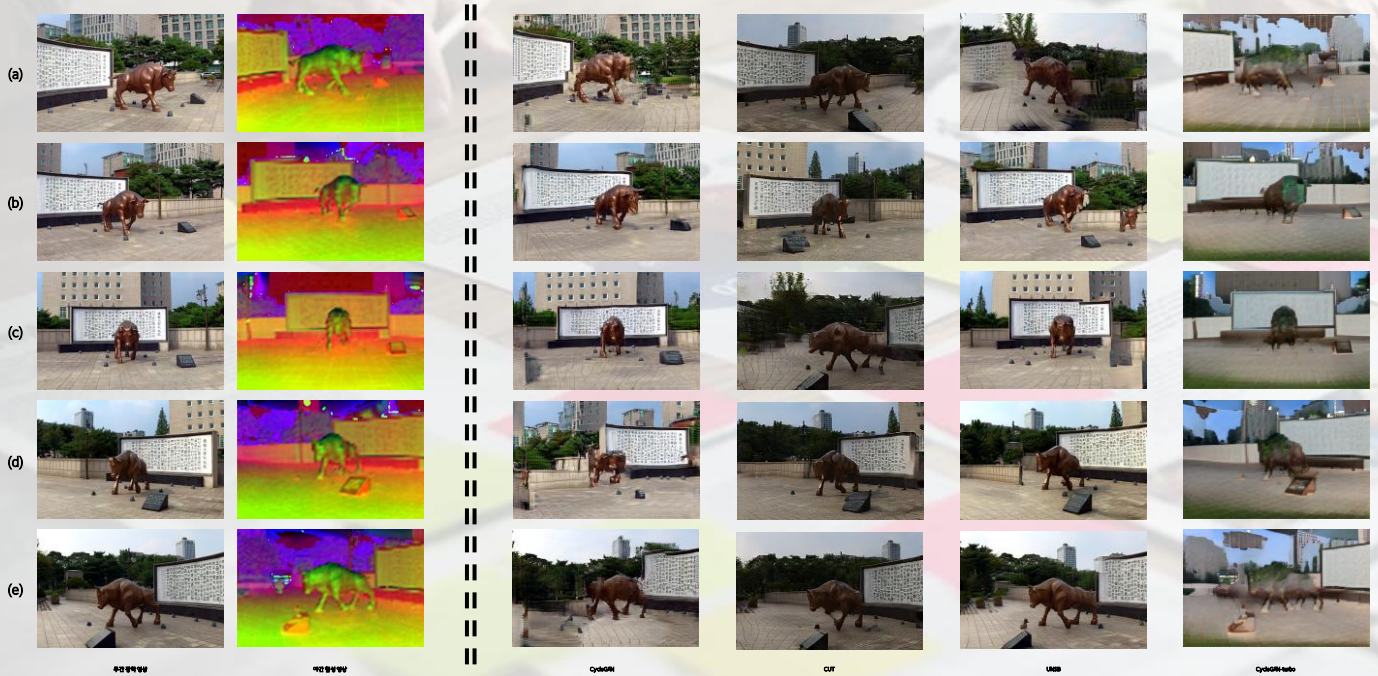


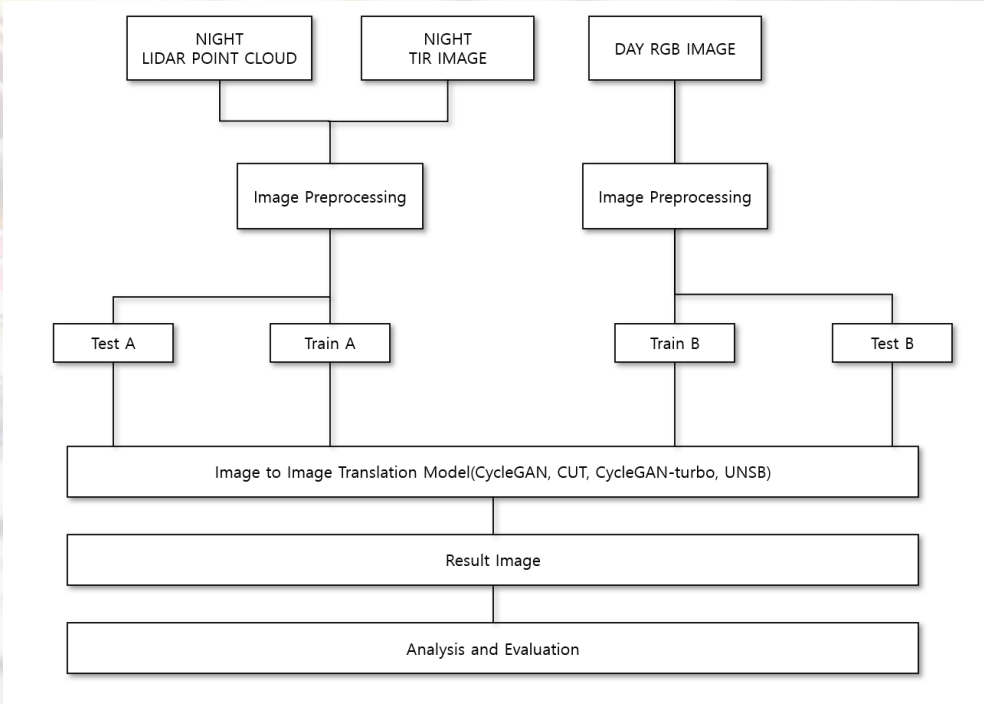
요 약

라이다 (3차원 점 데이터)를 보간을 통하여 이미지형태로 변환 후 열 영상과 Calibration을 통하여 이미지를 정합 시킴

주요 포트폴리오

모델 학습 (하이퍼 파라미터 튜닝)



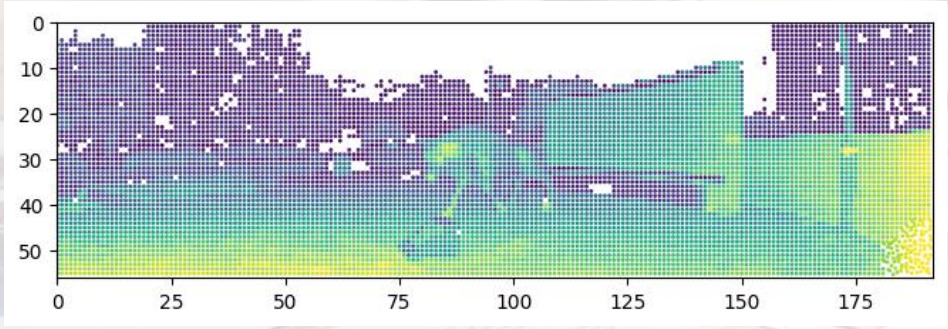


요 약

야간 라이다(Train A), 주간 RGB영상(Train B)를 비지도 학습 방식을 통하여 학습시키고
수 차례의 실험을 통하여 적합한 하이퍼 파라미터를 도출합니다.

주요 포트폴리오

모델 활용 (라이다 컬러화)



5	4652	2011	1467	1788	5278	187	168	141
6	4730	1983	1487	1925	5342	184	169	145
7	4800	1951	1505	2042	5397	185	167	144
8	4856	1911	1517	2154	5435	185	167	151
9	4866	1853	1516	2126	5424	168	153	141
10	4857	1789	1509	2037	5392	145	136	124
11	4837	1721	1499	1900	5350	132	127	119
12	4816	1654	1488	1759	5306	134	127	121
13	4769	1579	1470	1682	5236	135	126	118
14	4735	1510	1456	1629	5180	135	129	126
15	4683	1437	1437	1585	5105	133	129	123
16	4630	1365	1417	1528	5032	125	121	112
17	4621	1307	1411	1543	5006	112	109	102
18	4569	1238	1393	1539	4935	125	118	113
19	4536	1175	1380	1491	4885	125	118	113
20	4505	1114	1368	1512	4839	122	116	113
21	4466	1053	1354	1472	4785	120	117	116
22	4445	996	1345	1489	4751	119	116	113
23	4429	941	1338	1511	4723	118	115	108
24	4399	884	1327	1470	4679	113	110	104
25	4385	831	1321	1491	4655	114	110	105
26	4349	775	1308	1491	4608	116	110	107
27	4330	722	1301	1492	4580	119	112	107
28	4301	669	1290	1483	4540	120	113	108

2025-01-10_10-04-40_13_fake_xyz

준비 접근성: 사용할 수 없음

요 약

학습모델 결과를 통해 얻은 이미지(Fake RGB)와 원본 라이다간 기하는 동일하여(Pix2Pix)
라이다 원본에 Fake색상을 붙여서 3D PointCloud로 생성

