

보안 과제(), 일반 과제(○) / 공개(○), 비공개()발간등록번호()

국토교통기술사업화지원사업 제3차 연도 최종결과 보고서

인공지능기반의 도로 파손 정보 제공 기술 개발 최종보고서

2021. 01. 20.

주관연구기관 / (주) 케이웍스

국토교통부
국토교통과학기술진흥원

제출문

제 출 문

국토교통기술진흥원장 귀하

‘인공지능기반의 도로 파손 정보 제공 기술 개발’(연구개발 기간 : 2018.04.01. ~ 2020.12.31.)
과제의 최종보고서 10부를 제출합니다.

2020. 01. 20.

주관연구기관명 : 케이웍스

(대표자) 최 종 선



공동연구기관명 : 한국건설기술연구원

(대표자) 한 승 현



주관연구기관책임자 : 박 준 규

공동연구기관책임자 : 류 승 기

국토교통부소관 연구개발사업 운영규정 제29조에 따라 최종보고서 열람에
동의합니다.

보고서 요약서

과제 고유 번호	20TBIP-C144 255-03	해당 단계 연구 기간	2020.01.01.~ 2020.12.31.	단계구분	(최종단계)
연구사업명	중사업명	인공지능 기반의 도로 파손 정보 제공 기술 개발			
	세부사업명	인공지능 기반의 도로 파손 정보 제공 기술 개발			
연구과제명	대과제명	국토교통기술사업화지원사업			
	세부과제명	인공지능 기반의 도로 파손 정보 제공 기술 개발			
연구책임자	박준규	해당단계 참여연구원 수	총: 27명 내부: 27명 외부: 명	해당단계 연구비	정부: 2,4600,000천원 민간: 1,6400,000천원 정부외: 천원 상대국: 천원 계: 4,100,334천원
		총 연구기간 참여연구원 수	총: 27명 내부: 27명 외부: 명	총 연구비	정부: 2,4600,000천원 민간: 1,6400,000천원 정부외: 천원 상대국: 천원 계: 4,100,334천원
연구기관명 및 소속 부서명	(주)케이웍스 / 마케팅기획팀			참여기업명	
국제공동연구	상대국명:			상대국 연구기관명:	
위탁연구	연구기관명:			연구책임자:	

※ 국내·외의 기술개발 현황은 연구개발계획서에 기재한 내용으로 같음

연구개발성과의 보안등급 및 사유	일반
-------------------------	----

9대 성과 등록·기탁번호

구분	논문	특허	보고서 원문	연구시설 · 장비	기술요약 정보	소프트 웨어	화합물	생명자원		신품종	
								생명 정보	생물 자원	정보	실물
등록·기 탁 번호											

국가과학기술종합정보시스템에 등록된 연구시설·장비 현황

구입기관	연구시설 · 장비명	규격 (모델명)	수량	구입연월일	구입가격 (천원)	구입처 (전화)	비고 (설치장소)	NTIS 등록번호

요약(연구개발성과를 중심으로 개조식으로 작성하되, 500자 이내로 작성합니다)

보고서 면수

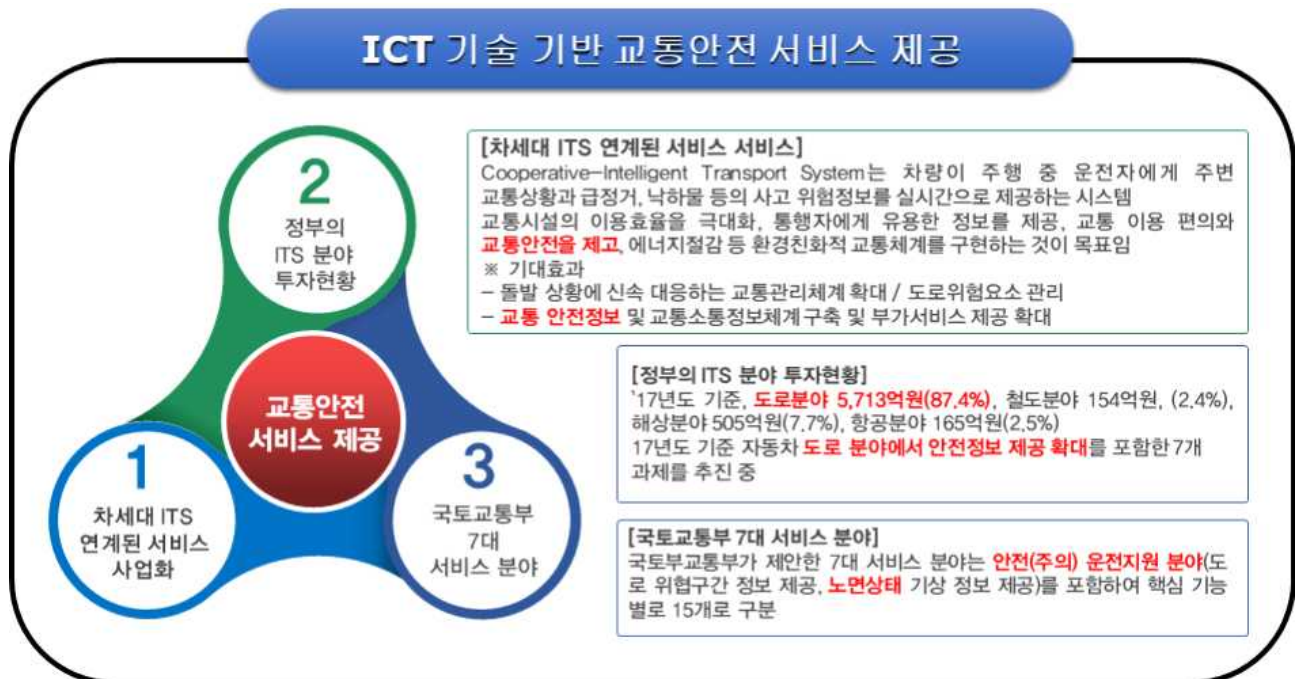
요약문

연구의 목적 및 내용	○ 교통사고 예방을 위한 도로 파손 정보 제공 기술 사업화				
연구개발성과	○ 민간 및 공공 기관을 대상으로 한 도로 노면 파손 제공 기술 사업화 - 실시간 도로 파손 정보 수집에 따른 신속한 도로 보수 공사 유도 - 정확한 도로 파손 정보 확보에 따른 현실적 보수 계획 수립 지원 ○ 차량 네비게이션을 활용한 도로 노면 파손 정보 제공 기술 사업화 - 일반 사용자에게 도로 파손 상태 정보 제공 기술 구축 - 주행 차로별 도로 노면 위험 지역 사전 알림 시스템 구축 ○ 도로 노면 파손에 의한 교통사고 예방 전략 수립 지원 사업화 - 도로 노면 파손에 의한 교통사고 사전 예방을 위한 데이터 분석 시스템 구축 - 도로 노면 위험물 정보 수집에 따른 실시간 도로 안전성 확보 지원 ○ 자율 주행차의 안전성 향상을 위한 도로 파손 정보 제공 기술 사업화 - 도로 노면 상태를 고려한 주행 알고리즘 개발을 위한 데이터 제공 기술 개발 - 도로 파손 상태 정보 제공을 통한 자율 주행차의 안전성 확보 지원				
연구개발성과의 활용계획 (기대효과)	○ 교통사고 예방을 위한 실시간 도로 노면 파손 제공 기술 사업화 ○ 고해상도 영상 기반의 차량 탑재형 도로 노면 파손 수집단말기 ○ 인공지능 기반의 도로 노면 파손 정보 탐지 및 분류 시스템 ○ 다수의 수집단말기를 활용한 시공간적으로 단절 없는 도로 노면상태 정보 수집 ○ 연간 교통사고 감소비용: 10,011백만원 (현재가치 환산 편익) ○ 차량 네비게이션 서비스(T-map 등)에 노면상태 정보가 추가될 통한 도로교통 안전 분야 신시장 및 양질의 일자리 창출 가능 ○ 도로 노면 파손 정보제공으로 인해 교통사고 감소 가능				
국문핵심어 (5개 이내)	교통사고	안전	수집단말기	도로노면파손	인공지능
영문핵심어 (5개 이내)	traffic accident	safety	Data collection terminals	road damage	AI

< 목 차 >

1. 연구개발과제의 개요	4
2. 연구수행내용 및 성과	9
3. 목표 달성도 및 관련 분야 기여도	81
4. 연구개발성과의 활용 계획 등	90

1. 연구개발과제의 개요



[차세대 ITS(C-ITS)와 연계된 서비스 사업화]

○ 차세대ITS 란?

- Cooperative-Intelligent Transport System는 차량이 주행 중 운전자에게 주변 교통 상황과 급정거, 낙하물 등의 사고 위험정보를 실시간으로 제공하는 시스템.
- 도로·차량·화물 등 교통의 구성요소에 통신기술을 적용하여 교통정보를 수집·관리·제공



<차세대 ITS 기본계획의 배경>

차세대(C-ITS) 홍보관 제공

- 교통시설의 이용효율을 극대화, 통행자에게 유용한 정보를 제공, 교통 이용 편의와 교통안전을 제고, 에너지절감 등 환경친화적 교통체계를 구현하는 것이 목표임

○ 차세대 ITS의 추진배경

- 교통사고의 획기적인 감축을 위해서 전 세계적으로 추진중인 차세대(C-ITS) 도입
- 미국, 유럽, 일본은 시범사업을 거쳐 주파수, 표준화, 법제화 등 차세대 ITS 시범사업 기반을 마련하는 등 실용화 직전단계로 적극 도입 추진

○ 現ITS와 차세대 ITS(C-ITS)의 차이점

구분	現 ITS	C-ITS
정보수집 및 제공	단방향 수집/제공 · 검지기, DSRC, CCTV, 연계 Data수집단말기 사용 · VMS, WEB, 모바일 제공	양방향 수집/제공 · 차량간 양방향 통신 체계 추가 · 차량OBU를 활용한 정보 수집 · RSU 장비를 활용한 정보 수집
Key-Point	교통관리 중심 · 수집-제공체계의 분리: 공간제약 · 센터중심의 정보제공: 지연발생 · 돌발상황 신속대응 한계	교통안전중심 · V2X간의 지속적인 데이터 공유 · 개별차량간을 실시간 정보공유 · 돌발상황 사전대응 및 예방

○ 기대효과

- 돌발 상황에 신속 대응하는 교통관리체계 확대
- 도로위험요소 관리 및 협력형 ITS 도입 확대
- 교통 안전정보 및 교통소통정보체계 구축 및 부가서비스 제공 확대
- 교통요금 전자지불 방법 및 시설 개선
- 여행자 맞춤 대중교통정보 통합 및 제공 확대
- 저탄소 녹색교통 지원 강화

구분	미구축	ITS	차세대ITS(C-ITS)(추정)	비고
혼잡비용	28.5조('10년)	연 4,300억원 절감	연 8,000억원 절감	
통행속도	20km/h 내외	15% 증가	30% 증가	도심
사고예방			46% 감소	총 사고건수

<정량적 기대 효과>

국토교통부 차세대 ITS 기본계획 제공

[정부의 ITS 분야 투자현황 (시장)]

- `16년도 6,325억원에서 `17년도 6,537억원으로 3.4% 증가하였으며, `18년도 투자 계획은 6,402억원으로 `17년 대비 1.2% 감소 예상
- `17년도 기준, 도로분야 5,713억원(87.4%), 철도분야 154억원, (2.4%), 해상분야 505억원(7.7%), 항공분야 165억원(2.5%)
- 지능형교통체계 시행계획에 의하면 지속 가능한 지능형교통체계의 성장과 성숙으로 생활형 스마트 도로교통 구현을 목표에 전체 예산 중 가장 많은 비중을 확보

분야별	단위 사업	`17년도 예산				`18년도 예산				비고
		계	국비	지방비	기타	계	국비	지방비	기타	
총계	188건	653,692	294,722	138,001	221,067	604,183	341,954	128,504	169,725	
도로부분	161건	571,287	225,106	133,925	212,256	544,692	253,482	127,747	163,463	

<분야별 투자현황(단위 : 백만원)>

출처: 국토교통부 2017년도 지능형교통체계(ITS) 시행계획

- `17년도 기준 자동차 도로 분야에서 7개 과제를 추진 중(교통관리체계 확대, 협력형 ITS 확대, 안전정보제공 확대, 교통소통정보체계 구축, 교통요금 전자기불 방법 및 시설 개선, 대중교통정보 제공, 저탄소 녹색교통 지원)

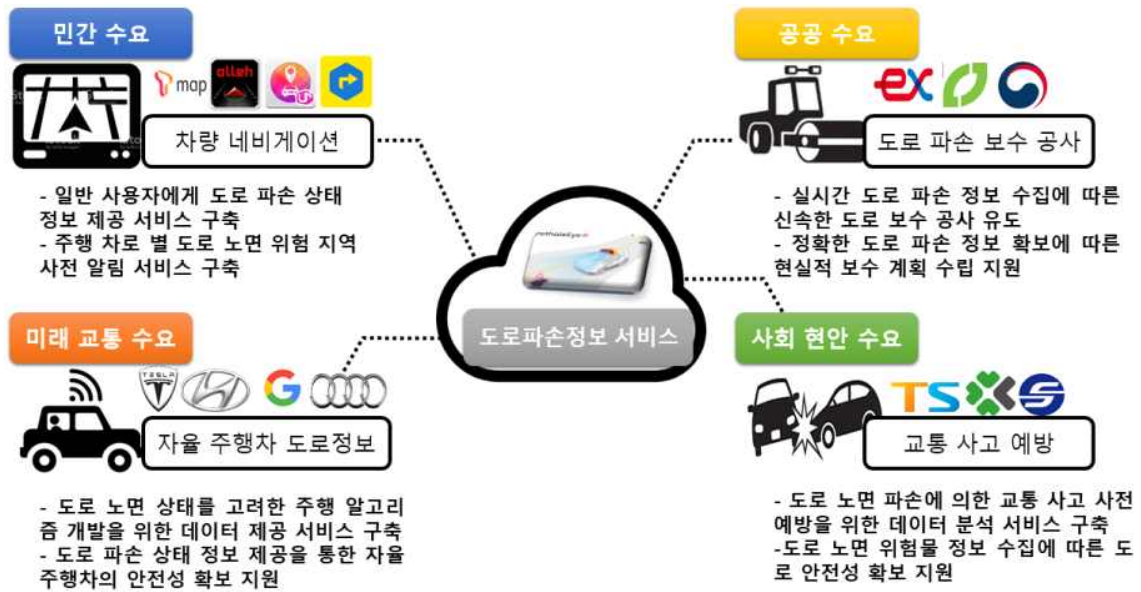
[국토교통부 7대 서비스 분야]

- 국토부교통부가 제안한 7대 서비스 분야는 핵심 기능별로 총 15개로 구분
- 기본정보 수집제공 분야는 위치 기반의 차량 데이터 수집 기능과 위치기반 교통정보 제공 기능으로 구성되어 있는데, 이 기능을 수행하기 위해서 차량 네비게이션을 활용한 도로 노면 파손 정보 제공 서비스 사업화를 추진하고자 함
- 안전(주의) 운전지원 분야는 도로 위험구간 정보 제공 기능, 노면 상태 기상정보 제공 기능, 도로 작업구간 주행지원 기능으로 구성되어 있는데, 이기능을 수행하기 위해서 민간 및 공공 기관을 대상으로 한 도로 노면 파손 정보 제공 서비스 사업화를 추진하고자 함
- 차량간 사고예방 분야는 긴급차량 접근경고 기능과 차량 긴급상황 경고 기능으로 구성되어 있는데, 이 기능을 수행하고자 자율 주행차의 안전성 향상을 위한 도로

파손 정보 제공 서비스 사업화를 추진하고자 함

- 또한, 차량간 사고예방 분야는 차량 추돌방지 지원 기능이 있는데, 이 기능을 수행하고자 도로 노면 파손에 의한 교통사고 예방 전략 수립 지원 사업화를 추진하고자 함

2. 연구목표



<사업화 서비스 모델>

구분	내 용
최종 목표	교통사고 예방을 위한 도로 파손 정보 시스템 구축 및 사업화
세부 목표	- 민간 및 공공 기관을 대상으로 한 도로 노면 파손 제공 기술 사업화 * 실시간 도로 파손 정보 수집에 따른 신속한 도로 보수 공사 유도 * 정확한 도로 파손 정보 확보에 따른 현실적 보수 계획 수립 지원 - 차량 네비게이션을 활용한 도로 노면 파손 정보 제공 기술 사업화 * 일반 사용자에게 도로 파손 상태 정보 제공 기술 구축 * 주행 차로별 도로 노면 위험 지역 사전 알림 시스템 구축 - 초저지연 기술기반의 도로 노면 상태 정보 제공 서비스 구현 * 통신망과 연계한 인공지능 기술 기반의 도로 노면 상태 정보 제공 * 인공지능 기술을 활용하여 도로 노면 정보에 대한 데이터 확보 * 향후 지역 내 운전자에게 공유할 수 있는 교통 정보 플랫폼 제공 - 자율 주행차의 안전성 향상을 위한 도로 파손 정보 제공 기술 사업화 * 도로 노면 상태를 고려한 주행 알고리즘 개발을 위한 데이터 제공 기술 개발 * 도로 파손 상태 정보 제공을 통한 자율 주행차의 안전성 확보 지원
활용계획	- 다수의 수집단말기를 활용한 시공간적으로 단절 없는 도로 노면상태 정보 수집 - 지자체/도로관리공사에 정보 제공을 통한 신속 보수체계 구축 - 탐지된 노면 정보를 실시간으로 피드백하여 안전운전을 위한 보조서비스로 활용 - 도로 노면 파손 정보 제공을 통한 안전운전 유도로 교통사고 감소 가능

2. 연구수행내용 및 성과

1. 개요

- 본 문서는 포트홀 자동 탐지 시스템을 구성하고 있는 단말, 수집서버, 1차 DB, Sender, RabbitMQ, Receiver, 2차 DB, 관제서비스에 대한 구성 내용에 대해 기술한다.

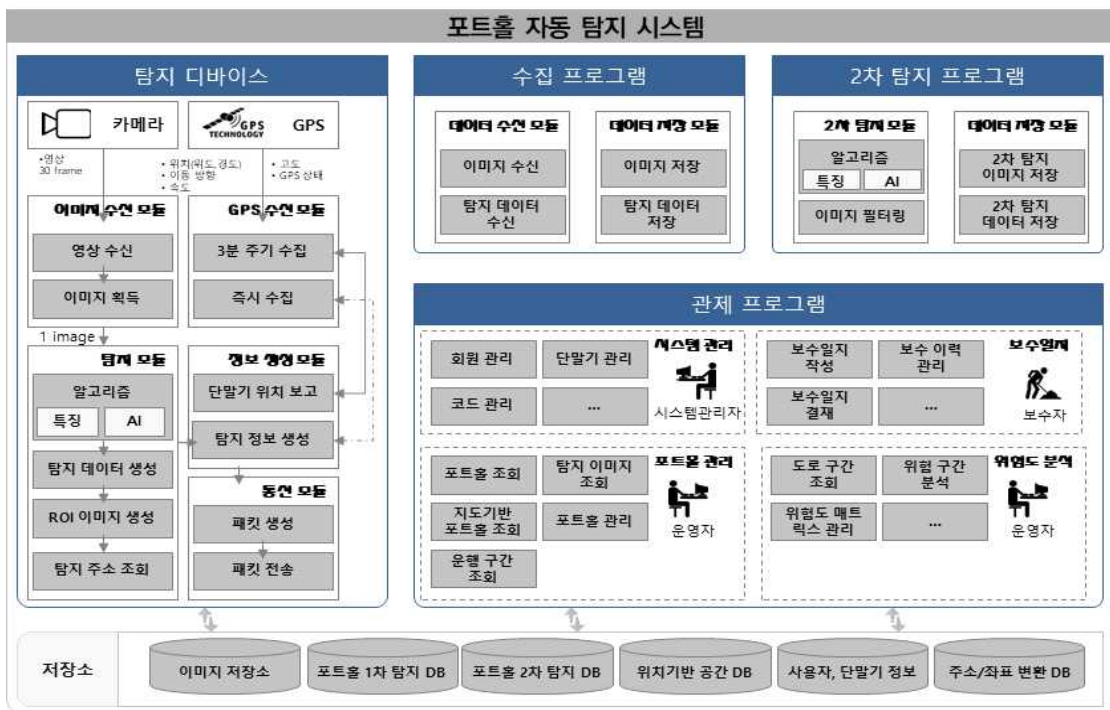
2. 구축범위

- 탐지 차량에 장착된 단말기에서 탐지된 포트홀 정보 데이터 실시간 수집 기능 구축
- 수집된 GPS위도(WGS84), GPS경도(WGS84), 탐지주소, 탐지일시, 탐지크기, 속도, 방향, 탐지사진 정보를 DATABASE로 저장 기능 구축
- 수집된 GPS위도(WGS84), GPS경도(WGS84) 위치정보 주소 변환 기능 구축
- 지도기반 포트홀 실시간 위치정보 제공 및 이력 조회 기능 구축
- 탐지된 포트홀에 대한 GPS위도(WGS84), GPS경도(WGS84), 탐지주소, 탐지일시, 탐지크기, 속도, 방향, 탐지사진, G센서 데이터 등의 상세 정보 모니터링 기능 구축
- 탐지된 포트홀에 대한 분류 긴급, 접수, 완료, 삭제 등 분류 기능 구축
- 웹 기반 포트홀 이미지 뷰어 기능 구축
- 탐지된 포트홀에 보수 일정 작성 및 보수 기록 관리 기능 구축
- 허가된 사용자만 프로그램에 접근 하여 사용 할 수 있는 권한 기능 구축
- 관제서비스 사용자 관리 기능 구축
- 전용 탐지 단말기 관리 기능 구축
- 수집된 포트홀 데이터를 기반으로 도로 구간별 포트홀 위험도 분석 및 지도 기반 위험 등급 제공 기능 구축

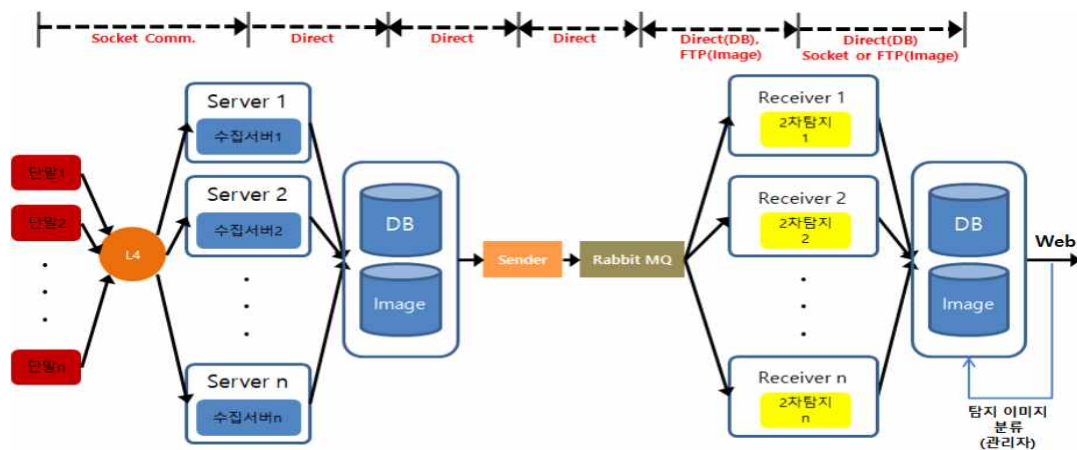
3. 시스템 구성

3.1. 시스템 구성

- 포트홀 자동탐지 시스템은 2단계 탐지구조로 구성되어 있으며 단말, 수집서버, 1차 DB, Sender, RabbitMQ, Receiver, 2차 DB, 관제서비스 등으로 구성된다.
- 1차 탐지를 통해 저장된 포트홀 이미지 및 GPS 데이터 등은 보다 정확한 판별을 위해 2차 탐지 알고리즘이 적용된 2차 서버를 통해 필터링 되어 2차 DB 및 2차 이미지 저장소에 저장되어 최종적으로 관제서비스에 표출되는 형태로 구성되어 있다.



[전체 시스템 구성도]



[포트홀 탐지 구성도]

가. 단말

단말은 Jetson TX2 기반으로 제작된 전용 단말로서 카메라, GPS 수신기 등과 함께 차량에 설치되어 인공지능기반의 알고리즘이 탑재된 응용프로그램을 통해 1차적으로 포트홀 정보를 탐지하여 탐지된 이미지와 GPS 데이터 등을 수집서버로 전송하는 역할을 한다.



나. 수집서버

수집서버는 단말에서 1차적으로 탐지된 포트홀 이미지 및 GPS 데이터 등을 수신하기 위한 역할을 하는 프로그램으로 소켓 통신을 통해 단말로부터 수신한 포트홀 이미지 및 GPS 데이터 등을 1차 이미지 저장소와 1차 DB에 저장하는 역할을 수행한다.

다. 1차 DB 및 1차 이미지 저장소

수집서버로부터 수집된 GPS 데이터 등을 저장하기 위한 1차 DB와 1차 탐지이미지 및 원본 이미지를 저장하기 위한 1차 이미지 저장소로 구성된다.

라. Sender

Sender는 2차 탐지 서버에 위치하여 1차 DB 및 1차 이미지 저장소로부터 데이터 및 이미지를 가져와서 메시지 큐의 일종인 RabbitMQ를 통해 Receiver에 일감을 분배해 주는 역할을 한다.

마. RabbitMQ



[수집프로그램 구성도]

RabbitMQ는 메시지 큐의 일종으로서 Sender가 보내온 일감을 여러 개의 Receiver에 분배해 주는 역할을 한다.

바. Receiver

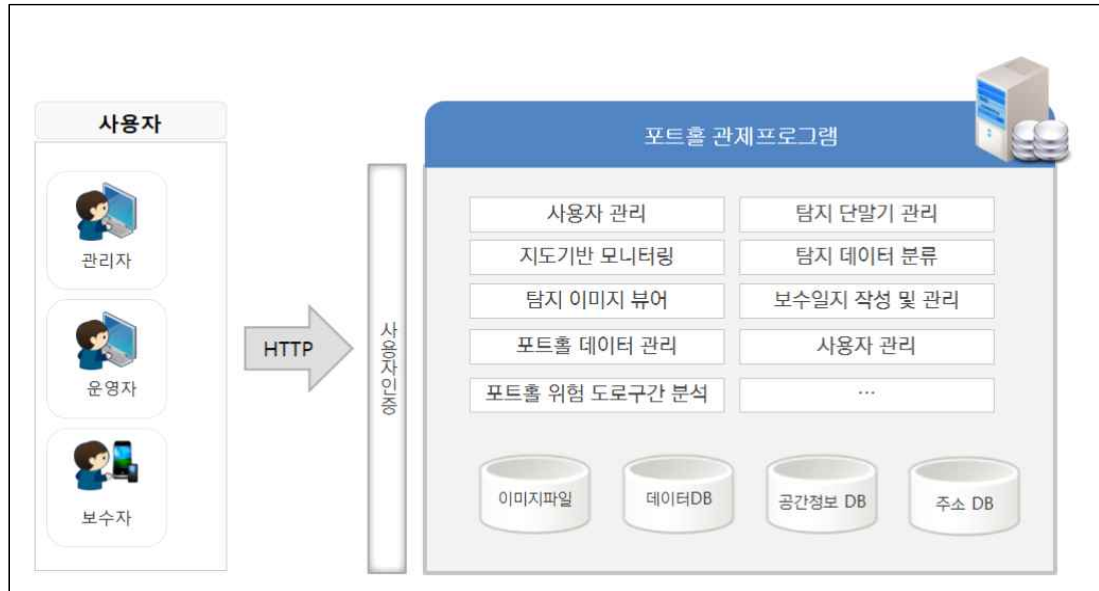
Receiver는 여러 개의 프로세스로 구성되며, 각각의 Receiver는 RabbitMQ로부터 일감을 할당받아 2차 탐지 알고리즘을 적용하여 1차에서 탐지된 포트홀을 좀 더 정확하게 필터링하여 2차 DB 및 2차 이미지 저장소에 전달하는 역할을 수행한다.

사. 2차 DB 및 2차 이미지 저장소

Receiver를 통해 필터링된 데이터 및 이미지를 저장하는 역할을 수행한다.

아. 관제서비스

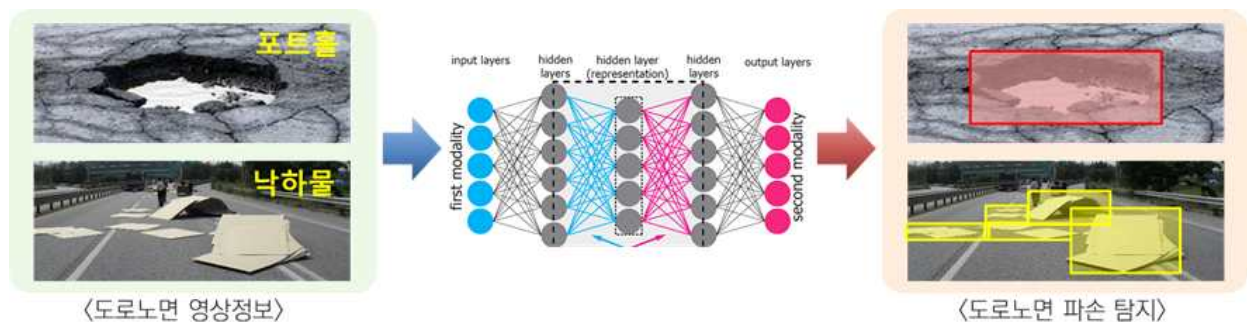
Receiver를 통해 필터링되어 저장된 2차 DB 및 2차 이미지 저장소의 포트홀 정보가 최종적으로 관제서비스를 통해 표시된다.



[관제프로그램 구성도]

자. 인공지능 기반의 노면파손 정보 수집 기술

- 차량 내에 블랙박스와 비슷한 형태로 거치된 스마트폰과 같은 단말기를 활용하여 노면 파손 정보를 수집하는 기술을 개발함. 개발된 기술은 비전 카메라를 활용하여 획득한 이미지를 딥러닝 기반으로 분석하여 노면파손을 분석하고, GPS 정보와 함께 서버로 전송되어 주행 중에 실시간으로 노면파손 정보를 수집함



[인공지능 기반의 노면파손 정보 수집기술]

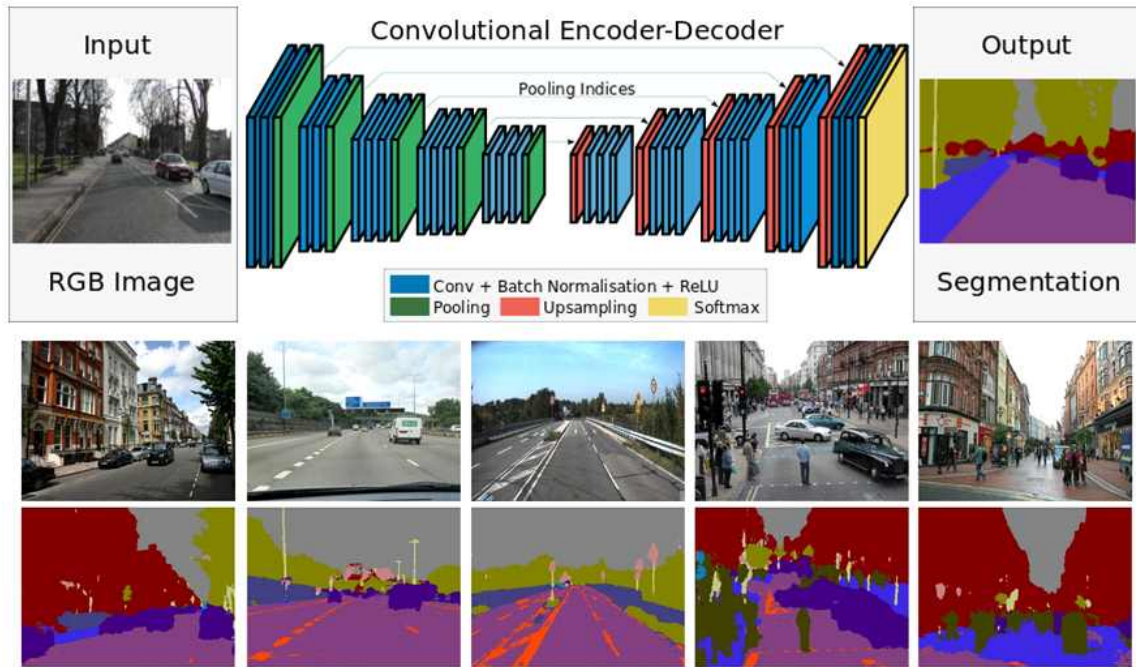


그림 11 노면파손 정보 수집을 위한 의미론적 분할 구조 예시

3.2. 시스템 구성 제원

가. 시스템 구성 제원(하드웨어)

Name	Resource Name	Type	Contents	Spec
WEB#1	POTHOLE01-WEB-01	VM	CentOs 7.5 (OpenLogic)	Standard_F1(v1core, 2GB Memory)
DB#1	pothole01-sql	PASS	SQL PASS	
APP#1	POTHOLE01-APP-01	VM	CentOs 7.5 (OpenLogic)	Standard_F1(v1core, 2GB Memory)
APP#2	POTHOLE01-APP-02	VM	CentOs 7.5 (OpenLogic)	Standard_F1(v1core, 2GB Memory)
STORAGE#1	pothole01storage	PASS	Blob Storage	
GPU#1	GPU SuperWorkstation7049 GP-TRT	Server	Ubuntu 18.04 Server 64bit	

나. 시스템 구성 제원(소프트웨어)

Name	Resource Name	Contents
WEB#1	POTHOLE01-WEB-01	- CentOs 7.5 - JDK 1.7.0_80 - Apache 2.4.35 - Tomcat 7.0.91 - tomcat-connectors 1.2.46
APP#1	POTHOLE01-APP-01	- CentOs 7.5 - OPEN JDK 1.7.0_191
APP#2	POTHOLE01-APP-02	- CentOs 7.5 - OPEN JDK 1.7.0_191
DB#1	pothole01-sql	- MS Azure (RTM) - 12.0.2000.8
GPU#1	GPU SuperWorkstation7049GP-TRT	- Python 3.6.7 - CUDA 10.0 - Cudnn 7.5.1 - Tensorflow-gpu 1.13.1 - TensorRT 5.0 - OpenCV 3.4.2 - Keras 2.2.4 - RabbitMQ 3.6.10

3.3. 주요 디렉토리 구성

가. WEB/WAS서버 주요 디렉토리 구조

디렉터리 명	설명
/usr/local/java/jdk1.7.0_80/	JAVA HOME
/usr/local/apache/	아파치 서버
/usr/local/apache/bin/	아파치 실행 파일
/usr/local/apache/conf/	아파치 설정 파일
/usr/local/apache/logs/	아파치 로그 파일
/usr/local/tomcat/	톰캣 서버
/usr/local/tomcat/bin/	톰캣 실행 파일
/usr/local/tomcat/conf/	톰캣 설정 파일
/usr/local/tomcat/logs/	톰캣 로그 파일
/usr/local/webapp/kictpothole/	관제서비스 소스 위치
/mnt/pothole01/	STORAGE#1 Mount 위치
/mnt/pothole01/pothole/download/	포트홀 이미지 파일 위치

나. 수집서버 주요 디렉토리 구조

디렉터리 명	설명
/usr/lib/jvm/java	JAVA HOME
/home/azadmin/PotHoleServer_v2.jar	포트홀 수집 프로그램 소스
/home/azadmin/log	수집 로그 디렉토리
/home/azadmin/config.properties	수집 프로그램 설정파일

다. 2차 탐지 서버 주요 디렉토리 구조

디렉터리 명	설명
/home/user/pothole	2차 탐지 프로그램 소스 디렉토리

3.4. DATABASE 구성

가. DATABASE 정보

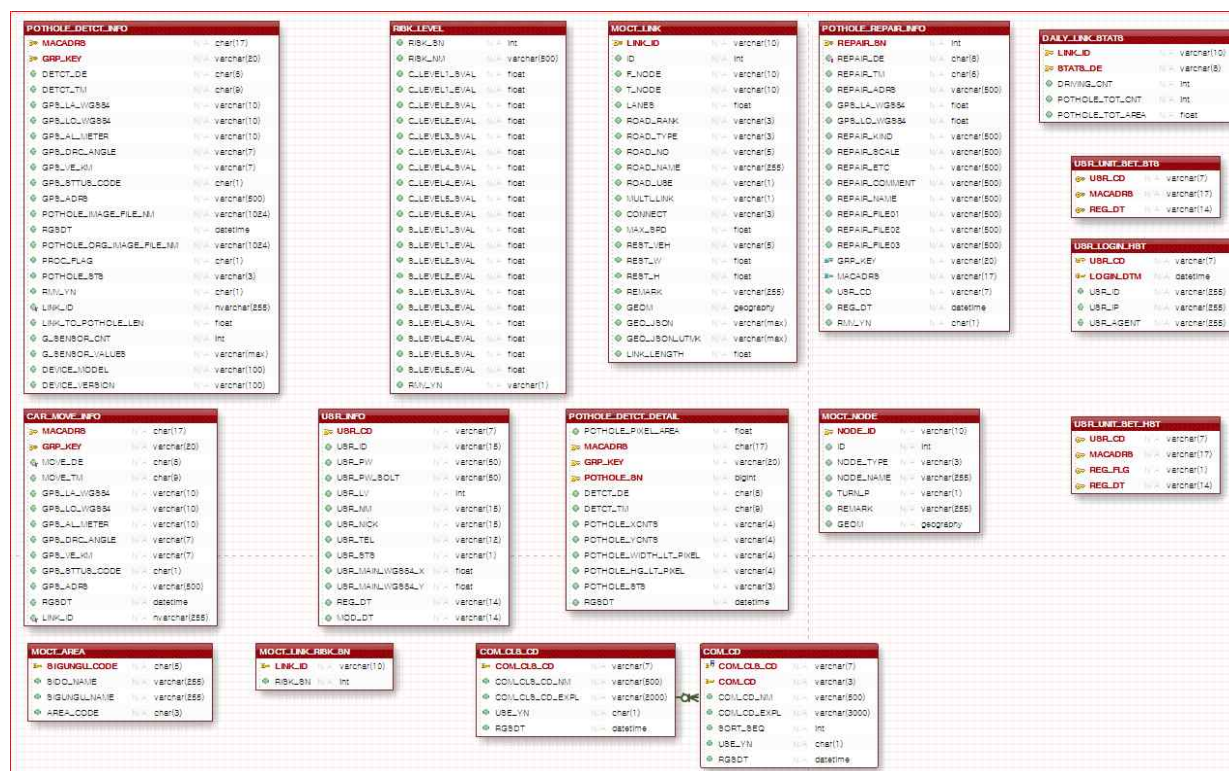
- DATABASE는 실시간 수집 포트홀 정보, 사용자 정보, 단말기 정보, 보수 정보를 저장하는 MS Azure Sql Server로 구성 된다.

MS Azure SQL	POTHOLE01-SQL-DB01	POTHOLE01-SQL-DB02
VERSION	12.0.2000.8	12.0.2000.8
TIMEZONE	UTC	UTC
TABLE COUNT	19 Tables	2 Tables
COMMENT	2차 탐지 포트홀 데이터 및 관제서비스 데이터	1차 탐지 포트홀 데이터

4. Entity Relationship Diagram



[MS Azure SQL Logical ERD]



[MS Azure SQL Physical ERD]

4. 관제프로그램 구축 내용

4.1. 화면 구성

- 포트홀 관제 프로그램은 크게 탐지된 포트홀 정보를 지도기반으로 실시간 모니터링 하는 ‘운행현황’, 포트홀에 대한 상세정보 목록, 이미지 뷰어를 통한 목록조회 및 분류 기능을 제공하는 ‘탐지기록’, 포트홀에 대한 보수일지 입력 및 관리 하는 ‘보수일지’, 도로구간별 포트홀 발생 위험도를 분석하여 표출 ‘위험구간’, 관제프로그램에 대한 사용자를 관리하고 사용자에게 배정된 포트홀 탐지 단말기와 공통코드를 관리하는 ‘관리’ 메뉴로 구성되어 있으며 해당 메뉴에 대한 화면 구성은 아래에 정의한다.



[포트홀 관제프로그램 메뉴 구성도]

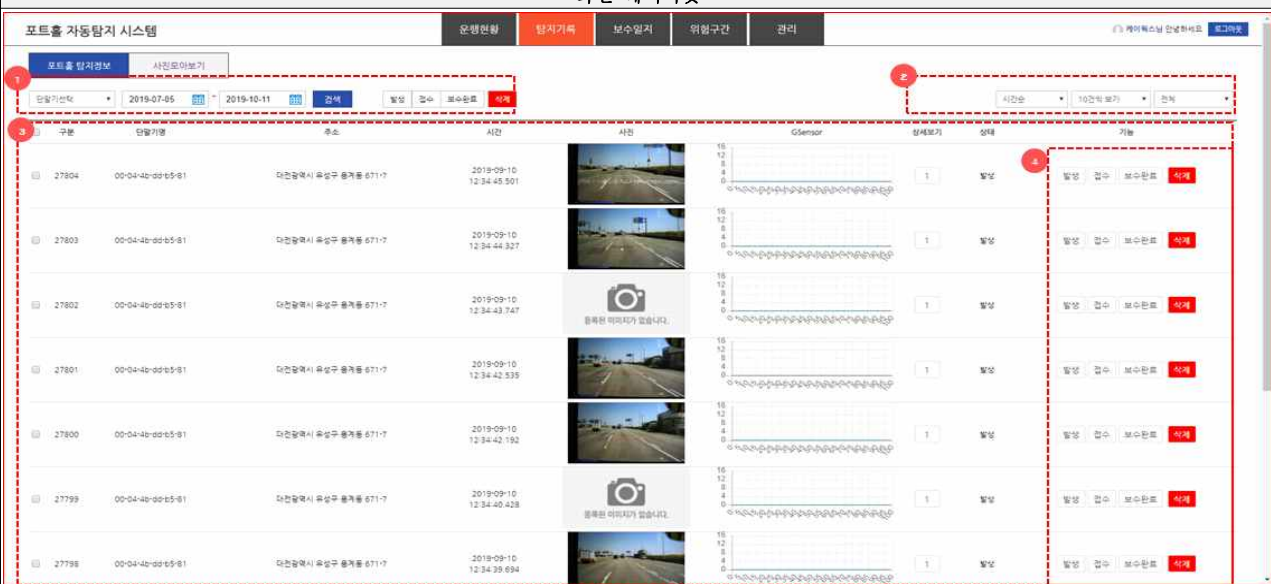
가. 로그인

화면 ID	L_WEB_LOGIN	화면 명	로그인
화면 레이아웃			
화면 설명			
1	접속 허가된 사용자의 ID와 비밀번호를 입력 한 후 [Login]버튼을 클릭하여 관제서비스에 관한 인증을 요청한다. 인증에 성공하면 관제서비스 운행현황 메뉴로 이동된다.		

나. 운행현황

화면 ID	L_WEB_DVSTAT	화면 명	운행현황
화면 레이아웃			
 The screenshot shows the L_WEB_DVSTAT interface. At the top, there's a navigation bar with '운행현황' (Operation Status) selected. Below it, a map displays a road network with a highlighted route. A table at the bottom lists vehicle data with columns for '순번' (Serial Number), '단말기명' (Device Name), '보고시간' (Report Time), '주소' (Address), '속도' (Speed), '발차번호' (Departure Number), and '사진' (Photo). The table contains 5 rows of data.			
화면 설명			
1	조회하고자 하는 단말기를 선택하고 포트홀 탐지 일시를 입력 한 후 [검색]버튼을 클릭하여 일시 별로 탐지 된 포트홀 정보를 조회한다.		
2	조회된 포트홀 목록에서 check box를 체크하여 [선택삭제] 버튼을 클릭하면 선택 된 포트홀을 탐지 정보를 삭제 할 수 있다.		
3	조회된 포트홀 목록을 출력한다.(순번, 단말기명, 보고시간, 주소, 좌표, 탐지갯수, 사진)		
4	조회된 포트홀 탐지 위치를 지도상에 표시한다.		
5	지도상의 포트홀 위치 Marker를 클릭하면 해당 포트홀의 상세정보를 확인 할 수 있다.		

다. 포트홀 정보

화면 ID	L_WEB_RECHISTROW	화면 명	포트홀 정보
화면 레이아웃			
 The screenshot shows the L_WEB_RECHISTROW interface. It features a table of pothole records with columns for '구분' (Category), '단말기명' (Device Name), '주소' (Address), '시간' (Time), '사진' (Photo), 'GSensor' (Graph), '상태' (Status), and '행동' (Action). The table contains 8 rows of data. To the right of the table, there's a sidebar with buttons for '발생' (Occurrence), '접수' (Receipt), '보수완료' (Maintenance Complete), and '삭제' (Delete).			
화면 설명			
1	기간에 따른 포트홀 탐지 정보를 조회한다. 조회된 목록에서 check된 포트홀에 대해 발생, 접수, 보수완료, 삭제 로 일괄 분류 기능을 제공한다.		
2	시간, 상태순으로 정렬 기능을 제공한다. 10,30,50,100건씩 조회 기능을 제공한다. 발생, 접수, 보수완료, 전체로 필터링 조회 기능을 제공한다.		
3	조회된 포트홀 목록을 출력한다. 주소필드에 마우스 Over시 위치정보를 보여준다. 사진을 클릭하면 이미지 뷰어 팝업으로 사진을 출력한다.		
4	상세보기 클릭시 해당 위치에 탐지된 포트홀들의 크기를 조회할 수 있다. GSensor 그래프 클릭시 그래프 뷰어로 GSensor 데이터를 출력한다.		
5	발생, 접수, 보수완료, 삭제로 포트홀 정보를 분류하는 기능을 제공한다.		

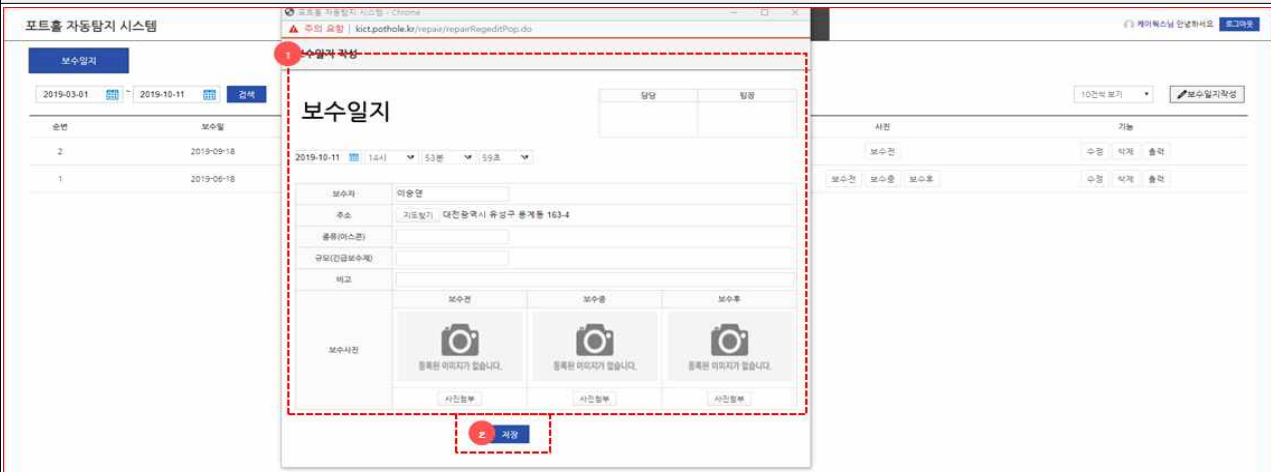
라. 사진모아보기

화면 ID	L_WEB_RECHISTPHOTO	화면 명	사진모아보기
화면 레이아웃			
화면 설명 1 기간에 따른 포트홀 탐지 정보 사진 및 상태를 조회한다. 조회된 목록에서 check된 포트홀에 대해 발생, 접수, 보수완료, 삭제 로 일괄 분류 기능을 제공. 2 시간, 상태순으로 정렬 기능을 제공한다. 발생, 접수, 보수완료, 전체로 필터링 조회 기능을 제공한다. 3 조회된 포트홀 목록을 이미지 뷰어 형태로 출력한다.			

마. 보수일지

화면 ID	L_WEB_REPAIRINFO	화면 명	보수정보
화면 레이아웃			
화면 설명 1 기간에 따른 포트홀 보수정보를 조회한다. 2 10,30,50,100건씩 조회 기능을 제공한다. 3 조회된 보수일지 목록을 출력한다. 4 [보수일지작성]버튼을 클릭하여 보수일지작성 팝업을 호출한다.			

바. 보수일지작성

화면 ID	L_WEB_REPAIRREPORT	화면 명	보수일지작성
화면 레이아웃			
			
화면 설명			
1	보수일지, 보수자, 주소, 종류, 규모, 비고, 보수사진의 내용으로 포트홀 보수일지 내용을 입력한다.		
2	[저장]버튼을 클릭하여 작성한 보수일지를 DATABASE에 저장한다.		

사. 위험구간분석

화면 ID	L_WEB_ROADANALYSIS	화면 명	위험구간분석
화면 레이아웃			
			
화면 설명			
1	시도, 시군구, 도로종류, 조회기간에 따른 포트홀 위험도로 구간 공간정보를 조회한다.		
2	위험등급에 대한 범례를 제공하며 check box 선택에 따른 공간정보 필터링 기능을 제공한다. Matrix관리 버튼을 클릭하여 위험도 분석에 사용되는 위험도 Matrix를 관리하는 팝업화면을 호출한다.		
3	조회된 포트홀 위험 도로구간에 대한 위험등급 별 건수를 제공한다.		
4	웹지도화면을 제공하며, [1]에서 조회된 시군구 경계 공간정보 Polygon과 위험 등급에 따른 도로구간 공간정보 Polyline(국토교통부의 표준노드 링크 공간정보)을 지도상에 표출한다. 도로구간 공간 정보는 포트홀 위험 등급에 따라 색상으로 구분된다.		
5	조회된 도로구간 공간정보 Polyline을 클릭하면 클릭한 도로구간에 대한 상세정보를 제공하는 팝업 화면을 호출한다.		
	해당 구간에 적용된 위험도 가중치 Matrix를 변경 할 수 있다.		

아. 사용자 관리

화면 ID	L_WEB_USR	화면 명	사용자 관리
화면 레이아웃			

화면 설명

1	사용자 ID, 사용자명, 사용자닉네임 별로 사용자를 조회하는 기능을 제공한다.
2	조회된 사용자 목록을 표출한다.
3	사용자에 배정된 단말기의 개수를 표출하고 해당 개수 버튼을 클릭하면 배정단말기 관리 화면으로 이동한다. 수정버튼을 클릭하면 해당 사용자를 수정 할 수 있는 수정팝업 화면을 호출한다. 삭제버튼을 클릭하여 등록된 사용자를 삭제하는 기능을 제공한다.
4	사용자등록 버튼을 클릭하여 사용자등록 팝업화면을 호출해 신규 사용자 정보를 입력하여 신규 사용자를 등록 하는 기능을 제공한다.
5	수정 버튼을 클릭하여 사용자수정 팝업화면을 호출해 해당 사용자 정보를 수정하는 기능을 제공한다.

자. 배정단말기 관리

화면 ID	L_WEB_MAC	화면 명	배정단말기 관리
화면 레이아웃			

화면 설명

1	사용자관리 화면의 사용자 목록에서 배정단말기 필드를 클릭하여 해당 사용자의 배정단말기 관리 팝업화면을 호출한다.
2	해당 사용자에게 배정할 단말기의 MAC 주소와 정보를 입력하고 등록 버튼을 클릭하여 사용자에게 단말기를 배정한다.
3	해당사용자에게 배정된 단말기 목록을 표출한다. 수정 하는 기능을 제공하며, 배정된 단말기를 삭제하는 기능을 제공한다.

차. 코드관리

화면 ID

L_WEB_ROADANALYSIS

화면 명

위험구간분석

화면 레이아웃

포트홀 자동탐지 시스템

운영현황

탐지기록

보수일지

위험구간

관리

게이웍스님 안녕하세요로그아웃

사용자관리

코드관리

1

2

분류코드명

3

4

분류코드등록

순번	분류코드	분류코드명	설명	최초등록일	등록코드	기능
5	A000005	위험등급코드	도로위험등급코드	2019-08-07	6	수정
4	A000004	연결코드	연결코드(CONNECT)	2019-08-07	9	수정
3	A000003	도로유형코드	도로유형코드(ROAD_TYPE)	2019-08-07	5	수정
2	A000002	도로등급코드	도로등급코드(ROAD_RANK)	2019-08-07	8	수정
1	A000001	포트홀상태구분코드	포트홀상태구분코드	2019-08-06	3	수정

1

화면 설명

1

분류코드명, 분류코드 별로 사용자를 조회하는 기능을 제공한다.

2

조회된 코드 목록을 표시한다.

3

분류별 등록된 코드 개수를 표시하고 해당 개수 버튼을 클릭하면 등록코드 관리 화면으로 이동한다. 수정버튼 클릭시 분류코드 정보를 수정 할 수 있다.

4

분류코드등록 버튼을 클릭하여 신규 분류코드를 생성 한다.

화면 설명

- 1 분류코드명, 분류코드 별로 사용자를 조회하는 기능을 제공한다.
- 2 조회된 코드 목록을 표출한다.
- 3 분류별 등록된 코드 개수를 표출하고 해당 개수 버튼을 클릭하면 등록코드 관리 화면으로 이동한다. 수정버튼 클릭시 분류코드 정보를 수정 할 수 있다.
- 4 분류코드등록 버튼을 클릭하여 신규 분류코드를 생성 한다.