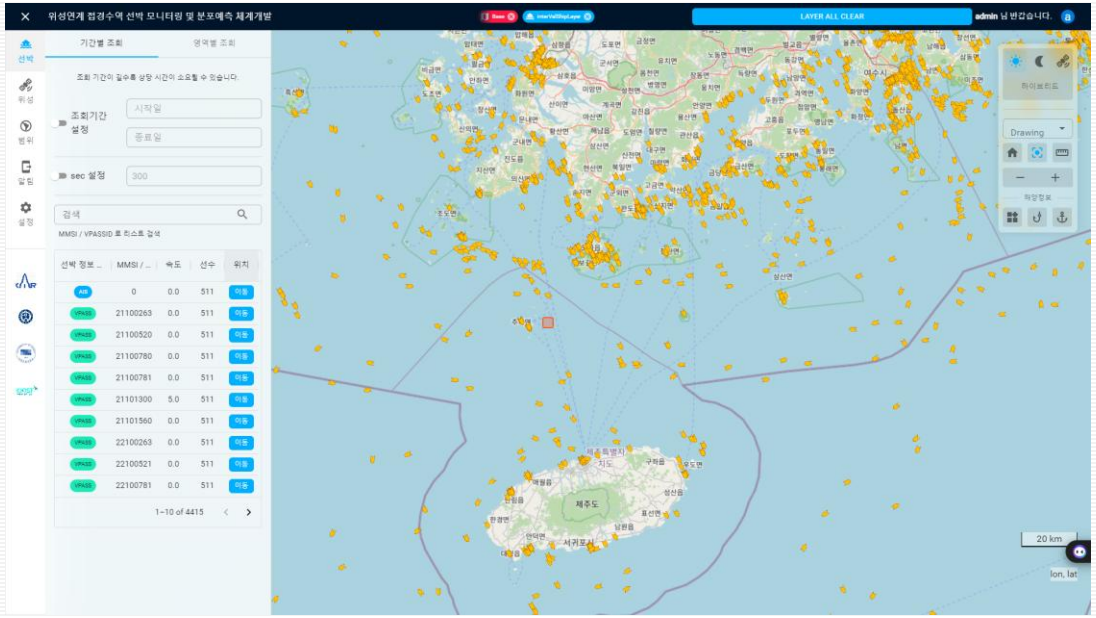
주요 포트폴리오

프로젝트 기본정보

프로젝트명	선박 모니터링 시스템
프로젝트 기간	23.02~24.02
프로젝트 인원	15명
설명	각 연구기관에서 개발한 알고리즘을 통합하여 불법선박탐지 웹사이트 구축 사업
담당역할	데이터 파이프라인, RestAPI, 쿼리튜닝