4.2. 프로그램 구성

가. 관제서비스 접속 권한 인증

프로그램ID	P_WEB_LOGIN	프로그램명	운영현황 관리
관련화면ID	L_WEB_LOGIN	관련화면명	로그인
개요	포트홀 관제프로그램에 등록 된 사용자인지 조회하고 사용자의 권한을 조회한다.		
처리 로직			
1. 로그인 접속 권한 인증			
1) 처리로직			
① 사용자가 ID, 비밀번호를 입력하여 시스템에 접속 권한을 요청한다.			
② 시스템은 사용자가 입력한 ID와 비밀번호(SHA256 알고리즘으로 암호화)를 인자 값으로 [USR_INFO] 테이블에 해당 사용자 정보가 존재하는지 조회한다.			
③ 사용자 존재 시 WEB서버에 SESSION을 생성하고 [운영현황] 화면으로 Redirect 한다.			
④ 존재하지 않는 사용자일 경우 ‘로그인 정보가 존재하지 않습니다.’ 메시지를 출력한다.			
2) 파라미터 체크			
① 필수 : ID, 비밀번호			
2) 관련테이블			
① USR_INFO(사용자정보) : 조회			

나. 운영현황 관리

프로그램ID	P_WEB_DVSTATMNG	프로그램명	운영현황 관리
관련화면ID	L_WEB_DVSTAT	관련화면명	운영현황
개요	지도 기반으로 실시간 포트홀 탐지 데이터를 조회 및 출력한다.		
처리 로직			
1. 포트홀 탐지 목록/상세 조회			
1) 처리로직			
① 사용자가 조회 할 날짜, 시간을 입력하여 포트홀 탐지 정보를 조회 요청한다.			
② 시스템은 사용자가 입력한 날짜, 시간과 로그인 Session에 저장된 사용자의 ID를 인자 값으로 [POTHOLE_DETCT_INFO], [POTHOLE_DETCT_DETAIL], [USR_INFO], [USR_UNIT_SET_STS] 테이블에서 사용자에게 배정된 단말기에서 수집된 포트홀 데이터를 조회한다.			
③ 조회된 데이터를 화면상에 LIST로 출력한다. (출력필드 : 순번, 단말기명, 보고시간, 주소, 좌표, 탐제갯수, 사진)			
④ 조회된 데이터를 지도상에 탐지시간 순서대로 넘버링 된 Marker Object로 출력한다.			
⑤ 사용자는 ④에서 출력된 지도상의 포트홀 Marker Object를 클릭한다.			
⑥ 시스템은 사용자가 클릭한 포트홀 Marker Object의 상세 정보를 팝업으로 출력한다.			
2) 파라미터 체크			
① 필수 : 단말기 MAC주소, 조회일(YYYYMMDD), 조회시간(00~23)			
2) 관련테이블			
① USR_INFO(사용자정보) : 조회			
② USR_UNIT_SET_STS(사용자단말기배정상태) : 조회			
③ POTHOLE_DETCT_INFO(포트홀탐지정보) : 조회			
④ POTHOLE_DETCT_DETAIL(포트홀탐지상세) : 조회			
2. 포트홀 탐지 정보 삭제			
1) 처리로직			
① 사용자는 1.에서 출력된 List 데이터의 Check Box를 선택하여 선택삭제 버튼을 클릭하여 선택된 데이터에 대한 삭제 처리를 요청한다.			
② 시스템은 사용자가 선택한 Check Box의 포트홀 그룹키와 MAC주소를 인자 값으로 [POTHOLE_DETCT_INFO] 테이블의 삭제여부(RMV_YN) 컬럼의 데이터를 'Y' 삭제 상태로 Update 한 후 성공 여부에 대한 메시지를 사용자에게 리턴한다.			
2) 파라미터 체크			
① 필수 : 그룹키, MAC주소			
2) 관련테이블			
① POTHOLE_DETCT_INFO(포트홀탐지정보) : 수정			

다. 포트홀 정보 관리

프로그램ID	P_WEB_RECHISTROW	프로그램명	포트홀 정보 관리
관련화면ID	L_WEB_RECHISTROW	관련화면명	포트홀 정보
개요	포트홀 탐지 데이터를 게시판 형태로 조회 및 검색, 정렬 기능을 제공하며 데이터에 대한 발생,접수,보수완료,삭제 등 분류 기능을 제공한다.		
처리 로직			
1. 포트홀 탐지 목록 조회			
1) 처리로직			
① 사용자가 조회 할 조회시작일, 조회종료일을 입력 및 정렬기준, 1페이지 출력 데이터 개수, 데이터 구분을 선택하여 포트홀 탐지 정보를 조회 요청한다.			
② 시스템은 사용자가 입력 및 선택한 값과 로그인 Session에 저장된 사용자의 ID를 인자 값으로 [POTHOLE_DETCT_INFO], [POTHOLE_DETCT_DETAIL], [USR_INFO], [USR_UNIT_SET_STS] 테이블에서 사용자에게 배정된 단말기에서 수집된 포트홀 데이터를 조회한다.			
③ 조회된 데이터를 화면상에 LIST로 출력한다. (출력필드 : 구분, 주소, 시간, 크기, 사진, 상태, 기능)			
2) 파라미터 체크			
① 필수 : 단말기MAC주소, 조회시작일(YYYYMMDD), 조회종료일(YYYYMMDD)			
② 선택 : 정렬기준(시간, 상태), 1페이지 출력 데이터 개수(10, 30, 50, 100), 분류(전체, 발생, 접수, 보수완료)			
2) 관련테이블			
① USR_INFO(사용자정보) : 조회			
② USR_UNIT_SET_STS(사용자단말기배정상태) : 조회			
③ POTHOLE_DETCT_INFO(포트홀탐지정보) : 조회			
④ POTHOLE_DETCT_DETAIL(포트홀탐지상세) : 조회			
2. 포트홀 탐지 데이터 분류 처리			
1) 처리로직			
① 사용자는 1.에서 출력된 List 데이터의 기능 필드의 발생, 접수, 보수완료, 삭제 기능을 사용하여 포트홀 데이터 분류를 요청한다.			
② 시스템은 사용자가 요청한 값을 인자 값으로 [POTHOLE_DETCT_INFO] 테이블의 분류코드(POTHOLE_STS) 컬럼 또는 삭제여부(RMV_YN) 컬럼의 데이터를 Update 한 후 성공 여부에 대한 메시지를 사용자에게 리턴한다.			
2) 파라미터 체크			
① 필수 : 그룹키, MAC주소, 분류(발생, 접수, 보수완료) 또는 삭제 FLAG			
2) 관련테이블			
① POTHOLE_DETCT_INFO(포트홀탐지정보) : 수정			
3. 포트홀 탐지 데이터 일괄 분류 처리			
1) 처리로직			
① 사용자는 1.에서 출력된 List 데이터의 Check 필드의 Check박스를 선택하여 선택된 데이터에 대한 포트홀 데이터 일괄 분류를 요청한다.			
② 시스템은 사용자가 요청한 값을 인자 값으로 [POTHOLE_DETCT_INFO] 테이블의 분류코드(POTHOLE_STS) 컬럼 또는 삭제여부(RMV_YN) 컬럼의 데이터를 Update 한 후 성공 여부에 대한 메시지를 사용자에게 리턴한다.			
2) 파라미터 체크			
① 필수 : 그룹키(Array), MAC주소(Array), 포트홀순번(Array), 분류(발생, 접수, 보수완료) 또는 삭제 FLAG			
2) 관련테이블			
① POTHOLE_DETCT_INFO(포트홀탐지정보) : 수정			

라. 포트홀 이미지 뷰어

프로그램ID	P_WEB_RECHISTPHOTO	프로그램명	포트홀 이미지 뷰어
관련화면ID	L_WEB_RECHISTPHOTO	관련화면명	사진모아보기
개요	포트홀 탐지 데이터를 이미지 뷰어 형태로 조회 및 검색, 정렬 기능을 제공하며 데이터에 대한 긴급, 접수, 완료, 삭제 등 분류 기능을 제공한다. 그룹별 포트홀 데이터 조회 기능을 제공한다.		
처리 로직			
1. 포트홀 탐지 그룹 목록 조회			
1) 처리로직			
① 사용자가 조회 할 조회시작일, 조회종료일을 입력 및 정렬기준, 데이터 구분을 선택하여 포트홀 탐지 정보를 조회 요청한다.			
② 시스템은 사용자가 입력 및 선택한 값과 로그인 Session에 저장된 사용자의 ID를 인자 값으로 [POTHOLE_DETCT_INFO], [POTHOLE_DETCT_DETAIL], [USR_INFO], [USR_UNIT_SET_STS] 테이블에서 사용자에게 배정된 단말기에서 수집된 포트홀 데이터를 조회한다.			
③ 조회된 데이터를 화면상에 이미지 List로 출력한다. (출력필드 : 시간, 갯수, 상태, 이미지 Object)			
2) 파라미터 체크			
① 필수 : 단말기MAC주소, 조회시작일(YYYYMMDD), 조회종료일(YYYYMMDD)			
② 선택 : 정렬기준(시간, 상태, 크기), 분류(발생, 접수, 보수완료)			
2) 관련테이블			
① USR_INFO(사용자정보) : 조회			
② USR_UNIT_SET_STS(사용자단말기배정상태) : 조회			
③ POTHOLE_DETCT_INFO(포트홀탐지정보) : 조회			
④ POTHOLE_DETCT_DETAIL(포트홀탐지상세) : 조회			
2. 포트홀 탐지 데이터 일괄 분류 처리			
1) 처리로직			
① 사용자는 1.에서 출력된 이미지 List 데이터의 이미지를 선택 체크하여 선택 체크된 데이터에 대한 포트홀 데이터 일괄 분류를 요청한다.			
② 시스템은 사용자가 요청한 값을 인자 값으로 [POTHOLE_DETCT_INFO] 테이블의 분류코드(POTHOLE_STS) 컬럼 또는 삭제여부(RMV_YN) 컬럼의 데이터를 Update 한 후 성공 여부에 대한 메시지를 사용자에게 리턴한다.			
2) 파라미터 체크			
① 필수 : 그룹키(Array), MAC주소(Array), 포트홀순번(Array), 분류(발생, 접수, 보수완료) 또는 삭제 FLAG			
2) 관련테이블			
① POTHOLE_DETCT_INFO(포트홀탐지정보) : 수정			

바. 보수일지 관리

프로그램ID	P_WEB_REPAIRINFO	프로그램명	보수일지 관리
관련화면ID	L_WEB_REPAIRINFO	관련화면명	보수일지
개요	탐지된 포트홀에 대한 보수일지를 입력하고 관리한다.		
처리 로직			
1. 보수일지 목록 조회			
1) 처리로직			
① 사용자가 조회 할 조회시작일, 조회종료일을 입력 및 정렬기준, 1페이지 출력 데이터 개수, 데이터 구분을 선택하여 보수일지 데이터를 조회 요청한다.			
② 시스템은 사용자가 입력 및 선택한 값과 로그인 Session에 저장된 사용자의 ID를 인자 값으로 [USR_INFO], [POTHOLE_REPAIR_INFO] 테이블에서 사용자가 입력한 보수일지 데이터를 조회한다.			
③ 조회된 데이터를 화면상에 List로 출력한다. (출력필드 : 순번, 보수일, 보수자명, 주소, 사진, 기능)			
2) 파라미터 체크			
① 필수 : 조회시작일(YYYYMMDD), 조회종료일(YYYYMMDD)			
② 선택 : 정렬기준(시간, 상태, 크기), 1페이지 출력 데이터 개수(10, 30, 50, 100)			
2) 관련테이블			
① USR_INFO(사용자정보) : 조회			
② POTHOLE_REPAIR_INFO(포트홀보수정보) : 조회			
2. 보수일지 등록			
1) 처리로직			
① 사용자는 보수자, 주소, 종류, 규모, 비교, 보수사진을 입력하여 보수일지 등록을 요청한다.			
② 시스템은 사용자가 등록 요청한 값을 [POTHOLE_REPAIR_INFO] 테이블에 Insert하고 성공 여부를 사용자에게 리턴한다.			
2) 파라미터 체크			
① 필수 : 보수자, 주소(주소, 위도, 경도), 종류, 규모			
② 선택 : 보수사진(보수전, 보수중, 보수후 사진), 비교			
2) 관련테이블			
① POTHOLE_REPAIR_INFO(포트홀보수정보) : 입력			
3. 보수일지 수정			
1) 처리로직			
① 사용자는 보수자, 주소, 종류, 규모, 비교, 보수사진을 수정 입력하여 보수일지 수정을 요청한다.			
② 시스템은 사용자가 수정 요청한 값을 [POTHOLE_REPAIR_INFO] 테이블에 Update하고 성공 여부를 사용자에게 리턴한다.			
2) 파라미터 체크			
① 필수 : 보수자, 주소(주소, 위도, 경도), 종류, 규모			
② 선택 : 보수사진(보수전, 보수중, 보수후 사진), 비교			
2) 관련테이블			
① POTHOLE_REPAIR_INFO(포트홀보수정보) : 수정			
4. 보수일지 삭제			
1) 처리로직			
① 사용자는 삭제버튼을 클릭하여 삭제하고자 하는 보수일지 데이터에 대한 삭제를 요청한다.			
② 시스템은 사용자가 요청한 값을 인자 값으로 [POTHOLE_REPAIR_INFO] 테이블의 삭제여부(RMV_YN) 컬럼을 'Y'로 업데이트 후 성공 여부에 대한 메시지를 사용자에게 리턴한다.			
2) 파라미터 체크			
① 필수 : 보수정보 순번			
2) 관련테이블			
① POTHOLE_REPAIR_INFO(포트홀보수정보) : 수정			

사. 사용자 관리

프로그램ID	P_WEB_USR	프로그램명	사용자 관리
관련화면ID	L_WEB_USR	관련화면명	사용자 관리
개요	포트홀 관제 프로그램을 사용할수 있는 사용자 정보를 등록 및 관리한다.		
처리 로직			
1. 사용자 목록 조회			
1) 처리로직			
① 사용자가 시스템에 등록된 사용자 데이터를 조회 요청한다.			
② 시스템은 [USR_INFO] 테이블에서 등록된 사용자 데이터를 조회한다.			
③ 조회된 데이터를 화면상에 List로 출력한다. (출력필드 : 순번, ID, 이름, 닉네임, 최초등록일, 배정담당기, 기능)			
2) 파라메터 체크			
① 선택 : 검색구분(ID, 성명, 닉네임), 검색어			
2) 관련테이블			
① USR_INFO(사용자정보) : 조회			
2. 사용자 등록			
1) 처리로직			
① 사용자는 아이디, 비밀번호, 이름, 닉네임, 연락처, 레벨을 입력하여 사용자 등록을 요청한다.			
② 시스템은 사용자가 등록 요청한 값을 [USR_INFO] 테이블에 Insert하고 성공 여부를 사용자에게 리턴한다.			
2) 파라메터 체크			
① 필수 : 아이디, 비밀번호, 이름, 레벨			
② 선택 : 닉네임, 연락처, 초기좌표			
2) 관련테이블			
① USR_INFO(사용자정보) : 입력			
3. 사용자 수정			
1) 처리로직			
① 사용자는 이름, 닉네임, 연락처, 레벨을 수정 입력하여 사용자 정보 수정을 요청한다.			
② 시스템은 사용자가 수정 요청한 값을 [USR_INFO] 테이블에 Update하고 성공 여부를 사용자에게 리턴한다.			
2) 파라메터 체크			
① 필수 : 사용자코드, 이름 , 레벨			
② 선택 : 닉네임, 연락처, 초기좌표			
2) 관련테이블			
① USR_INFO(사용자정보) : 수정			
4. 사용자 삭제 처리			
1) 처리로직			
① 사용자는 삭제버튼을 클릭하여 삭제하고자 하는 사용자 정보 데이터에 대한 삭제 처리를 요청한다.			
② 시스템은 사용자가 요청한 값을 인자 값으로 [USR_INFO] 테이블의 데이터를 삭제여부 컬럼을 삭제('0')로 Update하고 성공 여부에 대한 메시지를 사용자에게 리턴한다.			
2) 파라메터 체크			
① 필수 : 사용자코드			
2) 관련테이블			
① USR INFO(사용자정보) : 수정			

아. 배정단말기 관리

프로그램ID	P_WEB_MAC	프로그램명	배정단말기 관리
관련화면ID	L_WEB_MAC	관련화면명	배정단말기 관리
개요	포트홀 관제프로그램 사용자에게 단말기를 배정하고 관리한다.		
처리 로직			
1. 배정단말기 목록 조회			
1) 처리로직			
① 사용자가 선택한 사용자의 배정단말기 데이터를 조회 요청한다.			
② 시스템은 [USR_UNIT_SET_STS] 테이블에서 등록된 배정단말기 데이터를 조회한다.			
③ 조회된 데이터를 화면상에 List로 출력한다. (출력필드 : MAC주소, 단말기명, 관리자, 전화번호 등록일, 기능)			
2) 파라미터 체크			
① 필수 : 사용자코드			
2) 관련테이블			
① USR_UNIT_SET_STS(사용자단말기배정상태) : 조회			
2. 배정단말기 등록			
1) 처리로직			
① 사용자는 MAC주소, 단말기명, 관리자, 전화번호를 입력하여 배정단말기 등록을 요청한다.			
② 시스템은 사용자가 등록 요청한 값을 [USR_UNIT_SET_STS] 테이블에 Insert하고 [USR_UNIT_SET_STS] 테이블에 배정 이력정보를 Insert한 후 성공 여부를 사용자에게 리턴한다.			
2) 파라미터 체크			
① 필수 : 사용자코드, MAC주소, 단말기명			
② 선택: 관리자, 전화번호			
2) 관련테이블			
① USR_UNIT_SET_STS(사용자단말기배정상태) : 입력			
② USR_UNIT_SET_HST(사용자단말기배정이력) : 입력			
3. 배정단말기 수정			
1) 처리로직			
① 사용자는 MAC주소를 수정 입력하여 배정단말기 수정을 요청한다.			
② 시스템은 사용자가 수정 요청한 값을 [USR_UNIT_SET_STS] 테이블에 Update하고 [USR_UNIT_SET_STS] 테이블에 배정 이력정보를 Insert한 후 성공 여부를 사용자에게 리턴한다.			
2) 파라미터 체크			
① 필수 : 사용자코드, MAC주소, 단말기명			
② 선택: 관리자, 전화번호			
2) 관련테이블			
① USR_UNIT_SET_STS(사용자단말기배정상태) : 수정			
② USR_UNIT_SET_HST(사용자단말기배정이력) : 입력			
4. 배정단말기 삭제			
1) 처리로직			
① 사용자는 삭제버튼을 클릭하여 삭제하고자 하는 배정단말기 데이터에 대한 삭제를 요청한다.			
② 시스템은 사용자가 요청한 값을 인자 값으로 [USR_UNIT_SET_STS] 테이블의 데이터를 Delete하고 [USR_UNIT_SET_STS] 테이블에 배정 이력정보를 Insert한 후 성공 여부에 대한 메시지를 사용자에게 리턴한다.			
2) 파라미터 체크			
① 필수 : 사용자코드, MAC주소			
2) 관련테이블			
① USR_UNIT_SET_STS(사용자단말기배정상태) : 삭제			
② USR UNIT SET HST(사용자단말기배정이력) : 입력			

자. 위험구간분석

프로그램ID	P_WEB_ROADANALYSIS	프로그램명	위험구간분석								
관련화면ID	L_WEB_ROADANALYSIS	관련화면명	위험구간								
개요	도로구간의 포트홀에 따른 위험도를 분석하고 위험등급 별 도로구간 공간 데이터를 지도상에 표출한다.										
처리 로직											
1. 위험등급 계산											
1) 위험등급 매트릭스											
○ 중심극한정리(Central Limit Theorem)이론을 바탕으로 포트홀 개수와 포트홀 크기에 대한 등급화를 수행.											
○ 포트홀 개수와 포트홀 크기를 발생가능성(Likelihood)과 영향(Impact)으로 구분											
○ 1km를 단위구간으로 하는 경우의 위험도 매트릭스(Risk Matrix)를 아래 표와 같이 작성, DATABASE에 저장하여 관리											
Matrix명		기본가중치									
구분		포트홀 개수 (개/km)									
		1.6	이하	1.6	~1.9	1.9	~2.5	2.5	~2.8	2.8	이상
평균포트홀 면적 (1,000Pixels/개)	2.4	이하	매우양호	매우양호	양호	양호	양호	보통			
	2.4	~2.6	매우양호	양호	양호	양호	보통	위험			
	2.6	~3.1	양호	양호	보통	위험	위험	위험			
	3.1	~3.3	양호	보통	위험	위험	위험	매우위험			
	3.3	이상	보통	위험	위험	매우위험	매우위험	매우위험			
2. 도로구간 위험등급별 조회											
1) 처리로직											
① 사용자는 조회기간, 시도, 시군구, 도로종류를 선택하여 위험등급별 도로구간 건수 데이터 및 도로구간 Polyline 공간 데이터를 요청한다.											
② 시스템은 사용자가 요청한 조회기간,시도, 시군구, 도로종류를 인자 값으로 [MOCT_LINK], [MOCT_NODE] 테이블에서 입력된 조건에 해당하는 위험등급별 도로구간 건수 데이터 및 도로구간 Polyline 공간정보를 조회한다.											
③ 조회된 위험등급별 도로구간 건수 데이터를 화면에 출력하고 도로구간 Polyline 공간정보를 위험등급별 색상으로 구분하여 지도상에 PolyLine으로 Draw한다.											
2) 파라미터 체크											
① 필수 : 시도코드, 시군구코드, 도로유형코드, 도로명											
2) 관련테이블											
① MOCT_LINK(표준링크) : 조회											
② MOCT_NODE(표준노드) : 조회											
6. 도로구간 상세 정보 조회											
1) 처리로직											
① 사용자는 5.에서 출력된 지도상의 도로구간 PolyLine을 클릭하여 클릭한 도로구간에 대한 상세정보를 요청한다.											
② 시스템은 요청한 값을 인자 값으로 [MOCT_LINK], [MOCT_NODE] 테이블에서 해당 도로구간에 대한 상세정보를 조회한다.											
③ 조회된 도로구간에 대한 상세정보를 팝업화면으로 출력한다. (출력항목 : 링크ID, 기점노드, 종점노드, 도로길이, 도로명, 도로등급, 차로수, 위험도, 주행횟수, 평균포트홀갯수, 1Km당 포트홀 개수, 포트홀 평균면적)											
2) 파라미터 체크											
① 필수 : 링크식별자(도로구간식별자)											
2) 관련테이블											
① MOCT_LINK(표준링크) : 조회											
② MOCT_NODE(표준노드) : 조회											
① pothole(포트홀) : 조회											

4.3. 인터페이스 구성

가. 전송 항목

◦ 포트홀 수집 전용 단말기와 수집프로그램 간의 전송 항목 구성은 아래와 같다.

항목	설명	예시	비고
맥어드레스	전송 단말기의 맥어드레스 값	50-77-05-3F-8F-8F	
일자	탐지 날짜	20180905	
시분초밀리	탐지 시분초밀리	092314337	
위도	탐지 위도	35.846710	
경도	탐지 경도	128.515521	
고도	탐지 고도	81.100	단위: m
이동방향	탐지시 이동방향	0	단위: 도
속도	탐지시 이동속도	0.000	단위: km/h
GPS상태코드	GPS 신호상태	1	0 : 신호불량 1 : 신호양호
포트홀 x좌표	탐지 이미지상의 포트홀 위치 x좌표	249	단위: 픽셀
포트홀 y좌표	탐지 이미지상의 포트홀 위치 y좌표	135	단위: 픽셀
포트홀 width	탐지 이미지상의 포트홀 width	94	단위: 픽셀
포트홀 height	탐지 이미지상의 포트홀 height	26	단위: 픽셀
이미지파일명	탐지 이미지 파일명	50-77-05-3F-8F-8F\20180905\092314337.jpg	
주소	탐지 위도 및 경도에 해당하는 주소	대구광역시 달서구 장동 374-27	
원본이미지 파일명	탐지된 이미지에 대한 원본이미지 파일명	50-77-05-3F-8F-8F\20180905\orig\092314337.jpg	
포트홀 면적	탐지된 포트홀에 대한 픽셀 단위 실면적	400	단위: pixel ²
G센서데이터 수	G센서 데이터 수 (최대: 100)	100	0~100
G센서데이터	7~8자리로 실수값으로 표시된 데이터 필드, 항목별 쉼표(",")가 삽입됨. 마지막 항목 쉼표 미삽입. 예시) 0.12345 (7자리), 12.34567 (8자리)	0.12345, 12.34567, 12.34567, 12.34567, 56.78901	0~100항목 (최대999자)
단말기종	스마트폰, TX2	0	0 : 스마트폰 1 : TX2
소프트웨어 버전 정보	탐지 소프트웨어 버전	1.01	
포트홀 정보	탐지된 포트홀 정보들	x,y,w,h,area^249,135,94,24,400^...	

나. 전송 방법

- 전송방법은 아래의 예시와 같이 전송항목에 대해 구분자 '#' 으로 구분하여 전송하고 서버측의 stored procedure에서 전송된 값을 splitting하여 DB에 insert하는 형태로 전송한다.

예시) - 전송파일명: 50-77-05-3F-8F-8F_20180905092314337_1.txt

- 전송파일내용:

```
50-77-05-3F-8F-8F#20180905#092314337#35.846710#128.515521#81.100#0#0.000#1#249#135#94#26#50-77-05-3F-8F-8FW20180905#092314337.jpg#대구광역시 달서구 장동 374-27#50-77-05-3F-8F-8FW20180905#orig#092314337.jpg#400#100#12.34567,89.12345,00000002,00000003,.....생략.....,99.12345#0#1.02#
```

4.4. 소스코드

가. 운행현황

```
package kr.co.kworks.pothole.controller;

import java.util.HashMap;

@Controller
public class DrvStatController {
    @Resource(name = "drvStatService")
    private DrvStatService drvStatService;

    @Resource(name = "recService")
    private RecService recService;

    @Resource(name = "systemService")
    private SystemService systemService;

    //운행현황 (미인라인)
    @RequestMapping(value = "/drvstat/drvStat.do")
    public String drvStat(Model model, HttpServletRequest req, HttpServletResponse res) throws Exception {
        //로그인유저 정보 확인
        UserInfoVO loginVO = (UserInfoVO) req.getSession().getAttribute("loginVO");

        //사용자 변경 인입기 조회
        UserUnitSetStsVO userUnitSetStsVO = new UserUnitSetStsVO();
        userUnitSetStsVO.setUserCd(loginVO.getUserCd());
        List<KworkMap> macList = systemService.selectUserUnitSetStsList(userUnitSetStsVO);

        model.addAttribute("macList", macList);

        return "drvstat/drvStat";
    }

    //포토를 찾기 목록 조회 - 운영
    @ResponseBody
    @RequestMapping(value = "/drvstat/potholeDrvListAjax.do")
    public HashMap<String, Object> selfPotholeData(Model model, MockLinkVO searchMockLinkVO, PotholeDetctInfoVO searchPotholeDetctInfoVO, HttpServletRequest req, HttpServletResponse res) throws Exception {
        HashMap<String, Object> hashMap = new HashMap<String, Object>();
        System.out.println(searchPotholeDetctInfoVO.getSearchCategory());
        //로그인유저 정보 확인
        UserInfoVO loginVO = (UserInfoVO) req.getSession().getAttribute("loginVO");

        //포토를 찾기정보 목록
        searchPotholeDetctInfoVO.setSearchUserCd(loginVO.getUserCd());
        searchPotholeDetctInfoVO.setStartDt(searchPotholeDetctInfoVO.getStartDt().replaceAll("-", ""));
        searchPotholeDetctInfoVO.setEndDt(searchPotholeDetctInfoVO.getEndDt().replaceAll("-", ""));
        searchPotholeDetctInfoVO.setSearchArray(searchPotholeDetctInfoVO.getSearchCategory().split(","));
        List<KworkMap> potholeInfoList = drvStatService.selectPotholeDetctInfoTimeList(searchPotholeDetctInfoVO);

        //운행구간
        searchMockLinkVO.setSearchUserCd(loginVO.getUserCd());
        searchMockLinkVO.setStartDt(searchPotholeDetctInfoVO.getStartDt().replaceAll("-", ""));
        searchMockLinkVO.setEndDt(searchPotholeDetctInfoVO.getEndDt().replaceAll("-", ""));
        searchMockLinkVO.setSearchArray(searchPotholeDetctInfoVO.getSearchCategory().split(","));
        List<KworkMap> mockLinkList = drvStatService.selectMockLinkTimeList(searchMockLinkVO);

        hashMap.put("potholeInfoList", potholeInfoList);
        hashMap.put("mockLinkList", mockLinkList);

        return hashMap;
    }

    //포토를 상세 + 세부 목록조회
    @RequestMapping(value = "/drvstat/potholeMarkerInfo.do")
    public String potholeMarkerInfo(Model model, PotholeDetctInfoVO searchPotholeDetctInfoVO, PotholeDetctDetailVO searchPotholeDetctDetailVO, HttpServletRequest req, HttpServletResponse res) throws Exception {
        List<KworkMap> potholeDetailList = recService.selectPotholeDetctDetailList(searchPotholeDetctDetailVO);

        model.addAttribute("potholeInfo", potholeInfo);
        model.addAttribute("potholeDetailList", potholeDetailList);
        return "drvstat/potholeMarkerInfo";
    }
}
```

나. 탐지기록

```
package kr.co.kworks.pothole.controller;

import java.util.ArrayList;

@Controller
public class RecController {

    @Resource(name = "recService")
    private RecService recService;

    @Resource(name = "systemService")
    private SystemService systemService;

    //포토홀 목록 화면
    @RequestMapping(value="/rec/potholeList.do")
    public String potholeList(Model model, HttpServletRequest req, HttpServletResponse res) throws Exception{
        //로그인유저 정보 확인
        UserInfoVO loginVO = (UserInfoVO)req.getSession().getAttribute("loginVO");

        //사용자 버전 전달기 조회
        UserUnitSetStsVO userUnitSetStsVO = new UserUnitSetStsVO();
        userUnitSetStsVO.setUser_cd(loginVO.getUser_cd());
        List<KworkMap> macList = systemService.selectUserUnitSetStsList(userUnitSetStsVO);

        //포토홀 상하코드 목록
        ComCdVO searchComCdVO = new ComCdVO();
        searchComCdVO.setCom_cls_cd("A000001");
        searchComCdVO.setUse_yn("Y");
        List<KworkMap> comCdList = systemService.selectComCdList(searchComCdVO);

        model.addAttribute("macList", macList);
        model.addAttribute("comCdList", comCdList);
        return "rec/potholeList";
    }

    //포토홀 목록조회
    @ResponseBody
    @RequestMapping(value="/rec/potholeListAjax.do")
    public HashMap<String, Object> potholeListAjax(Model model, PotholeDetctInfoVO searchPotholeDetctInfoVO, HttpServletRequest req, HttpServletResponse res) throws Exception{
        //로그인유저 정보 확인
        UserInfoVO loginVO = (UserInfoVO)req.getSession().getAttribute("loginVO");

        searchPotholeDetctInfoVO.setSearchUserCd(loginVO.getUser_cd());
        searchPotholeDetctInfoVO.setStartDt(searchPotholeDetctInfoVO.getStartDt().replaceAll("-", ""));
        searchPotholeDetctInfoVO.setEndDt(searchPotholeDetctInfoVO.getEndDt().replaceAll("-", ""));
        int totCnt = recService.selectPotholeDetctInfoListCnt(searchPotholeDetctInfoVO);

        PaginationInfo paginationInfo = new PaginationInfo();
        paginationInfo.setTotalRecordCount(totCnt);
        PaginationToHtml paginationToHtml = new PaginationToHtml(paginationInfo, searchPotholeDetctInfoVO);
        List<KworkMap> potholeInfoList = recService.selectPotholeDetctInfoList(searchPotholeDetctInfoVO);

        HashMap<String, Object> hashMap = new HashMap<String, Object>();
        hashMap.put("potholeInfoList", potholeInfoList);
        hashMap.put("paginationInfo", paginationInfo);
        hashMap.put("paginationToHtml", paginationToHtml.getPaginationHTML());

        return hashMap;
    }

    //포토홀 삭제처리
    @ResponseBody
    @RequestMapping(value="/rec/potholeInfoRemoveAjax.do", method = RequestMethod.POST)
    public HashMap<String, Object> potholeInfoRemoveAjax(Model model, PotholeDetctInfoVO potholeDetctInfoVO, HttpServletRequest req, HttpServletResponse res) throws Exception{
        HashMap<String, Object> retMap = new HashMap<String, Object>();

        int ret = recService.updatePotholeDetctInfoRmv(potholeDetctInfoVO);
        if(ret > 0){
            retMap.put("rtnCode", "1");
            retMap.put("rtnMsg", "삭제되었습니다.");
        }else{
            retMap.put("rtnCode", "0");
            retMap.put("rtnMsg", "삭제에 실패하였습니다.");
        }
        return retMap;
    }

    //포토홀 상태변경
    @ResponseBody
    @RequestMapping(value="/rec/potholeInfoChangeStsAjax.do", method = RequestMethod.POST)
    public HashMap<String, Object> potholeInfoChangeStsAjax(Model model, PotholeDetctInfoVO potholeDetctInfoVO, HttpServletRequest req, HttpServletResponse res) throws Exception{
        HashMap<String, Object> retMap = new HashMap<String, Object>();

        int ret = recService.updatePotholeDetctInfoSts(potholeDetctInfoVO);
        if(ret > 0){
            retMap.put("rtnCode", "1");
            retMap.put("rtnMsg", "변경되었습니다.");
        }else{
            retMap.put("rtnCode", "0");
            retMap.put("rtnMsg", "변경에 실패하였습니다.");
        }
        return retMap;
    }
}
```

다. 보수기록

```
package kr.co.kworks.pothole.controller;

import java.util.HashMap;

@Controller
public class RepairController {
    @Resource(name = "repairService")
    private RepairService repairService;

    // 보수기록 목록 화면
    @RequestMapping(value="/repair/repairList.do")
    public String repairList(Model model, HttpServletRequest req, HttpServletResponse res) throws Exception{
        return "repair/repairList";
    }

    // 보수기록 목록
    @ResponseBody
    @RequestMapping(value = "/repair/repairListAjax.do")
    public HashMap<String, Object> repairListAjax(PotholeRepairInfoVO searchPotholeRepairInfoVO, HttpServletRequest req, HttpServletResponse res) throws
        //로그인유저 정보 확인
        UserInfoVO loginVO = (UserInfoVO)req.getSession().getAttribute("loginVO");

        searchPotholeRepairInfoVO.setSearchUsrCd(loginVO.getUsr_cd());
        searchPotholeRepairInfoVO.setStartDt(searchPotholeRepairInfoVO.getStartDt().replaceAll("-", ""));
        searchPotholeRepairInfoVO.setEndDt(searchPotholeRepairInfoVO.getEndDt().replaceAll("-", ""));

        int totCnt = repairService.selectPotholeRepairInfoListCnt(searchPotholeRepairInfoVO);

        PaginationInfo paginationInfo = new PaginationInfo();
        paginationInfo.setTotalRecordCount(totCnt);
        PaginationToHtml paginationToHtml = new PaginationToHtml(paginationInfo, searchPotholeRepairInfoVO);
        List<KworkMap> repairList = repairService.selectPotholeRepairInfoList(searchPotholeRepairInfoVO);

        HashMap<String, Object> hashMap = new HashMap<String, Object>();
        hashMap.put("repairList", repairList);
        hashMap.put("paginationInfo", paginationInfo);
        hashMap.put("paginationToHtml", paginationToHtml.getPaginationHTML());

        return hashMap;
    }

    // 보수기록 등록 화면
    @RequestMapping(value="/repair/repairRegeditPop.do")
    public String repairRegeditPop(Model model, HttpServletRequest req, HttpServletResponse res) throws Exception{
        return "repair/repairRegeditPop";
    }

    // 보수기록 수정 화면
    @RequestMapping(value="/repair/repairModifyPop.do")
    public String repairModifyPop(Model model, PotholeRepairInfoVO searchPotholeRepairInfoVO, HttpServletRequest req, HttpServletResponse res) throws Ex
        KworkMap repair = repairService.selectPotholeRepairInfoDetail(searchPotholeRepairInfoVO);
        model.addAttribute("repair", repair);
        return "repair/repairModifyPop";
    }

    // 보수기록 등록
    @ResponseBody
    @RequestMapping(value="/repair/repairInsertAjax.do", method = RequestMethod.POST)
    public HashMap<String, Object> repairInsertAjax(MultipartHttpServletRequest req, PotholeRepairInfoVO potholeRepairInfoVO) throws Exception{
        HashMap<String, Object> retMap = new HashMap<String, Object>();
        //로그인유저 정보 확인
        UserInfoVO loginVO = (UserInfoVO)req.getSession().getAttribute("loginVO");

        potholeRepairInfoVO.setUsr_cd(loginVO.getUsr_cd());
        potholeRepairInfoVO.setRepair_de(potholeRepairInfoVO.getRepair_de().replaceAll("-", ""));

        Map<String, MultipartFile> files = req.getFileMap();

        int ret = repairService.insertPotholeRepairInfoDetail(potholeRepairInfoVO, files);
        if(ret > 0){
            retMap.put("rtnCode", "1");
            retMap.put("rtnMsg", "등록되었습니다.");
        }else{
            retMap.put("rtnCode", "0");
            retMap.put("rtnMsg", "등록에 실패하였습니다.");
        }

        return retMap;
    }

    // 보수기록 수정
    @ResponseBody
    @RequestMapping(value="/repair/repairUpdateAjax.do", method = RequestMethod.POST)
    public HashMap<String, Object> repairUpdateAjax(MultipartHttpServletRequest request, PotholeRepairInfoVO potholeRepairInfoVO) throws Exception{
        HashMap<String, Object> retMap = new HashMap<String, Object>();
        Map<String, MultipartFile> files = request.getFileMap();

        potholeRepairInfoVO.setRepair_de(potholeRepairInfoVO.getRepair_de().replaceAll("-", ""));
        int ret = repairService.updatePotholeRepairInfoDetail(potholeRepairInfoVO, files);
    }
```

라. 위험구간

```
package kr.co.kworks.pothole.controller;

import java.util.HashMap;

@Controller
public class LinkController {

    //링크 Service
    @Resource(name = "linkService")
    private LinkService linkService;

    @Resource(name = "systemService")
    private SystemService systemService;

    //지도기반 위험도 분석 화면
    @RequestMapping(value = "/link/roadAnalysis.do")
    public String roadAnalysis(Model model, HttpServletRequest req, HttpServletResponse res) throws Exception {
        //도로영역 목록
        ComCdVO searchComCdVO = new ComCdVO();
        searchComCdVO.setCom_cls_cd("A0000002");
        searchComCdVO.setUse_yn("Y");
        List<KworksMap> comCdList = systemService.selectComCdList(searchComCdVO);

        //시도 목록
        List<KworksMap> sidoList = linkService.selectMootAreaSidoList();

        model.addAttribute("comCdList", comCdList);
        model.addAttribute("sidoList", sidoList);

        return "link/roadAnalysis";
    }

    //시군구 목록
    @ResponseBody
    @RequestMapping(value = "/link/sigunguListAjax.do", method = RequestMethod.POST)
    public List<KworksMap> sigunguListAjax(MootAreaVO mootAreaVO, HttpServletRequest req, HttpServletResponse res) throws Exception {
        List<KworksMap> sigunguList = linkService.selectMootAreaSigunguList(mootAreaVO);

        return sigunguList;
    }

    //도로구간-위험도 목록
    @ResponseBody
    @RequestMapping(value = "/link/mootLinkRiskListAjax.do", method = RequestMethod.POST)
    public HashMap<String, Object> mootLinkRiskListAjax(MootLinkVO searchMootLinkVO, HttpServletRequest req, HttpServletResponse res) throws Exception {
        HashMap<String, Object> retMap = new HashMap<String, Object>();

        //구간 목록
        searchMootLinkVO.setStartDt(searchMootLinkVO.getStartDt().replaceAll("-", ""));
        searchMootLinkVO.setEndDt(searchMootLinkVO.getEndDt().replaceAll("-", ""));
        List<KworksMap> mootLinkList = linkService.selectMootLinkRiskList(searchMootLinkVO);

        retMap.put("mootLinkList", mootLinkList);
        retMap.put("searchMootLinkVO", searchMootLinkVO);

        return retMap;
    }

    //도로구간-위험도 상세
    @ResponseBody
    @RequestMapping(value = "/link/mootLinkRiskDetailAjax.do", method = RequestMethod.POST)
    public HashMap<String, Object> mootLinkRiskDetailAjax(MootLinkVO searchMootLinkVO, HttpServletRequest req, HttpServletResponse res) throws Exception {
        HashMap<String, Object> retMap = new HashMap<String, Object>();

        //위험도 하위 목록
        RiskLevelVO riskLevelVO = new RiskLevelVO();
        List<KworksMap> riskLevelList = linkService.selectRiskLevelList(riskLevelVO);

        //구간상세
        searchMootLinkVO.setStartDt(searchMootLinkVO.getStartDt().replaceAll("-", ""));
        searchMootLinkVO.setEndDt(searchMootLinkVO.getEndDt().replaceAll("-", ""));
        KworksMap mootLink = linkService.selectMootLinkRiskDetail(searchMootLinkVO);

        retMap.put("mootLink", mootLink);
        retMap.put("riskLevelList", riskLevelList);

        return retMap;
    }

    //도로구간-위험도 메트릭스 수정 목록
    @ResponseBody
    @RequestMapping(value = "/link/linkRiskSnChangeAjax.do", method = RequestMethod.POST)
    public HashMap<String, Object> linkRiskSnChangeAjax(MootLinkRiskSnVO mootLinkRiskSnVO, MootLinkVO searchMootLinkVO, HttpServletRequest req, HttpServ
        int ret = linkService.updateMootLinkRiskSn(mootLinkRiskSnVO);
        HashMap<String, Object> retMap = new HashMap<String, Object>();
        retMap.put("ret", ret);
        if (ret > 0) {
            //위험도 하위 목록
            RiskLevelVO riskLevelVO = new RiskLevelVO();
            List<KworksMap> riskLevelList = linkService.selectRiskLevelList(riskLevelVO);

            KworksMap mootLink = linkService.selectMootLinkRiskDetail(searchMootLinkVO);
            retMap.put("mootLink", mootLink);
        }
    }
}
```

마. 관리

```
package kr.co.kworks.pothole.controller;

import java.util.HashMap;

import org.springframework.web.servlet.mvc.annotation.AnnotationMethodMapping;
import org.springframework.web.servlet.mvc.annotation.AnnotationMethodMapping;
import org.springframework.web.servlet.mvc.annotation.AnnotationMethodMapping;

@Controller
public class SystemController {
    @Resource(name = "systemService")
    private SystemService systemService;

    // 사용자 목록 조회
    @RequestMapping(value="/system/usrList.do")
    public String unitList(Model model, HttpServletRequest req, HttpServletResponse res) throws Exception{
        return "system/usrList";
    }

    // 사용자 목록
    @ResponseBody
    @RequestMapping(value = "/system/usrListAjax.do")
    public HashMap<String, Object> usrListAjax(UserInfoVO searchUserInfoVO, HttpServletRequest req, HttpServletResponse res) throws Exception {
        int totCnt = systemService.selectUserInfoListCnt(searchUserInfoVO);

        PaginationInfo paginationInfo = new PaginationInfo();
        paginationInfo.setTotalRecordCount(totCnt);
        PaginationToHtml paginationToHtml = new PaginationToHtml(paginationInfo, searchUserInfoVO);
        List<KworksMap> usrList = systemService.selectUserInfoList(searchUserInfoVO);

        HashMap<String, Object> hashMap = new HashMap<String, Object>();
        hashMap.put("usrList", usrList);
        hashMap.put("paginationInfo", paginationInfo);
        hashMap.put("paginationToHtml", paginationToHtml.getPaginationHTML());

        return hashMap;
    }

    // 사용자 등록 화면
    @RequestMapping(value="/system/usrRegeditPop.do")
    public String usrRegeditPop(Model model, HttpServletRequest req, HttpServletResponse res) throws Exception{
        return "system/usrRegeditPop";
    }

    // 사용자 수정 화면
    @RequestMapping(value="/system/usrModifyPop.do")
    public String usrModifyPop(Model model, UserInfoVO searchUserInfoVO, HttpServletRequest req, HttpServletResponse res) throws Exception{
        KworksMap usr = systemService.selectUserInfoDetail(searchUserInfoVO);
        model.addAttribute("usr", usr);
        return "system/usrModifyPop";
    }

    // 사용자 등록
    @ResponseBody
    @RequestMapping(value="/system/usrInsertAjax.do", method = RequestMethod.POST)
    public HashMap<String, Object> usrInsertAjax(ModelMap model, UserInfoVO userInfoVO) throws Exception{
        HashMap<String, Object> retMap = new HashMap<String, Object>();

        int usrCnt = systemService.selectUserInfoIdCnt(userInfoVO);
        if(usrCnt > 0){
            retMap.put("rtnCode", "0");
            retMap.put("rtnMsg", userInfoVO.getUser_id() + "은 이미 사용중이거나 탈퇴한 아이디입니다.");
            return retMap;
        }
        String salt = EgovFileSorty.generateSalt();
        String enpassword = EgovFileSorty.encryptPassword(userInfoVO.getUser_pw(), salt);
        userInfoVO.setUser_pw(enpassword);
        userInfoVO.setUser_pw_salt(salt);
        int ret = systemService.insertUserInfoDetail(userInfoVO);
        if(ret > 0){
            retMap.put("rtnCode", "1");
            retMap.put("rtnMsg", "등록되었습니다.");
        }else{
            retMap.put("rtnCode", "0");
            retMap.put("rtnMsg", "등록에 실패하였습니다.");
        }

        return retMap;
    }

    // 사용자 수정
    @ResponseBody
    @RequestMapping(value="/system/usrUpdateAjax.do", method = RequestMethod.POST)
    public HashMap<String, Object> usrUpdateAjax(ModelMap model, UserInfoVO userInfoVO, HttpServletRequest req, HttpServletResponse res) throws Exception{
        HashMap<String, Object> retMap = new HashMap<String, Object>();

        int ret = systemService.updateUserInfoDetail(userInfoVO);
        if(ret > 0){
            retMap.put("rtnCode", "1");
            retMap.put("rtnMsg", "수정되었습니다.");
        }else{
            retMap.put("rtnCode", "0");
            retMap.put("rtnMsg", "수정에 실패하였습니다.");
        }

        return retMap;
    }
}
```

4.5. 프로그램 알고리즘

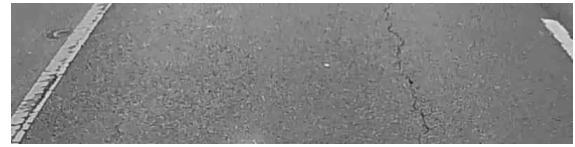
1.1 Rule base Algorithm

○ RGB2Y

- RGB(Red, Green, Blue) 기반의 3채널 컬러 영상을 Y(흑백) 단일 채널 영상으로 변환하는 알고리즘 $Y = 0.299 * R + 0.587 * G + 0.114 * B$
- RA 또는 DLA 연산 시 메모리 사용량을 줄이기 위하여 사용 검토
- 알고리즘 프로세싱 속도 향상에 따른 성능 저하 여부 테스트 하고, 알고리즘에 포함 여부 검토함



RGB(컬러) 영상

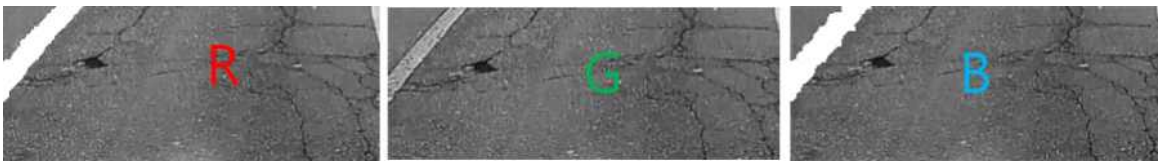


Y(흑백) 영상

- RA에서는 활용하고, DLA는 RGB 사용시의 속도 테스트

○ RGB-Y

- RGB(Red, Green, Blue) 기반 3채널 컬러 영상의 각 채널에서 Y(흑백) 단일 채널 영상을 빼는 알고리즘 검토
- RA의 RGB 기반 영상에서 컬러를 포함하는 비포트홀 영역 위치 돌출 후 제거하기 위하여 사용
- RA 탐지 성능 향상을 위한 테스트, 각 채널 별 Y 채널을 빼 RGB 중 비포트홀 영역이 가장 많이 제거되는 영역 판단, 적용 가능성있지만 활용성이 높지 않다고 보임



RGB-Y 후 Y에서 돌출 영역 제거, 왼쪽에서부터 R-Y, G-Y, B-Y

○ Threshold (이진화)

- Y(흑백) 단일 채널 영상의 값(0 ~ 255)에서 임의의 값 기반으로 작으면 최소 값(0), 크면 최대 값(255)로 0과 255로만 표현하여 배경 및 물체 나타내는 알고리즘, 영상 프레임 별 자동 임의의 값 설정하는 동적 및 전프레임 고정 값인 정적 알고리즘 존재
- 특징점 기반 알고리즘의 포트홀 영역 탐지하기 위하여 사용
- 특징점 기반 탐지 알고리즘으로 사용하기 위해 중간 값(125) 이용 탐지 정확도 및 적용 가능성 여부 판단, 동적 알고리즘 적용 가능성 확인

- OTSU (오츠 알고리즘)
 - Adaptive Threshold (동적 이진화) 알고리즘 중 하나, 동적 이진화에 사용되는 임의의 값 설정하여 배경 및 물체 나타내는 알고리즘
 - 특징점 기반 알고리즘의 포트홀 탐지하기 위하여 사용
 - 특징점 기반 탐지 알고리즘으로 사용하기 위해 포트홀 영역 탐지 정확도 및 적용 가능성 여부 판단
- Super Pixel (수퍼 픽셀) - SLIC (Simple Linear Iterative Clustering)
 - 유사 픽셀 클러스터링(군집화), 셀로 표현하는 알고리즘
 - 특징점 기반 알고리즘의 포트홀 탐지하기 위하여 사용
 - 특징점 기반 알고리즘의 포트홀 영역 탐지에 사용하기 위해 옵션 (셀 크기, 룰러, 연결성, 반복횟수) 별 테스트 하고, 본 연구에 적용 가능성을 확인
- 배경제거 (BackgroundSubtractor)
 - 영상 내부의 배경과 전경 분리 알고리즘. 기반하는 알고리즘에 따라 GMG, MOG, MOG2, KNN으로 구분됨
 - 포트홀 영역 탐지를 위해 사용
 - 도로 노면이 아닌 특징들을 드러내기 위해 사용됨. 화이트 포트홀, 패치, 크랙, 포트홀, 차선 등이 드러남
- Convex Hull (컨벡스 헐)
 - 경계선의 꼭지점을 이어 최소 크기의 다각형 그리는 알고리즘
 - 수퍼 픽셀 결과 셀 경계선의 다각형 검출
 - 특징점 기반 알고리즘의 수퍼 픽셀 셀 경계선 특징 이용 실험, 도로 노면과 같이 유사 색상 및 넓은 이미지의 경우 사각형 경계선 돌출, 포트홀과 같이 원형 또는 타원형의 형태의 경우 사각형이 아닌 다각형 검출 확인
- Morphology (모폴로지)
 - 영상의 화질 개선 및 특징 돌출 위해 사용, 영역을 넓히는 팽창, 영역을 좁히는 침식, 침식 후 팽창하여 경계선을 잇는 열기 그리고 팽창 후 침식으로 구멍을 제거할 수 있는 닫기, 4가지 연산 알고리즘
 - 특징점 기반 알고리즘에서 흰색의 잡음 제거 및 포트홀 영역 정확도 향상
 - 특징점 기반 알고리즘의 정확도 향상을 위한 침식, 팽창, 열기 그리고 닫기 연산 테스트, 적용 가능성을 확인
- Saliency map (세일리언시 맵)

- 영상 내부의 특징되는 물체만 드러내는 알고리즘
- 포트홀 영역 탐지 위하여 사용
- 영상중 시각적으로 돌출되는 객체를 추출하는 알고리즘으로, 정상적인 노면에서 포트홀과 같이 배경색과 다르거나, 시각적으로 띄는 객체를 추출함
- 연산속도도 빨라 RA에 활용할 수 있음
- WEF (웨이블릿 에너지 필드)
 - 이미지 압축 방식인 웨이블릿 압축 기반 이미지 질감 나타내는 알고리즘
 - 포트홀 후보 영역 질감 판단하여 포트홀 탐지하기 위하여 사용
 - 타 연구에서 높은 정확도 기반하여 적용하였으나, 일반적인 환경에서는 높지 않은 정확도를 보였고, 연산속도가 느려서 실시간용으로 활용성이 낮다고 보임

1.2 Deep learning Algorithm

- 딥러닝 프레임워크
 - 딥러닝을 위한 하드웨어 및 소프트웨어 집합체. 개발에 있어 하드웨어 및 라이브러리를 제공하여 딥러닝 개발에 집중할 수 있도록 함. 대표적인 딥러닝 프레임워크로 Google TensorFlow, Theano, Caffe, Torch 등이 있으며, 본 기술에는 텐서플로우 프레임워크를 주로 사용하기로 했음
 - 포트홀 탐지를 위한 딥러닝 프레임워크 선정을 위한 분석
 - Google TensorFlow가 가장 많은 사용자 층을 보유
 - 기계학습 프레임워크로서 Google TensorFlow를 검토
- Google TensorFlow
 - Google에서 개발한 딥러닝 프레임워크. C언어 기반으로 작성. 하드웨어 자원 제어 및 딥러닝 모델 작성을 위한 라이브러리를 지원
 - 포트홀 탐지용 딥러닝 모델 생성을 위한 기반 프레임워크로서 사용
 - 가장 많은 딥러닝 프레임워크 사용자층 보유. 많은 피드백 및 사용 예제가 많음. Tensor board 도구를 통하여 모델의 학습 상태, 오류 상황, 모델 형태 분석 등을 시각적으로 제공. 본 연구에서 딥러닝에 집중할 수 있는 도구로서 가능성을 확인
 - 본 연구의 딥러닝 프레임워크 모델로서 사용
- TensorFlow Slim
 - Google에서 제공하는 딥러닝 학습 모델 라이브러리 모음. TensorFlow위에서 동작할 수 있도록 설계됨. ImageNet Large Scale Visual Recognition Competition (ILSVRC)에서 우