개발 주요사항

- AIS 데이터 처리
- 알고리즘 통합 운용환경 구축
- 데이터 파이프라인 구축



위성연계 접경수역 선박 모니터링 대시보드

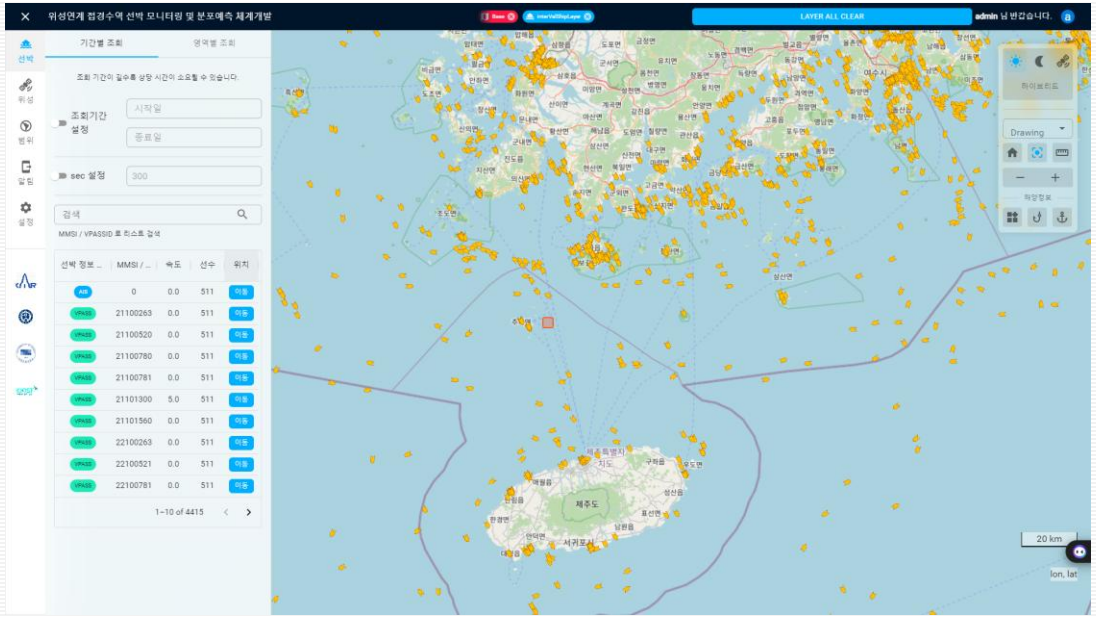
주요 포트폴리오

사용 기술

- Python
- Flask
- Socket
- Redis/Celery

설명

자동선박식별시스템 AIS 데이터를 수집 정제하고, AIS데이터와 각 종 선박탐지 알고리즘을 통하여 불법 선박탐지 및 미 등록 선박을 탐지하고 그 결과를 (준)실시간으로 WebSite에 표출한다.



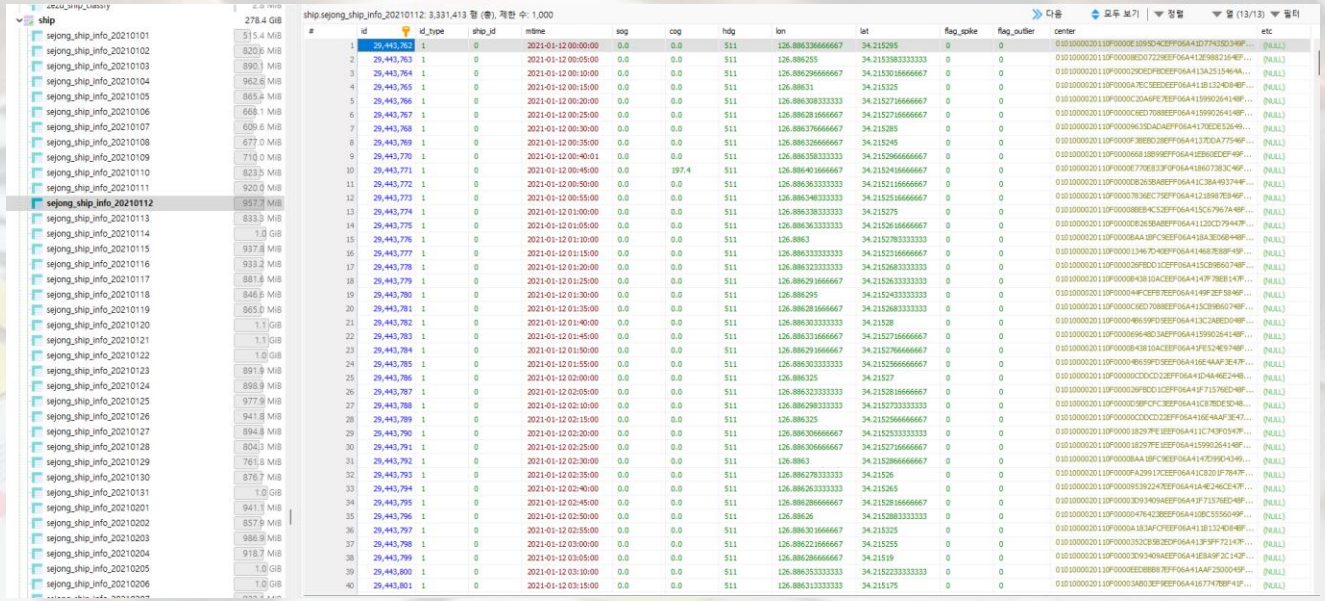
위성연계 접경수역 선박 모니터링 대시보드

주요 포트폴리오

AIS (대용량) 데이터 프로세싱

1. FTP 폴더, RestAPI 를 통하여 AIS 전송
2. Celery (분산처리)를 통하여 데이터 Access 및 Reading
3. 테이블 파티셔닝을 통한 일별 데이터 처리
4. 속도측정 및 분산 처리 테스트

1. 일 300만~2000만 건 대용량 데이터
2. 데이터 Row가 증가함에 따라 Select 시간 증가 – 테이블 파티셔닝을 통한 조회속도 해결 (45~3분 –〉 3초 이내)
3. 테이블 파니셔닝 시 where문 조회 조건이 복잡해짐 – 파티션 프루닝을 통한 성능 향상