- 승한 모델인 Inception, ResNet, Inception_ResNet, MobileNet 등 다양한 모델을 제공하고 있어, 제공하는 모델 중에서 적합한 것을 탐색하여 활용하기로 함
- 영상 내 포트홀 이미지 분석을 위한 높은 성능의 딥러닝 모델 적용
 - 딥러닝 기반 포트홀 탐지에서 각 모델을 적용, 모델별 성능을 평가하여 포트홀 탐지 가능성을 확인
 - 딥러닝 기반 포트홀 탐지에 Tensorflow Slim 모델의 딥러닝 모델 라이브러리로 사용
- 이미지 셋
- 딥러닝 모델을 학습할 때 사용되는 이미지 집합. 이미지들이 각 카테고리에 맞추어 저장됨. 딥러닝 모델은 각 카테고리 별로 이미지를 학습
 - 포트홀 및 비 포트홀 이미지로 분류 및 딥러닝 모델에 적용
 - 도로 주행 영상의 이미지화 후 포트홀 및 비 포트홀 이미지로 분류
 - 딥러닝 학습 모델에 적용하기 위한 이미지
- 딥러닝 학습 모델
- 딥러닝 모델이 이미지 셋을 통해 학습이 된 상태
 - 딥러닝 모델이 포트홀 이미지 셋을 학습하여 포트홀 탐지에 사용
 - Tensorflow Slim에서 제공되는 모델을 포트홀 이미지 셋을 사용하여 학습
- TensorFlow Serving
- 서버 기반의 원격 딥러닝 학습 모델 서비스 라이브러리. 원격으로 학습 모델을 서비스하기 위한 라이브러리. 딥러닝 학습 모델들의 버전관리 기능 제공 및 모델 버전별 서비스 기능 제공. 딥러닝 학습이 어려운 장치에서 딥러닝 학습 가능한 서버의 모델을 원격으로 호출하여 결과 획득
 - 딥러닝 학습 모델을 원격으로 모바일 기기에 판독 결과를 서비스하는 프레임워크임
- Inference 모델
- 딥러닝 학습 모델이 데이터에 대한 판독을 수행 할 수 있는 모델이며, Inference화 된 모델은 추가 학습이 불가능 함. 딥러닝 학습 모델을 라이브러리화하여 타 장치에서 호출하여 사용
 - 포트홀 딥러닝 학습 모델을 모바일에 사용할 수 있도록 검토함
- Inception 모델
- Google에서 개발하고 ILSVRC 2014에서 우승한 GoogLeNet 학습 모델의 개량형 버전. 최신 모델은 v4. Inception 모델을 사용하여 이미지 분류를 수행
- ResNet 모델

- Microsoft에서 개발하고 ILSVRC 2015에서 우승한 학습 모델. 최신 모델은 v2. 학습 시 시간이 매우 오래 걸리나 데이터 판단에는 시간이 빠른 것이 특징임
- Inception-ResNet 학습 모델
 - Google의 Inception모델을 ResNet 모델의 특징을 적용하여 데이터 판단속도를 향상 시킨 모델임
- MobileNet 모델
 - Inception모델을 저 성능의 모바일 장치에 사용될 수 있도록 경량화 시킨 모델
 - 모바일용 포트홀 딥러닝 학습 모델로 활용성 검토

1.3 임베디드 단말 프레임워크

- OpenCV for Android
 - 입력된 영상을 통하여 사용자가 정의한 특징점을 탐지하기 위하여 제공되는 오픈소스 기반 영상처리 라이브러리
 - 안드로이드 카메라를 통하여 실시간으로 입력되는 영상으로 포트홀 탐지에 사용
 - 안드로이드의 카메라로부터 입력된 영상 기반, 해당 라이브러리가 제공 하고 있는 알고리즘을 사용하여 본 연구의 특징점 분류를 테스트 하고, 본 연구의 적용 가능성을 확인
 - 기본 영상 처리 라이브러리로 사용가능함
- 안드로이드 카메라 API
 - 영상 처리를 위하여 필요한 기능들이 사전에 구현이 되어있는 라이브러리
 - 안드로이드 영상처리 작업 필요시, 사전에 구현되어 있는 기능들을 가져다가 사용
 - 영상처리 라이브러리 기반으로 작성되어, 안드로이드 포팅이 가능할 것으로 판단
- NDK
 - c/c++ 으로 만들어진 알고리즘을 JAVA에서도 사용 가능하게 해주는 개발 도구
 - 알고리즘이 카메라를 통하여 입력된 영상을 통해 실시간 탐지가 되는지 테스트하는 과정으로 활용함
- Exif for Android
 - JPEG, WAV 등의 이미지 확장 포맷에서 사용되는 사진정보를 저장하는 라이브러리
 - 도로파손객체 탐지시 저장된 해당 이미지에 GPS 좌표를 기록을 위해 사용
 - 포트홀로 탐지된 이미지에 현재 좌표를 기록하여 해당 사진만을 가지고 사진이 저장된 위치를 찾아 표기하는 방법으로 설계함
- 안드로이드 어플리케이션 UI/UX

- 사용자가 어플리케이션을 사용 시에 화면상에 보여주는 버튼 등과 같은 레이아웃을 설계함
- 어플 UI 기능설계와 메뉴 디자인
- 수치를 입력하는 방식으로 ROI 설정, 설정 페이지를 통한 모드 변경 등을 설계
- 탐지 결과 관리 어플리케이션 연구 개발
 - 탐지한 결과를 이미지와 함께 지도에 표기토록 설계
 - 사용자 편의성의 어플리케이션은 계속 업데이트 필요

5. 응용프로그램 구축 내용

5.1. 화면 구성

- 포트홀 단말에 설치되는 탐지 프로그램의 화면 구성은 크게 환경설정 화면과 탐지메인화면으로 구성된다.

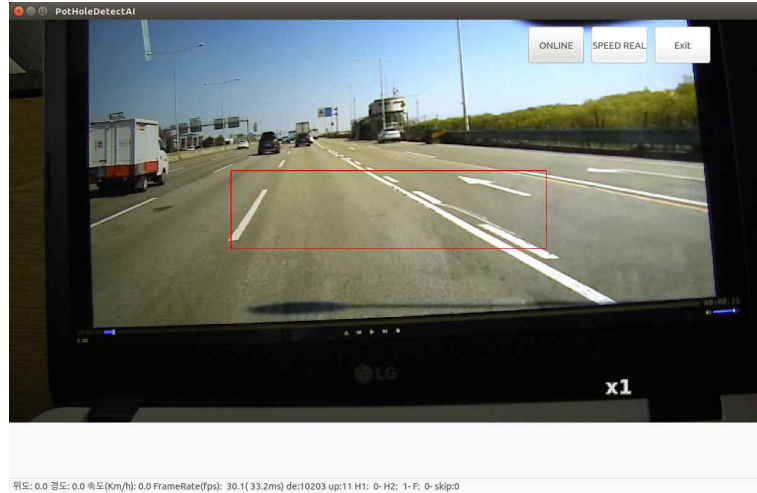
가. 환경설정

프로그램을 실행시키면 아래 그림과 같은 환경설정 화면이 표시되며, 환경설정 화면에는 간단한 장치 정보가 표시되고 탐지된 포트홀 이미지 및 GPS 데이터 등을 전송하기 위한 서버도메인, 포트, 주기 보고를 위한 GPS 전송주기, ROI 영역의 X, Y 좌표 등을 설정할 수 있다.

환경설정	
Device Test :	OK
GPS Speed:	Real
Input Source:	Camera
Auto Start in	60 seconds
서버 도메인:	g-receive.pothole.kr
서버 port:	12000
GPS 전송주기(s):	10
탐지영역 X좌표:	560
탐지영역 Y좌표:	440
Cancel OK (40)	

나. 탐지메인

환경설정을 마치고 확인 버튼을 클릭하면 아래 그림과 같이 탐지메인화면이 표시되고 ROI 영역이 빨간색 사각형으로 표시된다. ROI 영역이 도로 노면을 바라보도록 카메라를 조정한 후 탐지하면 된다.



5.2. 2차 탐지 구동 화면

◦ Sender 구동 화면

Sender는 2차 탐지 서버에 위치하여 1차 DB 및 1차 이미지 저장소로부터 데이터 및 이미지를 가져와서 메시지 큐의 일종인 RabbitMQ를 통해 Receiver에 일감을 분배해 주는 역할을 하는 프로그램으로서 아래 그림과 같이 구동된다.

```
root@ubuntu: /home/user/pothole_test
, 0.08622, 0.04790, 0.03832, 0.08622, 0.07664, 0.05269, 0.07185, 0.14370, 0.1724
4', '1', '1.00')
20191014113007617#00-04-4b-c6-ae-b0#20191014#113007617#0.0#0.0#0.0#0.0#10.0#1#0
0-04-4b-c6-ae-b0#20191014#113007617.jpg#2019-10-14 12:32:46.910000#00-04-4b-c6-ae-b0#20191014#orig\113007617.jpg#0#001#N#None#None#100# 0.09101, 0.17723, 0.1676
5, 0.06227, 0.10538, 0.05748, 0.10538, 0.09101, 0.13412, 0.07185, 0.09580, 0.091
01, 0.04790, 0.08622, 0.10059, 0.05748, 0.11017, 0.05269, 0.10538, 0.07185, 0.03
832, 0.09101, 0.05269, 0.09101, 0.07664, 0.05748, 0.05269, 0.11975, 0.10538, 0.0
8622, 0.04311, 0.15807, 0.18202, 0.05748, 0.05748, 0.06706, 0.06227, 0.07185, 0.0
05748, 0.08143, 0.10538, 0.06706, 0.03353, 0.09580, 0.02874, 0.05748, 0.02874, 0
.11975, 0.09101, 0.12933, 0.10538, 0.09101, 0.08622, 0.13412, 0.10538, 0.07664,
0.01916, 0.03353, 0.11017, 0.08622, 0.08622, 0.01916, 0.11975, 0.12454, 0.03832,
0.09101, 0.03832, 0.08622, 0.08622, 0.03832, 0.05269, 0.07185, 0.05748, 0.02874
, 0.01916, 0.06706, 0.07664, 0.07664, 0.10059, 0.17723, 0.09580, 0.06227, 0.0526
9, 0.08143, 0.13412, 0.11496, 0.03832, 0.04790, 0.09101, 0.07185, 0.09101, 0.086
22, 0.04790, 0.03832, 0.08622, 0.07664, 0.05269, 0.07185, 0.14370, 0.17244#1#1.0
0#
row('00-04-4b-c6-ae-b0', '20191014111630258', '20191014', '111630258', '0.0', '0
.0', '0.0', '0.0', '10.0', '1', '', '00-04-4b-c6-ae-b0\\20191014\\111630258.jpg',
datetime.datetime(2019, 10, 14, 12, 32, 46, 453000), '00-04-4b-c6-ae-b0\\20191
014\\orig\\111630258.jpg', '0', '001', 'N', None, None, 100, '0.24908, 0.13891,
0.22992, 0.31614, 0.03353, 0.18202, 0.28740, 0.22034, 0.03353, 0.08622, 0.39278
, 0.29698, 0.13412, 0.31135, 0.21555, 0.19639, 0.22513, 0.18681, 0.53169, 0.1532
8, 0.64666, 0.37841, 0.50295, 0.13412, 0.23950, 0.33051, 0.59875, 0.35925, 0.287
```

◦ Receiver 구동 화면

Receiver는 여러 개의 프로세스로 구성되며, 각각의 Receiver는 RabbitMQ로부터 일감을 할당받아 2차 탐지 알고리즘을 적용하여 1차에서 탐지된 포트홀을 좀 더 정확하게 필터링하여 2차 DB 및 2차 이미지 저장소에 전달하는 역할을 수행하는 프로그램으로서 아래 그림과 같이 구동된다.

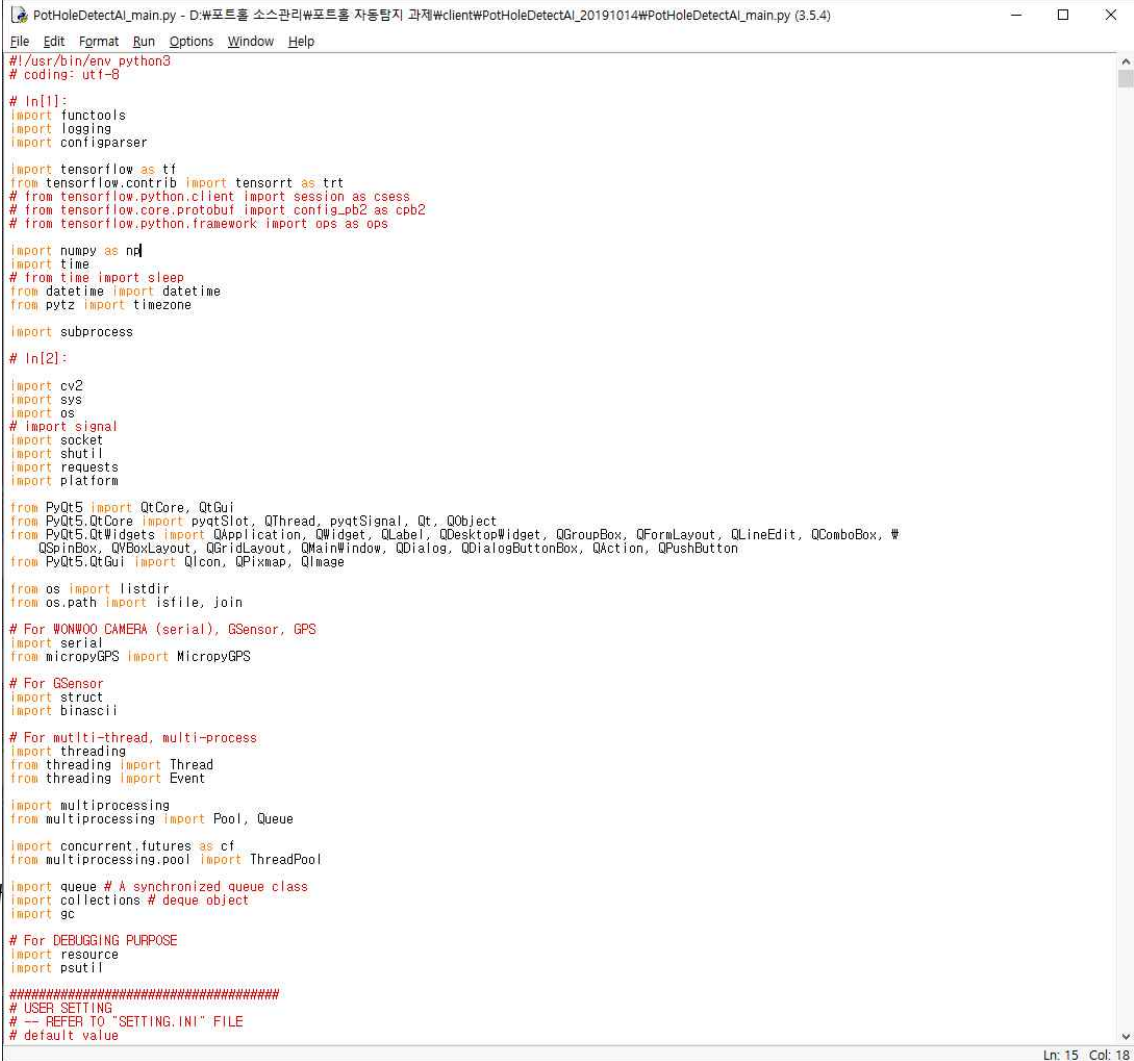
```
root@ubuntu: /home/user/pothole_test
ne/120559207.jpg
[2019-10-14 03:32:36,304] > 1차 이틀 = 0:00:00.766250

*****Callback() Start*****
[x] Received b'20191014115742134#00-04-4b-c6-ae-b0#20191014#115742134#0.0#0.0#0
.0#0.0#10.0#1#00-04-4b-c6-ae-b0\\20191014\\115742134.jpg#2019-10-14 12:32:32.06
7000#00-04-4b-c6-ae-b0\\20191014\\orig\\115742134.jpg#0#001#N#None#None#100# 0.0
9101, 0.08622, 0.10059, 0.03832, 0.11496, 0.07185, 0.06706, 0.12933, 0.03353, 0.
10059, 0.09101, 0.07185, 0.02874, 0.08143, 0.05269, 0.22513, 0.11496, 0.13891, 0
.12933, 0.09101, 0.06227, 0.05269, 0.10059, 0.05748, 0.09101, 0.11496, 0.12933,
0.12933, 0.11496, 0.10059, 0.10538, 0.03353, 0.10538, 0.12454, 0.11496, 0.10538,
0.10059, 0.15807, 0.05269, 0.16765, 0.14370, 0.04311, 0.01916, 0.05269, 0.05269
, 0.07664, 0.10538, 0.05748, 0.10059, 0.09580, 0.09101, 0.05748, 0.15328, 0.1341
2, 0.10538, 0.03353, 0.04790, 0.07664, 0.06227, 0.08143, 0.06227, 0.06227, 0.038
32, 0.05748, 0.09101, 0.04790, 0.05748, 0.08143, 0.06706, 0.06706, 0.07664, 0.05
748, 0.09101, 0.08622, 0.06227, 0.14370, 0.02874, 0.03832, 0.11975, 0.12933, 0.0
7664, 0.11496, 0.11975, 0.08622, 0.10059, 0.09101, 0.11017, 0.12454, 0.05269, 0.
05748, 0.07664, 0.10538, 0.10538, 0.12454, 0.09580, 0.10538, 0.11017, 0.16765, 0
.19160, 0.02874#1#1.00#
[2019-10-14 03:32:36,306] > *****20191014115742134/00-04-4b-c6-ae-b0
start *****
[2019-10-14 03:32:36,306] > make_directory 소요 시간 = 0:00:00.000019
```

5.3. 소스 코드

- 1차 탐지 함수 선언부

1차 탐지 프로그램에 사용되는 함수들에 대한 선언을 보여주는 코드이다.



```
# PotHoleDetectAI_main.py - D:\포트홀 소스관리\포트홀 자동탐지 과제\client\PotHoleDetectAI_20191014\PotHoleDetectAI_main.py (3.5.4)
File Edit Format Run Options Window Help

# l/usr/bin/env python3
# coding: utf-8

# In[1]:
import functools
import logging
import configparser

import tensorflow as tf
from tensorflow.contrib import tensorrt as trt
# from tensorflow.python.client import session as csess
# from tensorflow.core.protobuf import config_pb2 as cpb2
# from tensorflow.python.framework import ops as ops

import numpy as np
import time
# from time import sleep
from datetime import datetime
from pytz import timezone

import subprocess

# In[2]:
import cv2
import sys
import os
# import signal
import socket
import shutil
import requests
import platform

from PyQt5 import QtCore, QtGui
from PyQt5.QtCore import pyqtSlot, QThread, pyqtSignal, Qt, QObject
from PyQt5.QtWidgets import QApplication, QWidget, QLabel, QDesktopWidget, QGroupBox, QFormLayout, QLineEdit, QComboBox, #
QSpinBox, QBoxLayout, QGridLayout, QMainWindow, QDialog, QDialogButtonBox, QAction, QPushButton
from PyQt5.QtGui import QIcon, QPixmap, QImage

from os import listdir
from os.path import isfile, join

# For WONWOO CAMERA (serial), GSensor, GPS
import serial
from micropyGPS import MicropyGPS

# For GSensor
import struct
import binascii

# For multi-thread, multi-process
import threading
from threading import Thread
from threading import Event

import multiprocessing
from multiprocessing import Pool, Queue

import concurrent.futures as cf
from multiprocessing.pool import ThreadPool

import queue # A synchronized queue class
import collections # deque object
import gc

# For DEBUGGING PURPOSE
import resource
import psutil

#####
# USER SETTING
# -- REFER TO "SETTING.INI" FILE
# default value
```

- 1차 탐지 환경설정 다이얼로그

프로그램 실행시 최초로 표시되는 환경설정 다이얼로그 화면에 대한 코드이다.

```
PotHoleDetectAI_main.py - D:\소프트웨어 소스관리\소프트웨어 자동탐지 과제\wclient\WPotHoleDetectAI_20191014\WPotHoleDetectAI_main.py (3.5.4)
File Edit Format Run Options Window Help

class Dialog(QDialog):
    def __init__(self):
        super(Dialog, self).__init__()
        self.createFormGroupBox()

        self.buttonBox = QDialogButtonBox(QDialogButtonBox.Ok | QDialogButtonBox.Cancel)
        self.buttonBox.accepted.connect(self.accept)
        self.buttonBox.rejected.connect(self.reject)

        mainLayout = QVBoxLayout()
        mainLayout.addWidget(self.formGroupBox)
        mainLayout.addWidget(self.buttonBox)
        self.setLayout(mainLayout)

        self.setWindowTitle("PotHoleDetectAI")

        self.timer = QTimer()
        self.timeout = 60 # 60 seconds
        self.timer.timeout.connect(self.tick)
        self.timer.setInterval(1000) # 1000 ms

        self.t_btn_text = self.buttonBox.button(QDialogButtonBox.Ok)
        self.t_btn_text.setText("OK ({})".format(self.timeout))

        """
        # CONFIGPARSER
        # default - REAL SPEED, USE CAMERA
        """
        try:
            config = configparser.ConfigParser()
            config.read("settings.ini")
            v1 = config["settings"]["simulated_gps_speed"]
            settings.sim_gps(v1)
            v2 = config["settings"]["useCamera"]
            settings.use_camera(v2)
            print(">>> v1 {} v2 {}".format(v1, v2))

        except Exception as ex:
            print(">>> [INI FILE READING - Exception ", ex)
            print(">>> ASSUME -----")
            print(["settings"])
            print("simulated_gps_speed = No ")
            print("useCamera = yes ")
            print(">>>-----")
            settings.sim_gps("No") # False
            settings.use_camera("Yes") # True

        finally:
            print(">>> (test) useCamera : ", b_useCamera)
            print(">>> (test) b.simulated_gps_speed : ", b_simulated_gps_speed)

    def createFormGroupBox(self):
        global g_tdd_ret
        global g_tdd_ret_code
        global b_simulated_gps_speed
        global b_useCamera

        self.formGroupBox = QGroupBox("환경설정")
        layout = QFormLayout()

        self.leGPSPeriod = QLineEdit(SET_GPS_PERIOD) #GPS period (sec)
        self.leServerDomain = QLineEdit(SET_SERVER_DOMAIN_NAME) #Server Domain
        self.leServerPort = QLineEdit(SET_SERVER_PORT) #Server Port
        self.leROIx = QLineEdit(SET_ROI_START_X) # ROI x position
        self.leROIy = QLineEdit(SET_ROI_START_Y) # ROI y position
        self.leROIwidth = QLineEdit(800) # ROI width
        self.leROIheight = QLineEdit(200) # ROI height
```

◦ 1차 탐지 메인화면

환경설정을 마치면 표시되는 메인화면에 대한 코드이다.

```

class DetectMain(QMainWindow):
    video_shower = None

    # Set Window size, ROI size, GPS and so forth
    cvWidth = SET_VIEWWINDOW_WIDTH_INT # int value, not string
    cvHeight = SET_VIEWWINDOW_HEIGHT_INT # int value, not string

    roiX = int(SET_ROI_START_X)
    roiY = int(SET_ROI_START_Y)
    roiWidth = int(SET_ROI_WIDTH)
    roiHeight = int(SET_ROI_HEIGHT)
    gpsPeriod = int(SET_GPS_PERIOD)

    serverIP = ""
    serverPort = ""

    threads = []

    def __init__(self, serverDomain, serverPort, gpsPeriod, roiX, roiY):
        super().__init__()

        try:
            # directory 생성
            dirname = DetectThread.basePath + "~/nvidia/PotHole/detect/"
            if not os.path.exists(dirname):
                os.makedirs(dirname)

            # get server ip from domain name
            DetectMain.serverIP = socket.gethostbyname(serverDomain)
            # DetectMain.serverIP = "127.0.0.1" # "183.107.16.248"
            DetectMain.serverPort = serverPort
            DetectMain.gpsPeriod = gpsPeriod # GPS 전송주기 (sec)
            DetectMain.roiX = int(roiX) # roi X좌표
            DetectMain.roiY = int(roiY) # roi Y좌표

            # self.lati = lati #위도
            # self.longi = longi #경도
            # self.velocity = velocity #속도

            # self.lati = 32.456789 # 위도
            # self.longi = 127.123456 # 경도
            # self.velocity = 60 # 속도

            self.title = 'PotHoleDetectAI'
            self.left = 0
            self.top = 0
            self.width = SET_VIEWWINDOW_WIDTH_INT
            self.height = SET_VIEWWINDOW_HEIGHT_INT
            self.initUI()

        except Exception as ex:
            print("장애가 발생 했습니다. 잠시 후 다시 시도해 주시기 바랍니다.", str(ex))
            print("exit")

            # DEBUG
            if b_DEBUG_uselogging:
                logging.error("장애가 발생 했습니다. 잠시 후 다시 시도해 주시기 바랍니다." + str(ex))
                logging.error("exit")
            # self.timer.stop()
            # self.timerGPS.cancel()
            # self.thDetect.terminate()
            # self.thGPS.terminate()
            # DetectThread.cap.release()

            # QApplication.exit(0)
            sys.exit(0)

    @pyqtSlot(QImage)
    def setImage(self, image):
        self.label.setPixmap(QPixmap.fromImage(image))

```

- 1차 탐지 Video Getter

메인화면에서 카메라 입력을 받아서 각각의 프레임들을 가져오는 코드이다.

```

class VideoGet2:
    """
    Class that continuously gets frames from a VideoCapture object
    with a dedicated thread.
    """
    global h_frames

    def __init__(self, src=None):
        self.stream = cv2.VideoCapture(src)

        # -----
        # cv_fps2 = DetectThread.cap.get(5) # FPS
        # print(">>> fps (old) ", cv_fps2, n_CAM_SET_FPS)
        # -----

        # DetectThread.cap = cv2.VideoCapture(source)
        # cv_ben = DetectThread.cap.get(BackendName) # NOT SUPPORTED
        # cv_isopened = self.stream.isOpened() #
        # w = self.stream.set(3, n_CAM_SET_WIDTH) # width - 1920
        # h = self.stream.set(4, n_CAM_SET_HEIGHT) # height - 1080
        # fps = self.stream.set(5, n_CAM_SET_FPS) # cv2.CAP_PROP_FPS - 30 or 60
        # w = DetectThread.cap.set(3, 1080) # width - 1920
        # h = DetectThread.cap.set(4, 720) # height - 1080
        # fps = DetectThread.cap.set(5, 60.0) # cv2.CAP_PROP_FPS - 30 or 60

        # -----
        cv_w = self.stream.get(3) # width
        cv_h = self.stream.get(4) # height
        cv_fps = self.stream.get(5) # FPS
        # cv_format = DetectThread.cap.get(8) # FORMAT
        cv_mode = self.stream.get(9) # MODE

        print(">>> CAMERA INFO ")
        # print(">>> Backendname ", cv_ben)
        print(">>> IsOpened ", cv_isopened)
        print(">>> w, h ", w, h, cv_w, cv_h)
        print(">>> fps ", fps, cv_fps)
        # print(">>> format ", cv_format)
        print(">>> mode ", cv_mode)
        # -----

        if self.stream.isOpened() == False:
            print("Error opening video stream or file")

        self.grabbed, self.frame = self.stream.read()
        self.stopped = False
        self.index = 0 # index - sum of all
        self.vg2_index = 0 # index - normal case
        self.vg2_skipindex = 0 # index - timeout
        self.vg2_index_error = 0 # index - error cases
        self.vg2_index_notsend = 0 # index - not send due to throttling
        # throttling
        self.b_put_frame = True # enable put frame to H1 queue (GET --> SESS)
        # ev_vg.clear()
        logging.info("VGET - init:")

    def start(self):
        logging.info("VGET - start:")
        Thread(target=self.get, args=()).start()
        return self

    def get(self):
        global h_frames
        global b_MON_inhibit_Q_H_frame

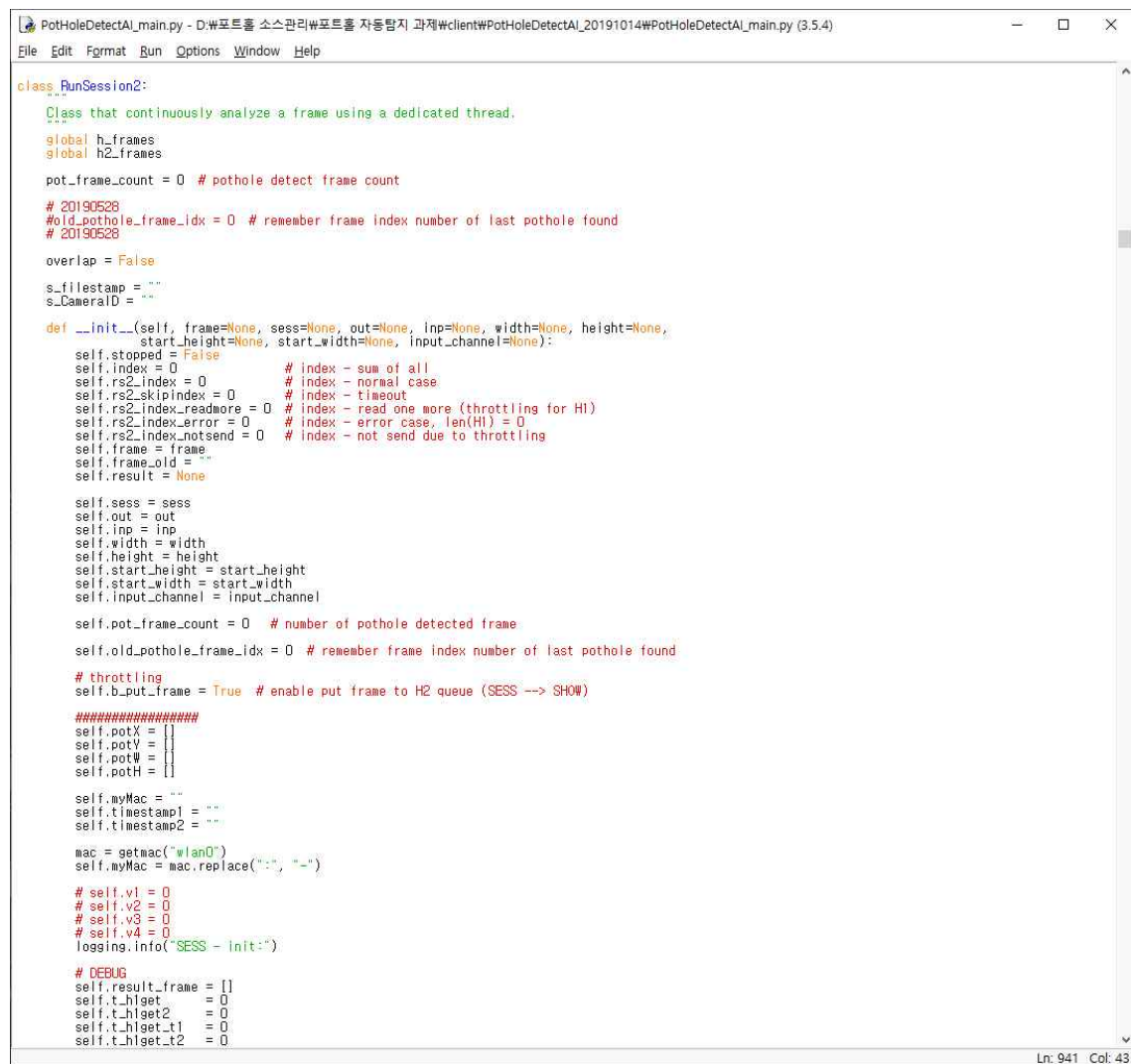
        self.t_end2 = time.time()
        log_DBG_VGET_e2e_time.append(0) # initial end-to-end time (ms)

        self.gc_inuse = False
        while self.stream.isOpened() and not self.stopped:

```

- 1차 탐지 Session Runner

Video Getter를 통해 받아온 프레임에 대해 인공지능 알고리즘을 적용하여 추론을 하기 위한 코드이다.



```
PotHoleDetectAI_main.py - D:\소프트웨어 소스관리\소프트웨어 자동탐지 과제\client\PotHoleDetectAI_20191014\PotHoleDetectAI_main.py (3.5.4)
File Edit Format Run Options Window Help

class RunSession2:
    """
    Class that continuously analyze a frame using a dedicated thread.
    """
    global h_frames
    global h2_frames

    pot_frame_count = 0 # pothole detect frame count

    # 20190528
    #old_pothole_frame_idx = 0 # remember frame index number of last pothole found
    # 20190528

    overlap = False

    s_filestamp = ""
    s_CameraID = ""

    def __init__(self, frame=None, sess=None, out=None, inp=None, width=None, height=None,
                 start_height=None, start_width=None, input_channel=None):
        self.stopped = False
        self.index = 0 # index - sum of all
        self.rs2_index = 0 # index - normal case
        self.rs2_skipindex = 0 # index - timeout
        self.rs2_index_readmore = 0 # index - read one more (throttling for H1)
        self.rs2_index_error = 0 # index - error case, len(H1) = 0
        self.rs2_index_notsend = 0 # index - not send due to throttling
        self.frame = frame
        self.frame_old = None
        self.result = None

        self.sess = sess
        self.out = out
        self.inp = inp
        self.width = width
        self.height = height
        self.start_height = start_height
        self.start_width = start_width
        self.input_channel = input_channel

        self.pot_frame_count = 0 # number of pothole detected frame

        self.old_pothole_frame_idx = 0 # remember frame index number of last pothole found

        # throttling
        self.b_put_frame = True # enable put frame to H2 queue (SESS --> SHOW)

        #####
        self.potX = []
        self.potY = []
        self.potW = []
        self.potH = []

        self.myMac = ""
        self.timestamp1 = ""
        self.timestamp2 = ""

        mac = getmac("wlan0")
        self.myMac = mac.replace(":", "-")

        # self.v1 = 0
        # self.v2 = 0
        # self.v3 = 0
        # self.v4 = 0
        logging.info("SESS - init:")

        # DEBUG
        self.result_frame = []
        self.t_higet = 0
        self.t_higet2 = 0
        self.t_higet_t1 = 0
        self.t_higet_t2 = 0
```

Ln: 941 Col: 43

◦ 1차 탐지 Detect Thread

포트홀 탐지를 위한 코드로서 Thread 형태로 구성된다.

```
PotHoleDetectAI_main.py - D:\포트홀 소스관리\포트홀 자동탐지 과제\client\WPotHoleDetectAI_20191014\WPotHoleDetectAI_main.py (3.5.4)
File Edit Format Run Options Window Help

class DetectThread(Thread):
    global b_useCamera

    changePixmap = pyqtSignal(QImage)
    queue_from_cam = multiprocessing.Queue()

    pot_frame_count = 0 # pothole detect frame count
    old_pothole_frame_idx = 0 # remember frame index number of last pothole found
    overlap = False

    sess_result = None
    inp_result = None
    pred_result = None

    # =====
    print(">>>> CHECK VERSIONS")
    print(">>>> ")
    print(">>>> Platform : ", platform.sys.version)
    print(">>>> Platform.uname : ", platform.uname())
    #print(">>>> Platform.platform : ", platform.platform())
    #print(">>>> Platform.system : ", platform.system())
    #print(">>>> Platform.release : ", platform.release())
    #print(">>>> Platform.version : ", platform.version())
    print(">>>> Platform.python_build : ", platform.python_build())
    print(">>>> Numpy : ", np.version.version)
    print(">>>> OpenCV : ", cv2.__version__)
    print(">>>> Tensorflow : ", tf.__version__)
    print(">>>> TensorRT : skip ") # TensorRT does not support version string
    print(">>>> ")
    print(">>>> CHECK GLOBAL SETTINGS")
    print(">>>> useTensorRT : ", b_useTensorRT)
    print(">>>> useRamdisk : ", b_DEBUG.useRamdisk)
    print(">>>> frame_skip_idx : ", frame_skip_idx)
    print(">>>> (test) useCamera : ", b_useCamera)
    print(">>>> (test) b_simulated_gps_speed : ", b_simulated_gps_speed)
    print(">>>> ")
    #print(">>>> (test) loadVideofileToRamdisk : ", b_DEBUG.loadVideofileToRamdisk)
    #print(">>>> (test) n_testInputFrameNumber : ", n_testInputFrameNumber)
    #print(">>>> ")
    print(">>>> (test) b_use_coarse_contour_detect : ", b_use_coarse_contour_detect)
    print(">>>> ")
    print(">>>> (DEBUG) OSD_mode : ", b_DEBUG.OSD_mode)
    print(">>>> (DEBUG) useLogging : ", b_DEBUG.useLogging)
    print(">>>> (DEBUG) logToFile : ", b_DEBUG.logToFileOnly)
    print(">>>> ")
    print(">>>> (DEBUG) READCAM_LOOPTEST : ", b_DEBUG.READCAM_LOOPTEST)
    print(">>>> (DEBUG) SAVE_FPS_LOOPTEST : ", b_DEBUG.SAVE_FPS_LOOPTEST)
    print(">>>> ")
    print(">>>> (DEBUG) bypass_CamLoop : ", b_DEBUG.bypass_CamLoop)
    print(">>>> ")
    print(">>>> (DEBUG) useGPSThread : ", b_DEBUG.useGPSThread)
    print(">>>> (DEBUG) useGsensorThread : ", b_DEBUG.useGsensorThread)
    print(">>>> (DEBUG) useSendThread : ", b_DEBUG.useSendThread)
    print(">>>> (DEBUG) useCamLoop : ", b_DEBUG.useCamLoop)
    print(">>>> (DEBUG) useVISCAThread : ", b_DEBUG.useVISCAThread)
    print(">>>> ")
    print(">>>> FILENAME : ", __file__)
    s_filestamp = checkFileTimestamp()
    print(">>>> FILE_TIMESTAMP : ", s_filestamp)
    s_CameraID = checkCameraID()
    print(">>>> USBCAMERAID : ", s_CameraID)
    print(">>>> done")

    if s_CameraID == "None":
        # CAMERA FAIL
        playth = threading.Thread(target=PlayBeep_ERROR, args=())
        playth.start()

    print(">>>> (DEBUG) b_DEBUG_3_READCAM_LOOPTEST : ", b_DEBUG_3_READCAM_LOOPTEST)
```

◦ 1차 탐지 Video Shower

탐지된 포트홀을 표시해 주기위한 코드이다.

```
class VideoShow2(QObject):
    """
    Class that continuously shows a frame using a dedicated thread.
    """

    global h2_frames

    changePixmap = pyqtSignal(QImage)

    def __init__(self, frame=None):
        super(VideoShow2, self).__init__()

        self.frame = frame
        self.stopped = False
        # ev_vs.clear()
        self.index = 0 # index - sum of all
        self.vs2_index = 0 # index - normal case
        self.vs2_skipindex = 0 # index - timeout
        self.vs2_index_error = 0 # index - error case, len(H2) = 0, H2 frame empty
        #self.vs2_index_notsend = 0 # index - not send due to throttling
        logging.info("SHOW - init:")

        # DEBUG
        self.t_emit_end = 0
        self.t_emit_start = 0

    def start(self):
        logging.info("SHOW - start:")
        self.th = Thread(target=self.show, args=())
        self.th.start()
        return self

    def show(self):
        global h2_frames

        self.t_end2 = time.time()
        log_DBG_VSHOW_e2e_time.append(time.time() + 1000) # initial end-to-end time (ms)

        while not self.stopped:
            try:
                # wait for event
                logging.info("SHOW - wait : {} {} {}".format(ev_vg.is_set(), ev_sr.is_set(), ev_vs.is_set()))
                self.t_1 = time.time()
                ev_vs.wait(timeout=TIMEOUT_VSHOW)
                self.t_2 = time.time()
                self.t_await = (self.t_2 - self.t_1) + 1000
                logging.info("SHOW - wait exit: {} {} {} {} {}ms".format(ev_vg.is_set(), ev_sr.is_set(), ev_vs.is_set(), self.t_await ))

                # DEBUG
                self.t_start = time.time()

                # check if event or timeout
                event_flag = ev_vs.is_set()

                if event_flag != True:
                    self.vs2_skipindex += 1
                    logging.warning("+++ SHOW - TIMEOUT")
                    if len(h2_frames) != 0:
                        logging.error("+++ SHOW - TIMEOUT & H2 IS NOT EMPTY")
                        # TODO - EVENT TIMEOUT but Q is not empty. will be handled at next EVENT.SET.
                else:
                    # NEW - clear event asap (to allow other faster switching)
                    ev_vs.clear()
                    logging.info("SHOW - clear(SHOW)")

                    # handle multiple items at a time
                    # TODO - qs2 should be 1(mean max_range is 1 enough), otherwise event(SESS->SHOW) have some problem
                    # set max_range by checking H2 THROTTLING FLAG
```

◦ 1차 탐지 File Saver

탐지된 포트홀 이미지 및 데이터 파일을 저장하기 위한 코드이다.

```
PotHoleDetectAI_main.py - D:\포트홀 소스관리\포트홀 자동탐지 과제\client\PotHoleDetectAI_20191014\PotHoleDetectAI_main.py (3.5.4)
File Edit Format Run Options Window Help
# logging.info("SAVER- img_save_file - texts {0:5.1f}ms".format( (t_end3 - t_start3)*1000 ))
#
def save_image_file_async(self, timestamp1, timestamp2, frame_pot, frame_orig):
    # self.myMac -> KEEP USE (static)
    # self.timestamp1 -> timestamp1
    # self.timestamp2 -> timestamp2
    # self.frame_pot -> frame_pot
    # self.frame_orig -> frame_orig

    # save detect frame
    filename = "v2_" + self.myMac + "_" + timestamp1 + timestamp2 + ".jpg"
    filepath = self.myMac + "_" + timestamp1 + "_" + timestamp2 + ".jpg"
    t_start = time.time()
    result = cv2.imwrite(DetectThread.imSavePath + filename, frame_pot)
    t_end = time.time()
    logging.info("SAVER- save_image_file - frame {0:5.1f}ms".format( (t_end - t_start)*1000))

    # save orig frame
    orig_filename = "v2_" + self.myMac + "_" + timestamp1 + "_orig_" + timestamp2 + ".jpg"
    orig_filepath = self.myMac + "_" + timestamp1 + "_orig_" + timestamp2 + ".jpg"
    t_start2 = time.time()
    orig_result = cv2.imwrite(DetectThread.imSavePath + orig_filename, frame_orig)
    t_end2 = time.time()
    logging.info("SAVER- save_image_file - orig_frame {0:5.1f}ms".format( (t_end2 - t_start2)*1000))

def save_data_file_async(self, timestamp1, timestamp2, list_ph, ph_pix, gps_lat, gps_lng, gps_alti, gps_course, gps_speed, gps_status, gps_addr):
    # self.myMac -> KEEP USE (static)
    # self.timestamp1 -> timestamp1
    # self.timestamp2 -> timestamp2
    # self.list_ph -> list_ph
    # self.ph_pix -> ph_pix
    # GPSThread.lat -> self.gps_lat -> gps_lat
    # GPSThread.lng -> self.gps_lng -> gps_lng
    # GPSThread.alti -> self.gps_alti -> gps_alti
    # GPSThread.course -> self.gps_course -> gps_course
    # GPSThread.speed -> self.gps_speed -> gps_speed
    # GPSThread.status -> self.gps_status -> gps_status
    # GPSThread.addr -> self.gps_addr -> gps_addr

    # data file create
    t_start3 = time.time()

    # NEW
    # direct write
    #
    # list_ph[0] : number of pothole
    # list_ph[1] : array of X
    # list_ph[2] : array of Y
    # list_ph[3] : array of W
    # list_ph[4] : array of H

    # get common data
    # GPSThread.lat -> self.gps_lat
    # GPSThread.lng -> self.gps_lng
    # GPSThread.alti -> self.gps_alti
    # GPSThread.course -> self.gps_course
    # GPSThread.speed -> self.gps_speed
    # GPSThread.status -> self.gps_status
    # GPSThread.addr -> self.gps_addr

    # get GSENSOR data - common one for multiple pothole in a frame
    time.sleep(1.0) # SLEEP 1.0 sec to get data after pothole detection
    # TODO - NEED TO MAKE A GRACE DESIGN, sleep(1.0) waste time.

    # PROTOCOL Ver 1.4 - GSENSOR FLOAT(F8.5) format
    # Data format is "noofitems# 0.12345,12.34567,....., 0.12345#" (data field in F8.5 format)
    #
    filepath = self.myMac + "_" + timestamp1 + "_" + timestamp2 + ".jpg"
    orig_filepath = self.myMac + "_" + timestamp1 + "_orig_" + timestamp2 + ".jpg"

    n_gs = len(DBG_gs_pkt_angle)
```

Ln: 2167 Col: 28

◦ 1차 탐지 파일 전송 Thread

저장된 포트홀 이미지 및 데이터 파일을 전송하기 위한 Thread 형식의 코드이다.

```
PotHoleDetectAI_main.py - D:\#포트홀 소스관리\#포트홀 자동탐지 과제\client\WPotHoleDetectAI_20191014\WPotHoleDetectAI_main.py (3.5.4)
File Edit Format Run Options Window Help

# @staticmethod
# sendThread - repeats every 3 seconds by timer
def sendThread(self):
    logging.info(">>>> sendThread - start")

    filePath = []
    fileName = []

#
    if DetectMain.serverIP != "" and DetectMain.serverPort != "":
        if DetectMain.serverIP != "" and DetectMain.serverPort != "" and b_DEBUG_useSendThread:
            ## DEBUG
            #srcPath = DetectThread.basePath + "/nvidia/PotHole/detect/"
            #dstPath = DetectThread.basePath + "/nvidia/PotHole/upload"
            #list_src = os.listdir(srcPath) # dir is your directory path
            #nof_src = len(list_src)
            #list_dst = os.listdir(dstPath) # dir is your directory path
            #nof_dst = len(list_dst)
            #print(">>>>> detect, upload, total ", nof_src, nof_dst, nof_src+nof_dst)
            ## DEBUG

            # Check if too many file is trying to send
            ## NUMBER OF OTHER THREADS = 20 ( DetectMain, Get, Sess, Show, FileSaver and so on)
            if threading.active_count() > FILE_SEND_LIMIT + 20:
                self.sendTh = threading.Timer(3.0, self.sendThread)
                self.sendTh.start()
                ## DEBUG
                #print("FILESEND SKIP - too much socket [{}]" .format(threading.active_count() - 20 ))
                return

            # src, dst directory 생성
            # srcPath = "/home/nvidia/PotHole/detect"
            # dstPath = "/home/nvidia/PotHole/upload"
            srcPath = DetectThread.basePath + "/nvidia/PotHole/detect/"
            if not os.path.exists(srcPath):
                os.makedirs(srcPath)
            dstPath = DetectThread.basePath + "/nvidia/PotHole/upload"
            if not os.path.exists(dstPath):
                os.makedirs(dstPath)

            allFiles = [f for f in os.listdir(srcPath) if os.path.isfile(os.path.join(srcPath, f))]
            filePath.clear()
            fileName.clear()

            ## DEBUG
            #print(">>>> sendThread - len ", len(allFiles) )
            ## DEBUG

            if len(allFiles) > 0:
                for filename in allFiles[:FILE_SEND_LIMIT]:
                    # print(srcPath + filename)
                    # tmpPath = srcPath + "/" + filename
                    # if os.path.isfile(tmpPath) and os.access(tmpPath, os.R_OK):
                    #     # print(filename)
                    ## DEBUG
                    #debug_move_start_t = time.time()

                    shutil.move(os.path.join(srcPath, filename), os.path.join(dstPath, filename))
                    filePath.append(dstPath + "/" + filename)
                    fileName.append(filename)
                    ## DEBUG
                    #debug_move_end_t = time.time()
                    #print(">>>> sendthread-move", (debug_move_end_t-debug_move_start_t)*1000) # average < 1.0 ms (for non RAMDISK)
                    ## DEBUG

                ## DEBUG
                #print(">>>> sendThread - moved ")
                ## DEBUG

            # DetectMain.threads = []
            DetectMain.threads.clear()

Ln: 5396 Col: 56
```

◦ 1차 탐지 GPS Thread

탐지된 순간의 GPS 데이터를 수집하기 위한 Thread 형식의 코드이다.

```
PotHoleDetectAI_main.py - D:\포트폴 소스관리\포트폴 자동탐지 과제\client\PotHoleDetectAI_20191014\PotHoleDetectAI_main.py (3.5.4)
File Edit Format Run Options Window Help

class GPSThread(Thread):
    lat = 0.0 # 위도
    lng = 0.0 # 경도
    alti = 0.0 # 고도
    speed = 0.0 # 속도
    course = 0.0 # 방향
    gps_status = 1 # GPS 상태
    addr = "" # GPS 주소
    valid = False # GPS Valid field for GPBMC, GPGLL

    # DEBUG
    gps_fixstat = 0
    satellites_in_use = 0
    gps_crcfail = 0
    DBG_invalid_count = 0 # MicroPyGPS .valid counter
    DBG_restart_count = 0
    DBG_run_count = 0
    DBG_run_timeout_count = 0
    DBG_addr_count = 0
    DBG_exception_count = 0
    DBG_lastgpsline = ""
    # DEBUG

    def run(self):
        global b_DEBUG_GPSlogfile
        print(">>> GPSThread - run")
        if b_DEBUG_uselogging:
            logging.info(">>> GPSThread - run")

        ser = serial.Serial("/dev/ttyTHS1", baudrate=9600, timeout=5)
        my_gps = MicropyGPS()
        # my_gps = MicropyGPS.MicropyGPS()
        my_gps.coord_format = "dd"

        self.get_gps_addr()

        # Initialize
        GPSThread.DBG_invalid_count = 0
        GPSThread.DBG_restart_count = 0
        GPSThread.DBG_run_count = 0
        GPSThread.DBG_run_timeout_count = 0
        GPSThread.DBG_addr_count = 0
        GPSThread.DBG_exception_count = 0

        while True:
            line = ser.readline().decode('ascii', errors='replace')
            GPSThread.DBG_lastgpsline = line
            # DEBUG
            # if b_DEBUG_GPSlogfile == True:
            #     my_gps.gps_debugprint_enable(True)
            # else:
            #     my_gps.gps_debugprint_enable(False)

            if b_DEBUG_GPSlogfile == True:
                print(">>> GPSThread - " + line)
                if b_DEBUG_uselogging:
                    logging.info(line)

            # CHECK IF RECEIVER RESET
            # CHECK IF GET "$GPRXT,01,01,02,u-blox ag - www.u-blox.com+50":
            if "www.u-blox.com" in line:
                # DEBUG
                msg1 = "-----"
                msg2 = "GPS RESET or RESTARTED"
                msg3 = msg1
                print(msg1)
                print(msg2)
                print(msg3)
                logging.warning(msg1)
                logging.warning(msg2)
```

◦ 1차 탐지 GPS 주기보고

차량의 현재 위치를 나타내기위해 주기적으로 전송하는 GPS 데이터를 수집하기위한 코드이다.

```
PotHoleDetectAI_main.py - D:\소프트웨어\소스관리\소프트웨어 자동탐지 과제\wclient\WPotHoleDetectAI_20191014\WPotHoleDetectAI_main.py (3.5.4)
File Edit Format Run Options Window Help

def Periodic_GPS_Thread(self):
    global PROTO_HW
    global PROTO_SW_VER
    # DEBUG
    #print("<<<< Periodic_GPS_Thread")
    # DEBUG

    # gps data file create
    mac = getmac("wlan0")
    myMac = mac.replace("-", "")
    t_now = datetime.now(timezone('Asia/Seoul'))
    timestamp1 = t_now.strftime('%Y%m%d')
    timestamp2 = t_now.strftime('%H%M%S%f')[:-3]

    data = myMac + "#" + timestamp1 + "#" + timestamp2 + "#" + str(GPSThread.lat) + "#" + str(
        GPSThread.lng) + "#" + str(GPSThread.alti) + "#" + str(GPSThread.course) + "#" + str(GPSThread.speed) + "#" + str(
        GPSThread.gps_status) + "#####" + str(GPSThread.addr) + "###"

    # 20190902 PROTOCOL VER 1.3
    data = myMac + "#" + timestamp1 + "#" + timestamp2 +
        + "#" + str(GPSThread.lat) + "#" + str(GPSThread.lng) + "#" + str(GPSThread.alti) + "#" + str(GPSThread.course) +
        + "#" + str(GPSThread.speed) + "#" + str(GPSThread.gps_status) +
        + "#####" + str(GPSThread.addr) +
        + "#####" + PROTO_HW + "#" + PROTO_SW_VER + "#"

    txt_filename = "v2_" + myMac + "_" + timestamp1 + timestamp2 + ".0.txt"

    # TODO - GPS NEED TO FOLLOW PROTOCOL VER 1.3

    if b_DEBUG_useSaveFile:
        f = open(DetectThread.inSavePath + txt_filename, "w+")
        result = f.write(data)
        f.close()

    self.timerGPS = threading.Timer(float(DetectMain.gpsPeriod), self.Periodic_GPS_Thread)
    self.timerGPS.start()

#
def updateStatusBar(self):
    global DBG_camloop_total_t_us

    # Take Median value
    # Note - Taking a latest value is not a good solution

    np_med = np.nanmedian # OPTIMIZATION - function name

    #
    vg_rt = np_med(log_DBG_VGET_time)
    # vg_fps = 1000.0 / np_med(log_DBG_VGET_time)
    vg_e2e_rt = np_med(log_DBG_VGET_e2e_time)
    # vg_e2e_fps = -1.0 if vg_e2e_rt == 0 else 1000.0 / vg_e2e_rt # prevent divide_by_zero (-1.0 means ERROR)
    # vg_wait = np_med(log_DBG_VGET_wait_time)

    #
    rs_rt = np_med(log_DBG_SRUN_time)
    # rs_fps = 1000.0 / np_med(log_DBG_SRUN_time)
    rs_e2e_rt = np_med(log_DBG_SRUN_e2e_time)
    # rs_e2e_fps = -1.0 if rs_e2e_rt == 0 else 1000.0 / rs_e2e_rt # prevent divide_by_zero (-1.0 means ERROR)
    # rs_wait = np_med(log_DBG_SRUN_wait_time)

    #
    rs_hlget = np_med(log_DBG_SRUN_hlget_time)
    rs_h2put = np_med(log_DBG_SRUN_h2put_time)

    #
    vs_rt = np_med(log_DBG_VSHOW_time)
    # vs_fps = 1000.0 / np_med(log_DBG_VSHOW_time)
    vs_e2e_rt = np_med(log_DBG_VSHOW_e2e_time)
    # vs_e2e_fps = -1.0 if vs_e2e_rt == 0 else 1000.0 / vs_e2e_rt # prevent divide_by_zero (-1.0 means ERROR)
    # vs_wait = np_med(log_DBG_VSHOW_wait_time)

    #
    avg_e2e_rt = (vg_e2e_rt + rs_e2e_rt + vs_e2e_rt) / 3.0
    avg_fps = -1.0 if avg_e2e_rt == 0 else 1000.0 / avg_e2e_rt
    avg_e2e_rt2 = (vg_e2e_rt + rs_e2e_rt) / 2.0 # not consider SHOW
    avg_fps2 = -1.0 if avg_e2e_rt2 == 0 else 1000.0 / avg_e2e_rt2

    #
    avg_camloop_tot = -1.0 if len(log_DBG_camloop_total_t) == 0 else log_DBG_camloop_total_t[-1]

Ln: 5199 Col: 34
```