주요 포트폴리오

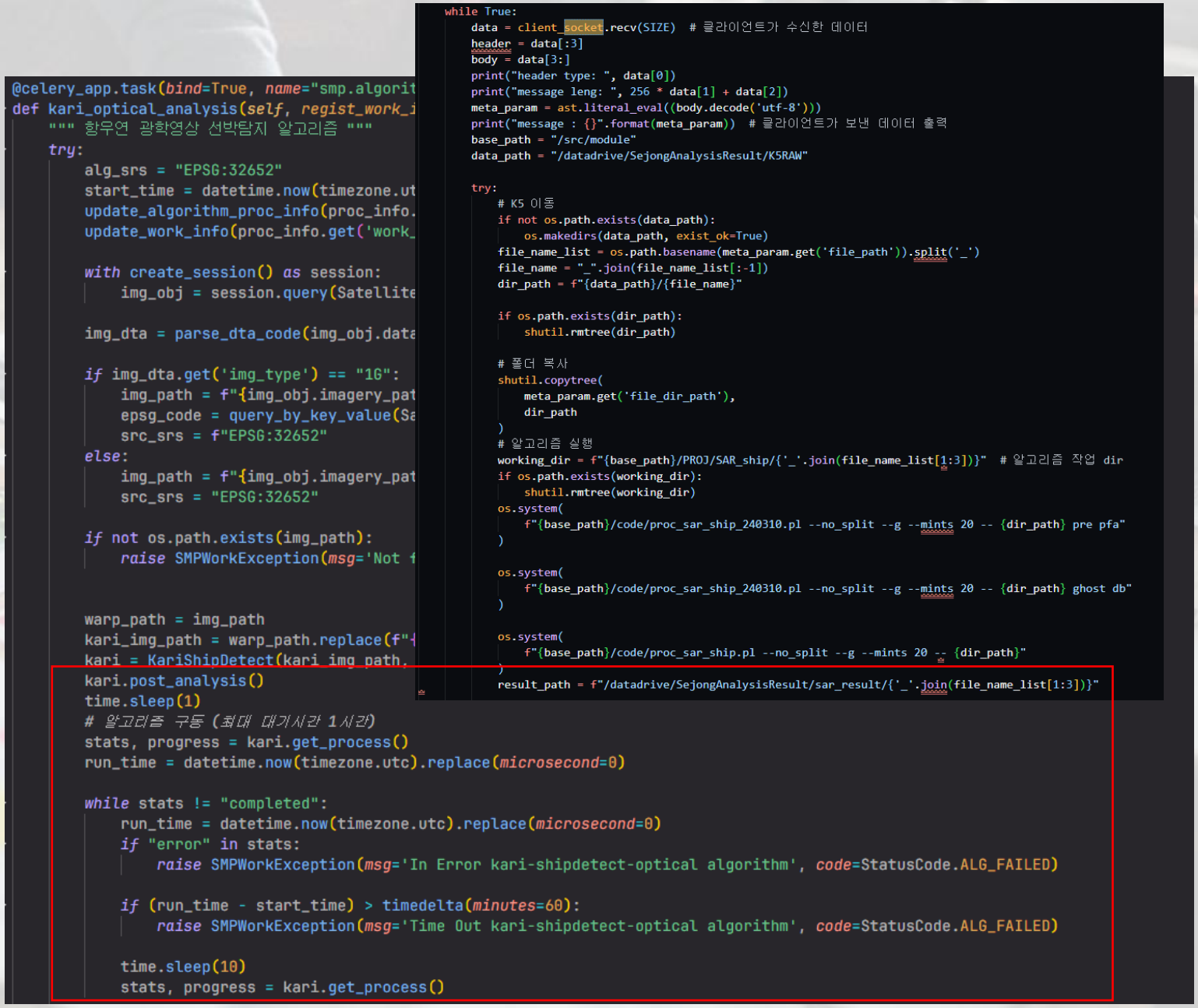
알고리즘 통합 환경 구축

기관1. Tensorflow GPU 사용
기관2. Python3.8 사용, OpenCV 사용
기관3. Matlab runtime, Perl , Python 3.10 사용
기관4. Tensorflow 2.5사용

각 기관별 알고리즘 버전 및 충돌 방지를 위하여 도커 컨테이너 기반 격리환경 사용합니다.