◦ 2차 탐지 Sender

2차 탐지 서버에 위치하여 1차 DB 및 1차 이미지 저장소로부터 데이터 및 이미지를 가져와서 메시지 큐의 일종인 RabbitMQ를 통해 Receiver에 일감을 분배해주는 역할을 하는 Sender 프로그램에 대한 코드이다.

```
PotHoleDetectAISender.py - D:\wijs\인수인계_박태성\프로젝트\PotHoleDetectAI\Second\PotHoleDetectAISender.py (3.5.4)
File Edit Format Run Options Window Help
#!/usr/bin/env python3
# coding: utf-8

import pika
import pymysql
from time import sleep
from datetime import datetime
import logging
import logging.handlers

# log = logging.getLogger('snowdeer_log')
# log.setLevel(logging.DEBUG)
#
# formatter = logging.Formatter('%(asctime)s %(filename)s: %(message)s\n\r')
# fileHandler = logging.handlers.TimedRotatingFileHandler(filename='sender_log.log', when='midnight', interval=1, encoding='utf-8')
# streamHandler = logging.StreamHandler()
#
# fileHandler.setFormatter(formatter)
# fileHandler.suffix = "%Y%m%d"
# streamHandler.setFormatter(formatter)
#
# log.addHandler(fileHandler)
# log.addHandler(streamHandler)

connection = pika.BlockingConnection(pika.ConnectionParameters('localhost'))
channel = connection.channel()
channel.queue_declare(queue='hello')

while True:
    # MSSQL 접속
    conn = pymysql.connect(server='pothole02-sql.database.windows.net', port=1433, user='pdis@pothole02-sql', password='Pothole2018!', database='POTHOLE')

    # Connection 으로부터 Cursor 생성
    cursor = conn.cursor()
    cursor_for_update = conn.cursor()

    # SQL문 실행
    cursor.execute("SELECT TOP 25 * FROM POTHOLE_DETECT_INFO WHERE PROC_FLAG = '0' AND POTHOLE_ORG_IMAGE_FILE_NM IS NOT NULL AND POTHOLE_IMAGE_FILE_NM")
    # cursor.execute("SELECT TOP 25 * FROM POTHOLE_DETECT_INFO WHERE GRP_KEY like '20190%' And POTHOLE_ORG_IMAGE_FILE_NM IS NOT NULL AND POTHOLE_IMAGE_FI")

    # 데이터 하나씩 Fetch하여 출력
    row = cursor.fetchone()

    arr_grp_key = []
    arr_macdrs = []
    arr_detectde = []
    arr_detecttm = []

    if row:
        while row:
            print("row" + str(row))

            try:
                arr_macdrs.append(row[0])
                arr_grp_key.append(row[1])
                arr_detectde.append(row[2])
                arr_detecttm.append(row[3])
                # log.info(row[1] + ", MAC= " + row[0])

                # send GRP_KEY, MACDRS, DETCT_DE, DETCT_TM
                body = row[1] + "#" + row[0] + "#" + row[2] + "#" + row[3] + "#" + row[4] + "#" + row[5] + "#" + row[6] + "#" + row[7] + "#" + row[8] + "#" + row[9] + "#" + row[10] + "#" + row[11] + "#" + row[12] + "#" + row[13] + "#" + row[14] + "#" + row[15] + "#" + row[16] + "#" + str(row[17]) + "#" + str(row[18]) + "#" + str(row[19]) + "#" + str(row[20]) + "#" + str(row[21]) + "#" + str(row[22]) + "#"
                channel.basic_publish(exchange='', routing_key='hello', body=body)
                print(body)

                row = cursor.fetchone()
            except:
                pass
```

◦ 2차 탐지 Receiver

여러 개의 프로세스로 구성될 수 있으며, 각각의 Receiver는 RabbitMQ로부터 일감을 할당받아 2차 탐지 알고리즘을 적용하여 1차에서 탐지된 포트홀을 좀 더 정확하게 필터링하여 2차 DB 및 2차 이미지 저장소에 전달하는 역할을 수행하는 Receiver 프로그램에 대한 코드이다.

```
PotHoleDetectAIReceiver_long_task.py - D:\wjjs\윈수인계_박태성\윈수인계\PotHoleDetectAISecond\PotHoleDetectAIReceiver_long_task.py (3.5.4)
File Edit Format Run Options Window Help

def callback(body):
    print("*****Callback() Start*****")
    strttime1 = datetime.now()
    print("[x] Received %r" % body)
    str_body = body.decode("utf-8") # 프로토콜 패킷 디코딩
    row = str_body.split("#")

    str_grp_key = row[0]
    str_macadr = row[1]
    str_detct_de = row[2]
    str_detct_tm = row[3]
    log3.info("*****"+str(row[0])+"/"+str(row[1])+" start *****")
    try:
        # src_base_path
        src_base_path = "/mnt/pothole02/pothole/download/"

        # src_orig_path
        src_orig_path = src_base_path + str_macadr + "/" + str_detct_de + "/orig/"
        src_orig_fullpath = src_orig_path + str_detct_tm + ".jpg"

        # src_first_path
        src_first_path = src_base_path + str_macadr + "/" + str_detct_de + "/"
        src_first_fullpath = src_first_path + str_detct_tm + ".jpg"

        # Local Path
        # create orig save directory
        orig_path = base_path + str_macadr + "/" + str_detct_de + "/" + "orig/"
        orig_fullpath = orig_path + str_detct_tm + ".jpg"
        strttime2 = datetime.now()
        make_directory(orig_path)
        endtime2 = datetime.now()
        log3.info("make_directory 소요시간 = " + str(endtime2 - strttime2))

        # create first save directory
        first_path = base_path + str_macadr + "/" + str_detct_de + "/" + "first/"
        first_fullpath = first_path + str_detct_tm + ".jpg"
        make_directory(first_path)

        # create second save directory
        second_path = base_path + str_macadr + "/" + str_detct_de + "/" + "second/"
        second_fullpath = second_path + str_detct_tm + ".jpg"
        make_directory(second_path)

        # create second None Pothole save directory
        none_second_path = base_path + str_macadr + "/" + str_detct_de + "/" + "second/none/"
        none_second_fullpath = none_second_path + str_detct_tm + ".jpg"
        make_directory(none_second_path)

        # create second Pothole mask images save directory
        mask_second_path = base_path + str_macadr + "/" + str_detct_de + "/" + "mask/"
        mask_second_fullpath = mask_second_path + str_detct_tm + ".jpg"
        make_directory(mask_second_path)

        # create second save path in remote server
        second_path2 = src_base_path + str_macadr + "/" + str_detct_de + "/" + "second/"
        second_fullpath2 = second_path2 + str_detct_tm + ".jpg"

        # setting FTP to first_DB_Server
        cnopts = pysftp.CnOpts()
        cnopts.hostkeys = None
        strttime = datetime.now()

        sftp = pysftp.Connection('52.141.38.48', username='azadmin', password='QWERasdfl123!', cnopts=cnopts) #gy_app2 storage
        isFileExist = True

        # get file orig
        get_file_ftp(sftp, src_orig_fullpath, orig_fullpath)

        # get file first detection result
        get_file_ftp(sftp, src_first_fullpath, first_fullpath)
```

6. 딥러닝 최적화 자료 및 고도화 자료

Potholes threatening road safety

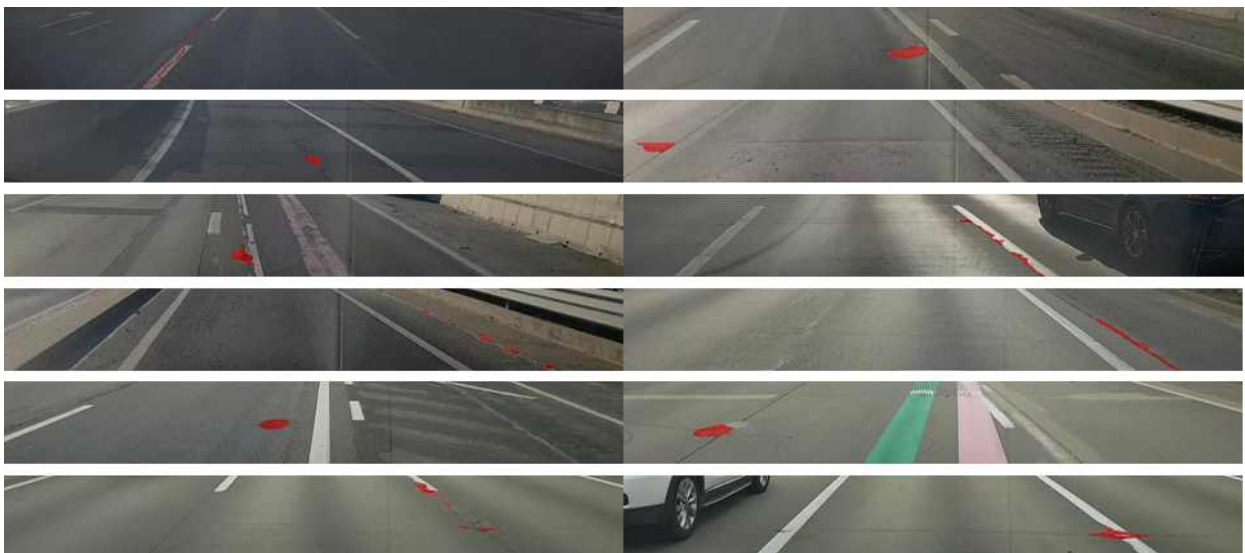
- 94,746 potholes in the last 5 years, costing about \$11.2 million
*Ministry of Land Infrastructure and Transportation in Korea
- Potholes increased from 6,400 to 25,600 in 2012 and 2013
*Reports of Seoul city
- 4,223 car accidents caused by potholes in the last 5 years
*Ministry of Land Infrastructure and Transportation in Korea

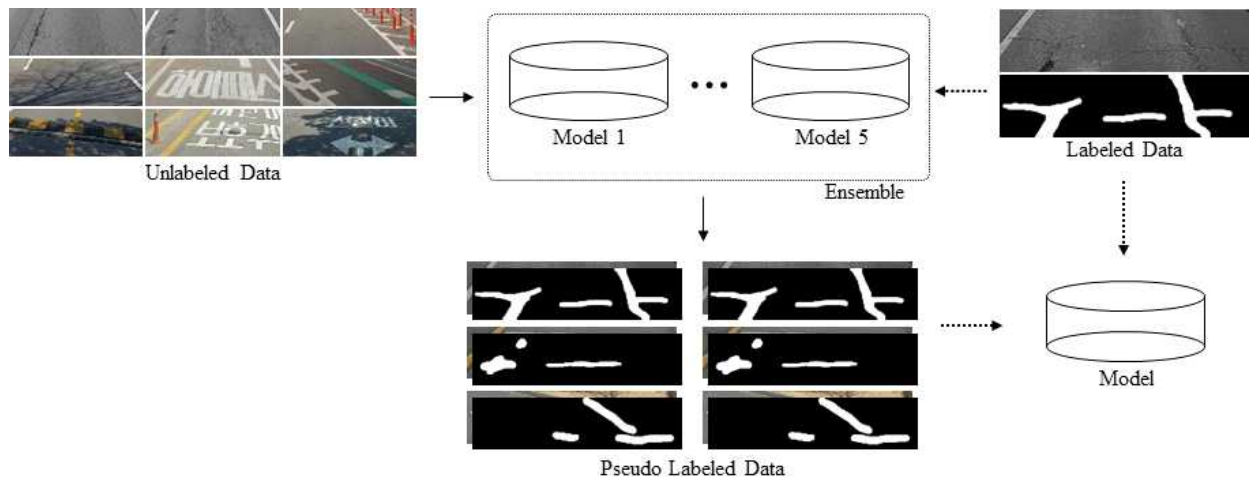


□ 이미지 데이터 셋

- 30,000 장 탐지 후 탐지율 알고리즘 판독
- 정탐 : 12,000 장 / 오탐 : 18,000 장
- Each image includes at least one object.

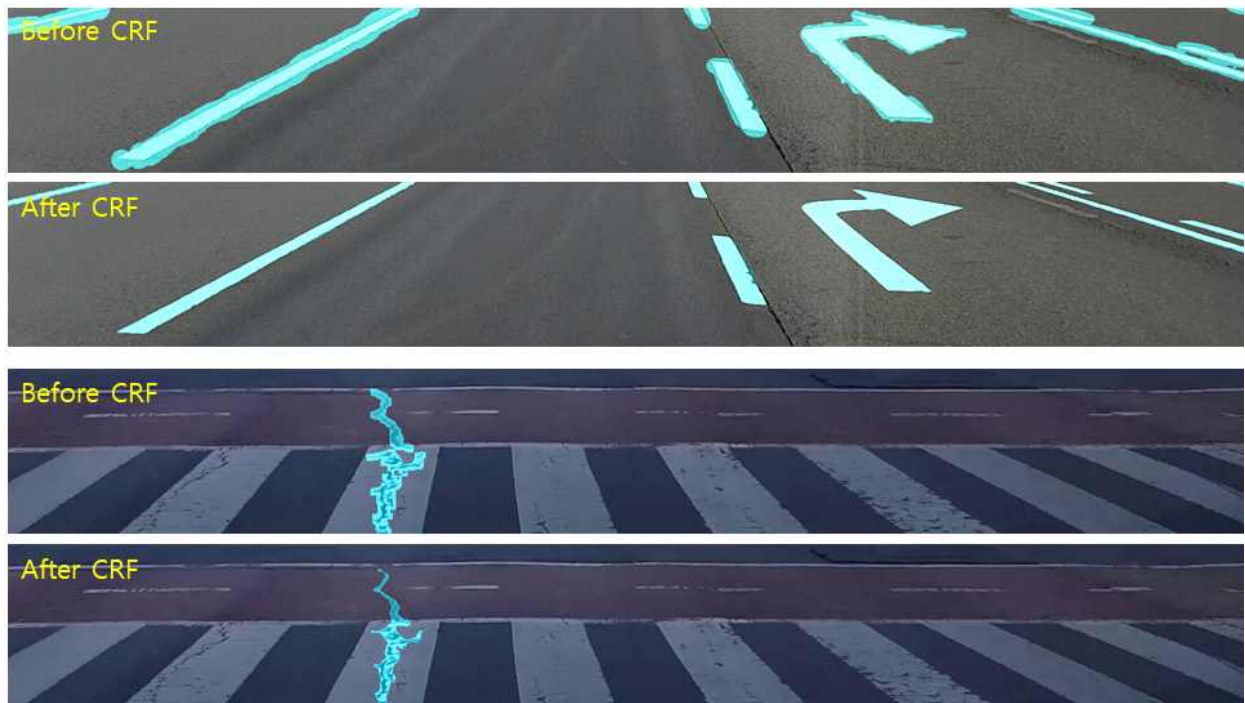
(Road marking / Facilities / Grooving / Shadow / Vehicle / **Potholes**)





□ contrast rendering factor ; CRF

○ CRF Result



Subjective Evaluation (450 images)

	Precision	Recall	Accuracy	F1-score
Supervised	0.7012	0.9264	0.8728	0.7982
Semi-supervised	0.9014	0.8822	0.9387	0.8917

Objective Evaluation (1,980 images)

	Semi-supervised
mIOU	0.7447
mAP ₅₀	0.9091

시험성적서

한국정보통신기술협회
소규모특허시험인증연구소
주소: 경기도 성남시 분당구 영정동 47
전화: 021-024-0276 Fax: 021-024-0188

공시번호: RT-A-00-0124

TTA

1. 시험 목적
- 특허시험인증(특) 한국정보통신기술협회
- 품 호: 021-024-0276
- 시험일자: 2020. 03. 27

2. 시험대상품명: 반도체
3. 품 호: 021-024-0276
4. 품 호: V1.0

5. 시험기준: 2020. 03. 27

6. 시험방법: 시험결과서 참조

7. 시험결과: 시험결과서 참조

8. 시험결과: 시험결과서(RT-A-00-0124) 참조

부속
부속
부속

2020년 3월 30일

한국정보통신기술협회 회장

TTA-0077-1(007)

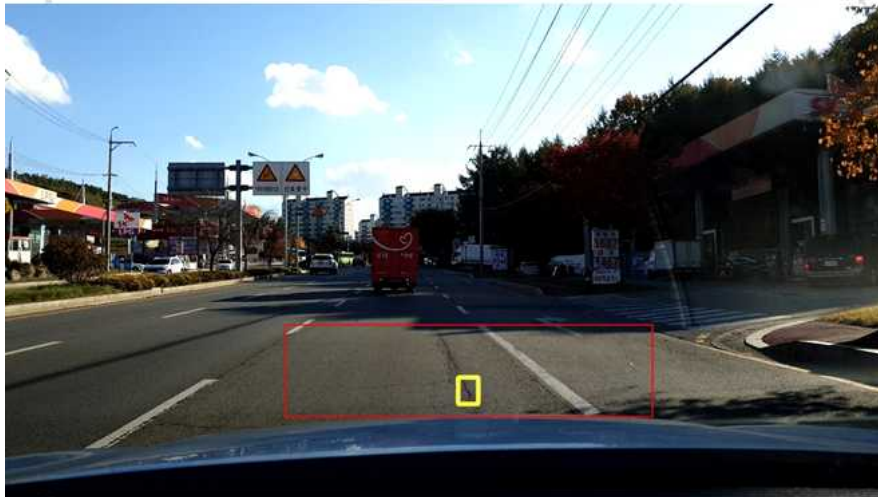
5. 시험결과 요약

본 시험의 시험결과 결과 O(합격) 시험결과에 대한 책임은 모두 인증대상품에 속한다(O).

ID	시험항목	시험내용	목표치	측정치	판정
TC1	도면능력 표준치 측정	도면능력 (도면능력) 도면능력 표준치 측정	mIOU 0.7 0.7	mIOU 0.74	합격
TC2	도면능력 표준치 측정	도면능력 (도면능력) 도면능력 표준치 측정	표준치 90% 0.7	측정치 91% 0.7	합격

7. 도로 파손 조사 매뉴얼

□ 도로 노면 파손 탐지



설명

올바른 설치 예시

설치시 화면 중앙과 ROI 영역이 최대한 “도로 노면”을 향하게 합니다.

- ① ROI 영역이 본넷의 바로 위에 위치하도록 설치합니다.
- ② ROI 영역에 대시보드 반사 이미지가 들어오지 않도록 설치합니다.

□ 권장 운행 조건



설명

운행 조건

운행시 노면의 탐지가 원활한 기상조건, 시간대에 운행을 권장합니다.

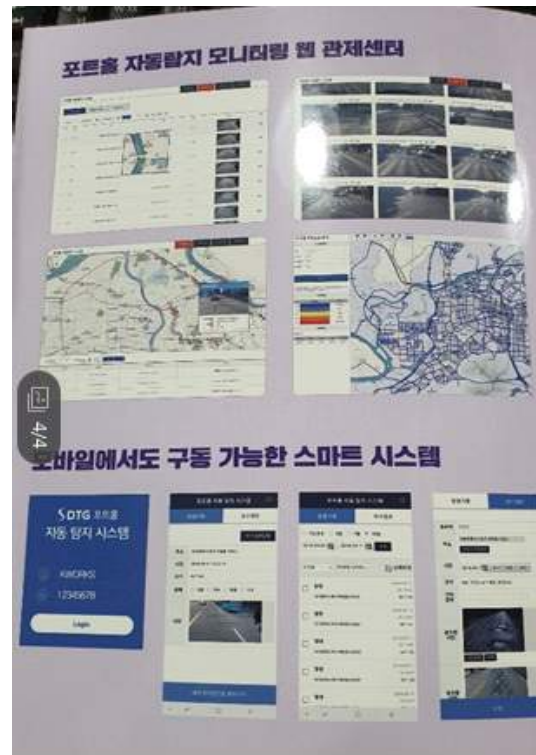
- ① 주행중 적절한 차간 간격을 유지하십시오 - ROI 영역에 노면이 보여야 함
- ② 적절한 기상 조건, 시간대에 운행하십시오.
부적절한 조건 예시)
 - 강설, 우천 및 우천 직후 등
 - 야간, 터널 내부 등
 - 이른 아침, 늦은 오후, 흐린 날 등에는 촬영이 원활하지 않을 수 있습니다.

8. 서비스 사업화

◦ 리플렛 발송 이력 : 517건

- 2019. 3. 26 : 204건 / 5. 21 : 38건 / 9. 10 : 163건 / 9. 27 : 112건

◦ 1차 리플렛 제작

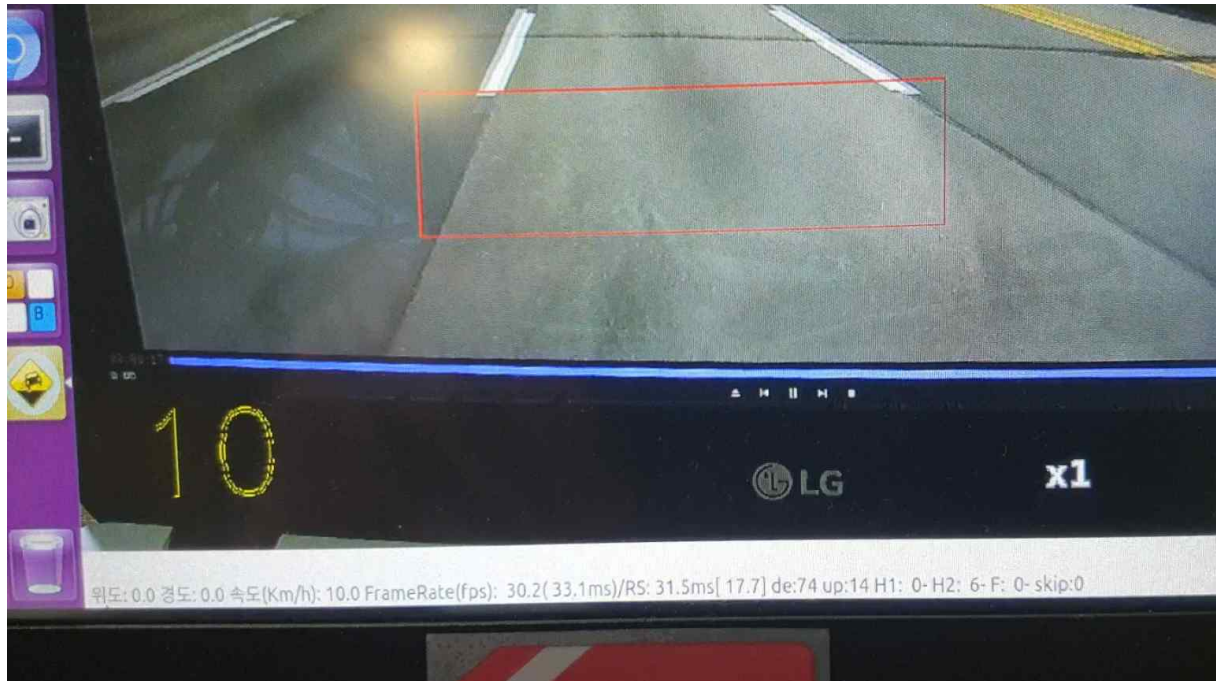


- 2차 리플렛 제작

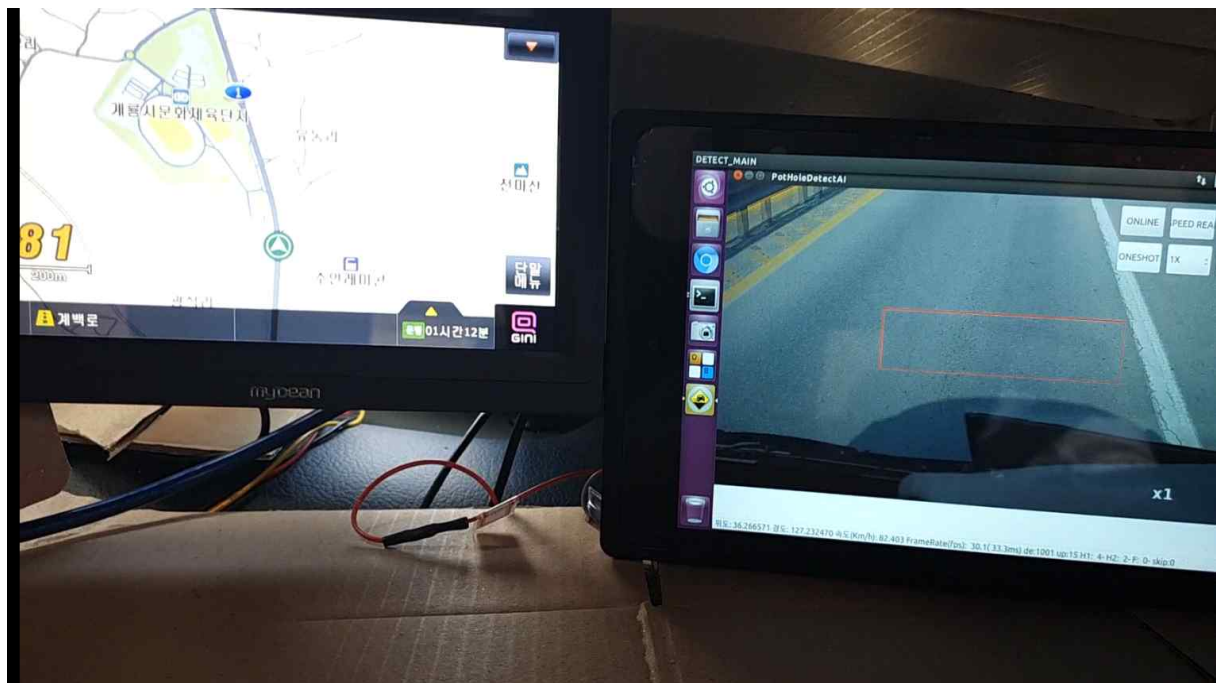


6. 수집 단말기 성능

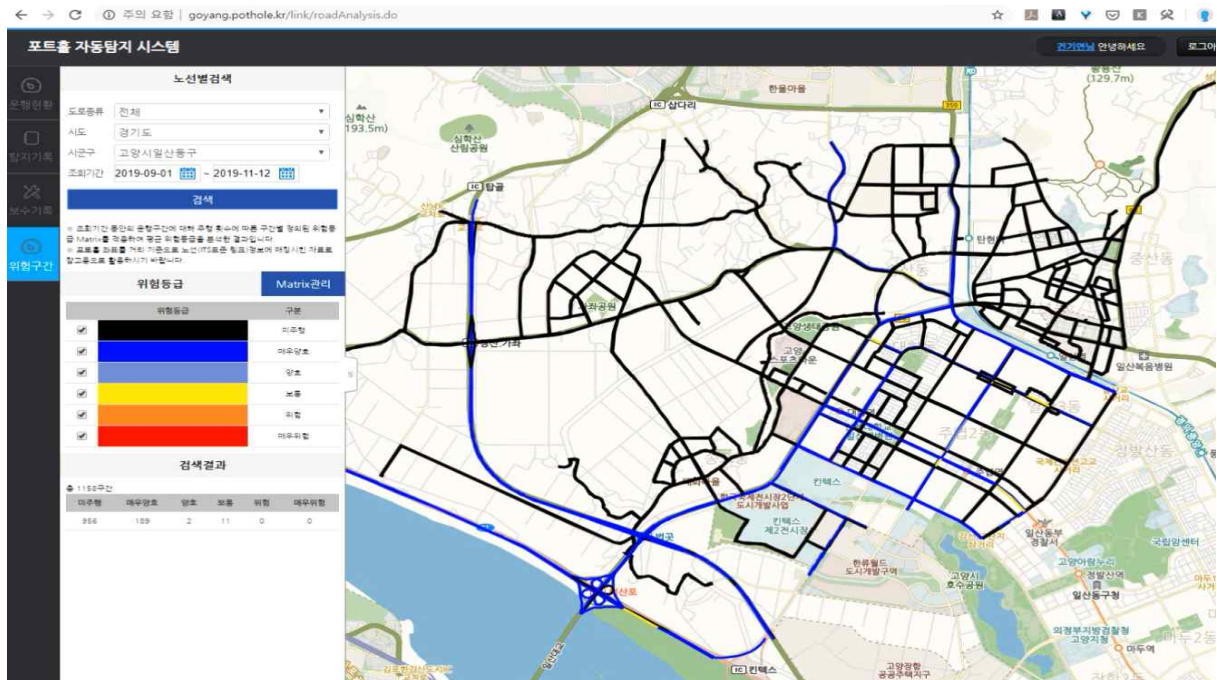
◦ 측정가능속도



◦ 연산속도



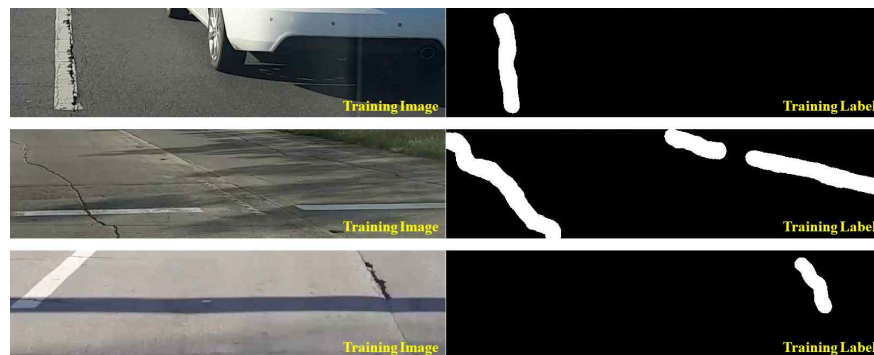
◦ 위험도 구간평가 결과



7. 포트홀 인식을 및 성능

◦ 데이터셋

- 자동 노면파손 정보 판별 기술에 대한 신경망 모델을 학습하기 위하여 실제 국내 도로를 주행하면서 차량에 설치된 카메라를 통하여 학습 데이터셋을 수집함
- [1920 x 1080] 해상도로 스마트폰을 활용하여 촬영되었고, 차량 내에 블랙박스가 설치될만한 위치에 스마트폰을 거치하였으며, 총 5,000장의 이미지에 대하여 분할 작업을 수행함