알고리즘 실행 및 결과를 Socket으로 전달 받습니다.
(Socket 통신으로 전달된 알고리즘은 프로세스 간 독립성이 실행되므로)

Celery Task와 WebSocket 예외 처리를 통한 이상 방지



```python
@celery_app.task(bind=True, name="smp.algorit
def kari_optical_analysis(self, regist_work_i
    """ 항우연 광학영상 선박탐지 알고리즘 """
    try:
        alg_srs = "EPSG:32652"
        start_time = datetime.now(timezone.ut
        update_algorithm_proc_info(proc_info.
        update_work_info(proc_info.get('work_

        with create_session() as session:
            img_obj = session.query(Satellite

        img_dta = parse_dta_code(img_obj.data

        if img_dta.get('img_type') == "1G":
            img_path = f"{img_obj.imagery_pat
            epsg_code = query_by_key_value(Sa
            src_srs = f"EPSG:32652"
        else:
            img_path = f"{img_obj.imagery_pat
            src_srs = "EPSG:32652"

        if not os.path.exists(img_path):
            raise SMPWorkException(msg='Not f


        warp_path = img_path
        kari_img_path = warp_path.replace(f"
        kari = KariShipDetect(kari_img_path,
        kari.post_analysis()
        time.sleep(1)
        # 알고리즘 구동 (최대 대기시간 1시간)
        stats, progress = kari.get_process()
        run_time = datetime.now(timezone.utc).replace(microsecond=0)

        while stats != "completed":
            run_time = datetime.now(timezone.utc).replace(microsecond=0)
            if "error" in stats:
                raise SMPWorkException(msg='In Error kari-shipdetect-optical algorithm', code=StatusCode.ALG_FAILED)

            if (run_time - start_time) > timedelta(minutes=60):
                raise SMPWorkException(msg='Time Out kari-shipdetect-optical algorithm', code=StatusCode.ALG_FAILED)

            time.sleep(10)
            stats, progress = kari.get_process()
```

```python
while True:
    data = client_socket.recv(SIZE)  # 클라이언트가 수신한 데이터
    header = data[:3]
    body = data[3:]
    print("header type: ", data[0])
    print("message leng: ", 256 * data[1] + data[2])
    meta_param = ast.literal_eval((body.decode('utf-8')))
    print("message : {}".format(meta_param))  # 클라이언트가 보낸 데이터 출력
    base_path = "/src/module"
    data_path = "/datadrive/SejongAnalysisResult/K5RAW"

    try:
        # K5 이동
        if not os.path.exists(data_path):
            os.makedirs(data_path, exist_ok=True)
        file_name_list = os.path.basename(meta_param.get('file_path')).split('_')
        file_name = "_".join(file_name_list[:-1])
        dir_path = f"{data_path}/{file_name}"

        if os.path.exists(dir_path):
            shutil.rmtree(dir_path)

        # 폴더 복사
        shutil.copytree(
            meta_param.get('file_dir_path'),
            dir_path
        )
        # 알고리즘 실행
        working_dir = f"{base_path}/PROJ/SAR_ship/{'_'.join(file_name_list[1:3])}"  # 알고리즘 작업 dir
        if os.path.exists(working_dir):
            shutil.rmtree(working_dir)
        os.system(
            f"{base_path}/code/proc_sar_ship_240310.pl --no_split --g --mints 20 -- {dir_path} pre pfa"
        )

        os.system(
            f"{base_path}/code/proc_sar_ship_240310.pl --no_split --g --mints 20 -- {dir_path} ghost db"
        )

        os.system(
            f"{base_path}/code/proc_sar_ship.pl --no_split --g --mints 20 -- {dir_path}"
        )
        result_path = f"/datadrive/SejongAnalysisResult/sar_result/{'_'.join(file_name_list[1:3])}"
```