<수집된 학습 이미지 데이터셋 예시>

- 5,000장의 이미지에서 도로노면 파손이 존재하는 이미지는 약 40%이며, 그 외의 이미지는 포트홀이라고 오인식 할 수 있을만한 차량, 그림자, 로드마킹, 건물 등의 이미

지를 포함함

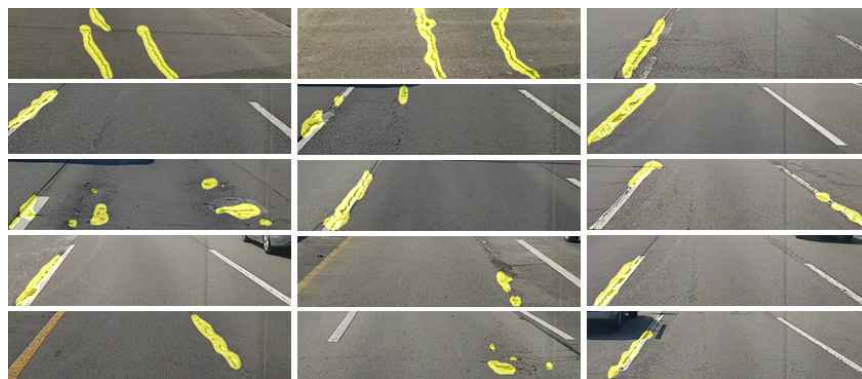
- 밝기를 총 6단계로 조정함으로써 그림자 혹은 시간에 따라 생기게 되는 어두운 환경에서 보다 강건하게 동작할 수 있도록 증강 작업을 수행함



<데이터셋 밝기에 따른 증강을 위한 작업>

◦ 성능 결과

- 도로노면의 파손이 있는 부분만을 이미지에서 분할하는 것을 확인하기 위하여, 주관적인 성능평가를 수행함
- 노면을 촬영하는 중에 ROI(Region of Interest)에 들어가는 이미지를 500장을 선정하고, 총 4인 평가자가 주관적으로 평가하였으며, 여기에서 500장의 이미지 순서는 임의적으로 나타나게 진행함



<노면파손 정보 수집 및 판별 예시>

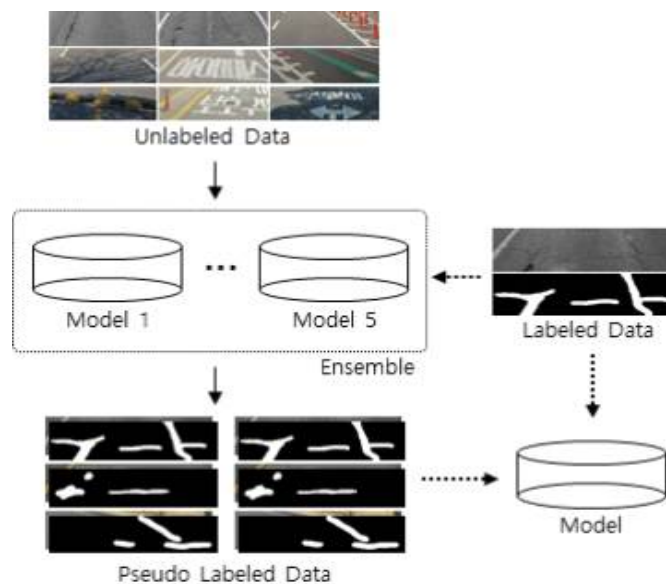
- 개발된 노면파손 판별 기술의 평균 민감도는 0.9264정도의 높은 수치를 나타냄. 정밀도의 경우에는 Positive 예측 수에서 실제 Positive 비율임
- 개발된 노면파손 판별 기술의 평균 정밀도는 0.7012으로 민감도에 비하면 상대적으로 다소 낮은 수치를 나타냄. 또한, 모델의 정확도를 표현할 때는 정확도 수치를 정밀도와 민감도를 하나의 수치로 총체적으로 나타낼 경우에는 F1-score로 나타낼 수 있는데, 개발된 노면파손 판별 기술의 경우 평균 정확도와 F1-score는 각각 0.8728 및

0.7982의 수치로 평가됨. 이는 개발된 노면파손 판별 기술의 성능이 정탐에 대해서는 강건한 성능을 보이고 있는 것으로 해석이 가능함

<1차 성능평가 결과>

	Tp	Tn	Fp	Fn	정밀도	민감도	정확도	F1-score
Expert I	112	281	47	10	0.7044	0.9180	0.8733	0.7972
Expert II	105	263	77	5	0.5769	0.9545	0.8178	0.7192
Expert III	125	263	51	11	0.7102	0.9191	0.8622	0.8013
Expert IV	111	311	18	10	0.8605	0.9174	0.9378	0.8880
Total	453	1118	193	36	0.7012	0.9264	0.8728	0.7982

- 노면파손 판별 기술의 성능 개선 : Semi-supervised learning
- 아래의 그림은 도로노면 파손을 탐지하기 위하여 본 논문에서 제안하는 수도 레이블을 활용한 준지도 학습 방법에 대한 블록 다이어그램을 가리킴. 먼저, 레이블된 데이터를 활용하여 합성곱 신경망 기반의 도로노면의 파손 탐지를 수행할 수 있는 모델을 학습함
- 5,000장의 레이블된 데이터셋을 활용하여 학습된 모델들의 결과로부터 또 다른 5,000장의 레이블되지 않은 이미지 데이터셋을 입력으로 넣어서 예측된 출력 이미지 데이터셋을 획득함. 이렇게 획득한 신경망 모델이 수도 레이블한 이미지 데이터셋으로 간주하고, 총 10,000장의 이미지 데이터셋 통하여 재차 신경망 모델을 학습하는 형태로 계속하여 진행



<노면파손 정보수집/판별 기술의 성능개선을 위한 준지도 학습>

- 정성적인 성능을 비교하기 위하여 마찬가지로 성능평가를 수행함. 1차 버전과 동일한 방식과 동일한 평가자가 동일한 방식으로 성능평가를 수행함. 먼저, 도로노면을 촬영

하는 도중에 ROI(Region of Interest) 안에 들어올 수 있는 이미지를 450장을 선정하였고, 여기서, 학습에 활용된 데이터셋과 겹치지 않는 이미지 데이터만을 활용함. 아래의 그림은 평가에 활용되는 데이터셋의 예시 사진을 나타냄

· 도로노면의 파손 영역이 포함된 이미지만 있는 것이 아니라 도로노면을 스캔하면서 ROI 안에 들어올 수 있는 로드 마킹, 이음선, 줄눈, 그림자, 자동차 등을 포함함



<정성적인 성능평가를 위한 이미지 데이터셋 예시>

· 총 4인 평가자가 주관적으로 450장 이미지를 평가하였으며 이미지 순서는 임의적으로 나타나게 진행함. 평가자는 이미지 한장 확인후 가장 근접하다고 판단되는 항목 평가함. 이미지 상에서 도로노면 파손 있는 경우에 제안된 모델이 도로노면 파손 부분만을 정분할한 경우와 그렇지 못한 경우, 이미지 상에서 도로노면 파손 부분이 없는 경우에 제대로 도로노면 파손이라고 분할하지 않은 경우와 그렇지 못한 경우로 나뉘게 됨

· 먼저, 정확도 측면에서는 기존 모델의 0.8728에서 제안된 모델이 0.9387로 향상된 것을 확인 가능함. 또한, 기존 모델의 경우 정밀도의 수치가 0.7012였지만, 제안된 모델의 경우에는 0.9014로 크게 향상됨. 이는 오탐에 대한 탐지 성능이 크게 개선되었다고 볼 수 있음. Fp의 수치를 보더라도 기존에는 총 193개의 Fp가 존재했지만, 제안된 모델에서는 50으로 감소된 것을 확인 가능함. 반면에, 민감도 부분에서는 기존의 0.9264의 수치였지만, 0.8822로 감소된 것을 확인 가능함.

· 이는 도로노면 파손에 대한 탐지가 민감도가 감소된 것으로 Fn가 밀접한 관련이 있는데, 실질적으로 Fn 수치가 기존에는 총 36개 존재하였는데, 제안된 모델에서는 61개로 증가함. 최종적으로 F1-score를 비교해보면 0.7982에서 0.8917로 향상된 것을 확인 가능함. 이는 전반적으로 기존 모델에 비하여 준지도 학습을 통하여 얻은 모델이 큰 성능 개선이 되었다고 평가될 수 있음

<최종 성능평가 결과>

구분	Tp	Tn	Fp	Fn	정밀도	민감도	정확도	F1-score
Expert I	119	308	13	10	0.9015	0.9225	0.9489	0.9119
Expert II	111	309	14	16	0.8880	0.8740	0.9333	0.8810
Expert III	119	313	13	15	0.9015	0.8881	0.9391	0.8947
Expert IV	108	312	10	20	0.9153	0.8438	0.9333	0.8781
Total	457	1242	50	61	0.9014	0.8822	0.9387	0.8917

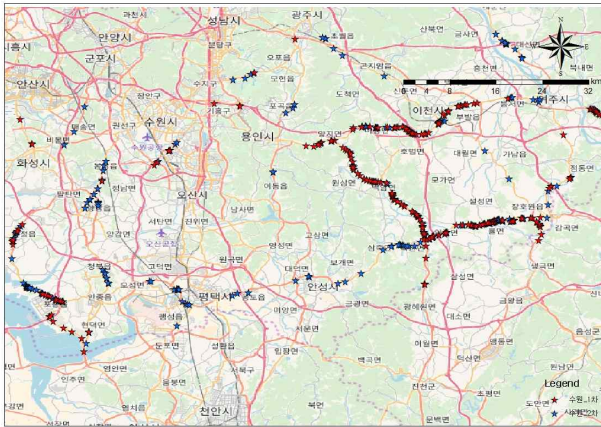
8. 단위 구간당 포트홀 개수 분석 및 우선 보수 구간 선정

◦ 데이터 개요

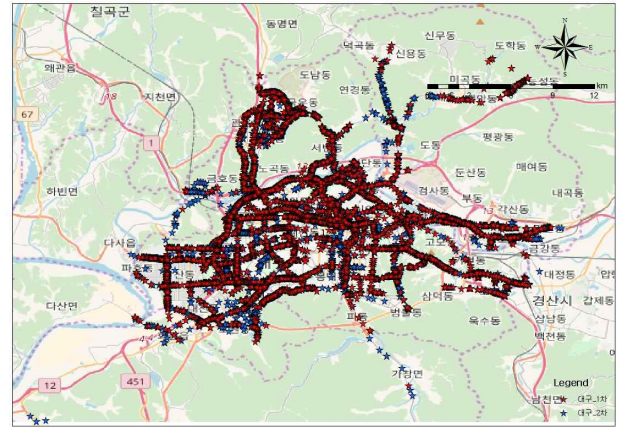
- 수원의 경우 수집범위, 기간등을 고려하여 1,2차 송부된 포트홀 데이터를 전부 활용하여 분석수행.
- 대구의 경우 수집기간의 연속성, 수집범위의 동일성등을 고려하여 2차송부된 데이터에 1차 데이터가 포함된 것으로 간주하여 1차 송부된 데이터만을 고려하여 분석수행.

전체 데이터 개요

구분		포트홀 발생건수	수집기간
수원	1차 송부	345	18.05.01~18.07.17
	2차 송부	412	17.04.01~17.04.30
	총 계	757	-
대구	1차 송부	12,107	18.05.01~18.07.17
	2차 송부	9,165	18.02.01~18.04.30
	총 계	21,272	-



(a) 수원



(b) 대구

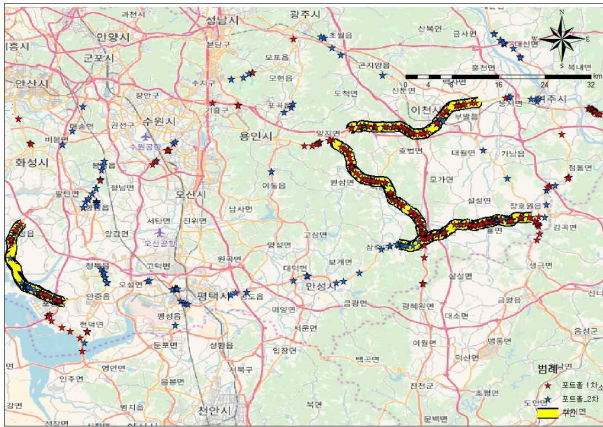
포트홀 분포현황

◦ 데이터 분석 현황

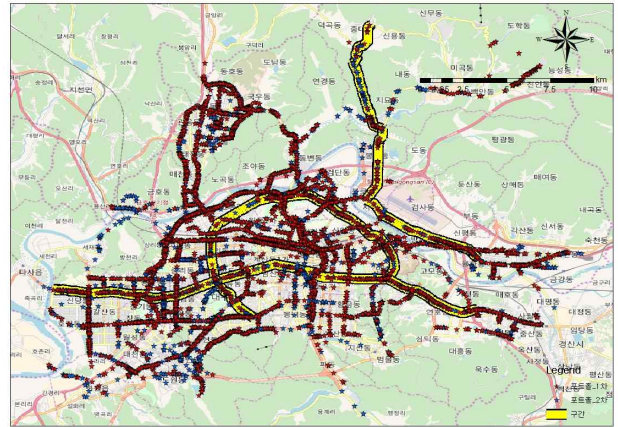
- 포트홀 개수를 기준으로 수원 62.0%, 대구 27.6%의 데이터에 대한 분석수행.
- 수원시의 경우 18년 5월 1일부터 7월 17일까지의 데이터(1차: 345건)와 17년 4월 1일부터 4월 30일까지의 데이터(2차: 412건) 중 지역별로 양방향 20km 범위의 포트홀을 추출하여 분석에 활용
- 대상구간을 100m, 500m, 1000m 단위로 구분하여 각 구간별 포트홀 개수 체크 및 분포도 확인.

데이터 분석 진행현황

구분	포트홀 분석 현황 (개수)			분석연장 (왕복, km)	분석구간수 (100m기준)
	분석 (A)	전체 (B)	비율 (A/B*100)		
수원	469	757	62.0%	162.9	1,629
대구	3,347	12,107	27.6%	111.8	1,118
전체	3,816	12,864	29.7%	274.7	2,747



(수원)구간분류 작업현황도



(대구)구간분류 작업현황도

◦ 단위구간당 포트홀 개수

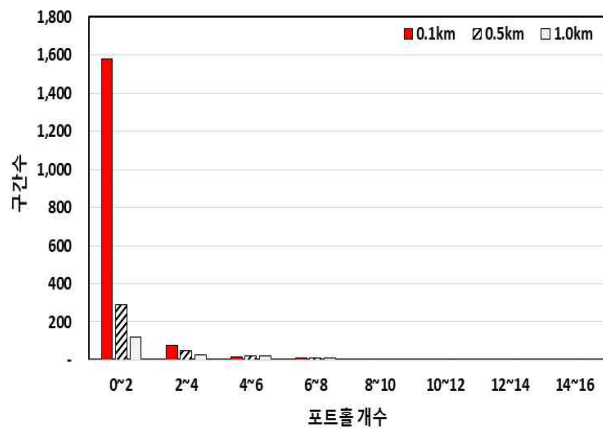
[수원 분석결과]

- 일반국도 17,38,42,77,82호선이 통과하는 왕복 4차로의 총 연장 162.9km의 구간.
- 구간길이별 평균 포트홀 개수는 0.3~2.2개. 구간길이별 최대 포트홀 개수는 15~24개로 나타남.

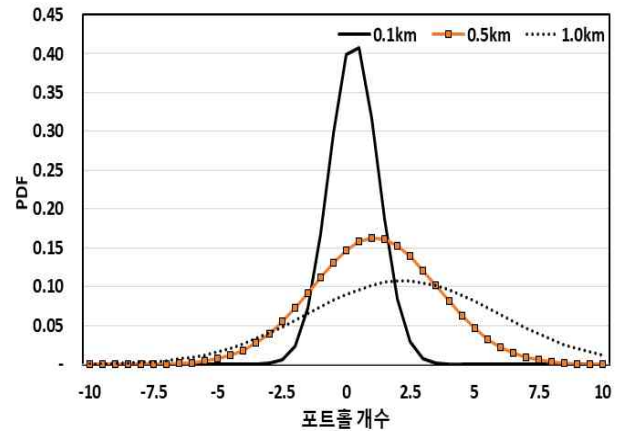
(수원) 구간당 포트홀 분석결과

구분		구간길이					
		0.1km		0.5km		1.0km	
전체 구간수		1,629		332		166	
포트홀 구간수		255		120		81	
		미포함 ¹⁾	포함 ²⁾	미포함 ¹⁾	포함 ²⁾	미포함 ¹⁾	포함 ²⁾
구간당	평균 포트홀 (개)	1.8	0.3	3.1	1.1	4.5	2.2
	표준편차	1.7	1.0	3.3	2.4	4.2	3.7
	최소 포트홀 (개)	1.0	0.0	1.0	0.0	1.0	0.0
	최대 포트홀 (개)	15.0	15.0	24.0	24.0	24.0	24.0

1) 포트홀 미발생구간을 미포함한 분석, 2) 포트홀 미발생구간을 포함한 분석.



(a) 히스토그램



(b) PDF

(수원) 구간당 포트홀 개수 분포도.

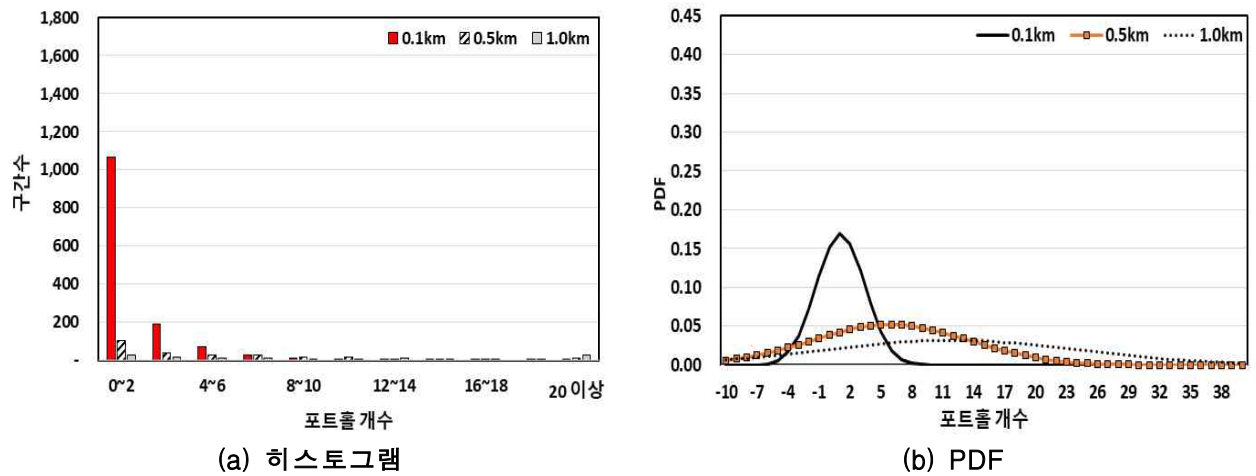
[대구 분석결과]

- 공항로, 달구벌대로, 두류공원로, 안심로등을 통과하는 왕복 6~10차로의 총 연장 162.9km의 구간.
- 구간길이별 평균 포트홀 개수는 1.1~11.8개. 구간길이별 최대 포트홀 개수는 29~72개로 나타남.

(대구) 구간당 포트홀 분석결과

구분		구간길이					
		0.1km		0.5km		1.0km	
전체 구간수		1,118		230		116	
포트홀 구간수		325		164		98	
		미포함 ¹⁾	포함 ²⁾	미포함 ¹⁾	포함 ²⁾	미포함 ¹⁾	포함 ²⁾
구 간 당	평균 포트홀 (개)	3.0	1.1	8.4	6.0	14.0	11.8
	표준편차	3.1	2.4	7.9	7.7	12.4	12.5
	최소 포트홀 (개)	1.0	0.0	1.0	0.0	1.0	0.0
	최대 포트홀 (개)	29.0	29.0	53.0	53.0	72.0	72.0

1) 포트홀 미발생구간을 미포함한 분석, 2) 포트홀 미발생구간을 포함한 분석.



(대구) 구간당 포트홀 개수 분포도.

◦ 단위구간당 포트홀 크기

[수원, 대구]

- 각 지역별 구간당 발생한 포트홀의 평균 크기($w \times h$, 픽셀단위)에 대한 기초통계는 아래 표와 같음.
- 픽셀 크기를 기준으로 단위구간당 포트홀의 평균 크기는 수원에 비해 대구가 상대적으로 더 큰 것으로 나타남.

구간당 평균 포트홀 크기

구분	구간길이					
	0.1km		0.5km		1.0km	
	수원	대구	수원	대구	수원	대구
평균 크기 (pixels)	2,666	3,824	2,776	3,472	2,833	3,462
표준편차	157	139	226	155	234	166
최소 (pixels)	700	616	704	739	704	1,152
최대 (pixels)	25,821	23,820	16,700	12,811	14,448	9,382

2. 포트홀 기반 리스크 매트릭스(Risk matrix) 모델

◦ 구간당 포트홀 개수 및 크기에 대한 등급화

- 중심극한정리(Central Limit Theorem)이론을 바탕으로 포트홀 개수와 포트홀 크기에 대한 등급화를 수행. 결과는 아래 표와 같음.
- 수원(일반국도)의 경우 Level 1에 해당하는 단위구간당 포트홀 개수는 1.63개 이하로 나타났으나 대구(시내부도로)의 경우는 9.50개 이하로 나타나 상대적으로 시내부 도로에서 포트홀이 빈번하게 발생하는 것으로 나타남.

구간당 포트홀 개수에 의한 등급화

구분	구분 범위 (개/km)					
	수원			대구		
	0.1km	0.5km	1.0km	0.1km	0.5km	1.0km
level 1	0.24개 이하	0.84개 이하	1.63개 이하	0.96개 이하	4.95개 이하	9.50개 이하
level 2	0.24~0.26	0.84~0.98	1.63~1.92	0.96~1.02	4.95~5.45	9.50~10.65
level 3	0.26~0.31	0.98~1.25	1.92~2.50	1.02~1.16	5.45~6.46	10.65~12.97
level 4	0.31~0.34	1.25~1.38	2.50~2.78	1.16~1.22	6.46~6.97	12.97~14.12
level 5	0.34개 이상	1.38개 이상	2.78개 이상	1.22개 이상	6.97개 이상	14.12개 이상

- 포트홀 크기를 기준으로 살펴볼 경우 수원(일반국도)의 경우 Level 1에 해당하는 단위 구간당 평균 포트홀 크기는 2.36천픽셀, 대구(시내부도로)는 3.13천픽셀로 상대적으로 도시부도로에서의 포트홀 크기가 큰 것으로 나타남.

구간당 평균 포트홀 크기에 의한 등급화

구분	평균 크기 (1,000pixels/개)					
	수원			대구		
	0.1km	0.5km	1.0km	0.1km	0.5km	1.0km
level 1	2.35 이하	2.33 이하	2.36 이하	3.55 이하	3.16 이하	3.13 이하
level 2	2.35~2.51	2.33~2.55	2.36~2.60	3.55~3.68	3.16~3.32	3.13~3.30
level 3	2.51~2.82	2.55~3.00	2.60~3.07	3.68~3.96	3.32~3.63	3.30~3.63
level 4	2.82~2.98	3.00~3.23	3.07~3.30	3.96~4.10	3.63~3.78	3.63~3.79
level 5	2.98 이상	3.23 이상	3.30 이상	4.10 이상	3.78 이상	3.79 이상

◦ 리스크 매트릭스 모델 개발

- 포트홀 개수와 포트홀 크기를 발생가능성(Likelihood)과 영향(Impact)으로 구분
- 1km를 단위구간으로 하는 경우의 수원과 대구의 도로에 대한 위험도 매트릭스(Risk Matrix)를 아래 표와 같이 작성.

포트홀에 대한 위험도 등급(안) - 수원(일반국도)

구분		포트홀 개수 (개/km)				
		1.6개 이하	1.6~1.9	1.9~2.5	2.5~2.8	2.8개 이상
평균 포트홀 크기 (1,000pixels/개)	2.4 이하	매우양호	매우양호	양호	양호	보통
	2.4~2.6	매우양호	양호	양호	보통	위험
	2.6~3.1	양호	양호	보통	위험	위험
	3.1~3.3	양호	보통	위험	위험	매우위험
	3.3 이상	보통	위험	위험	매우위험	매우위험

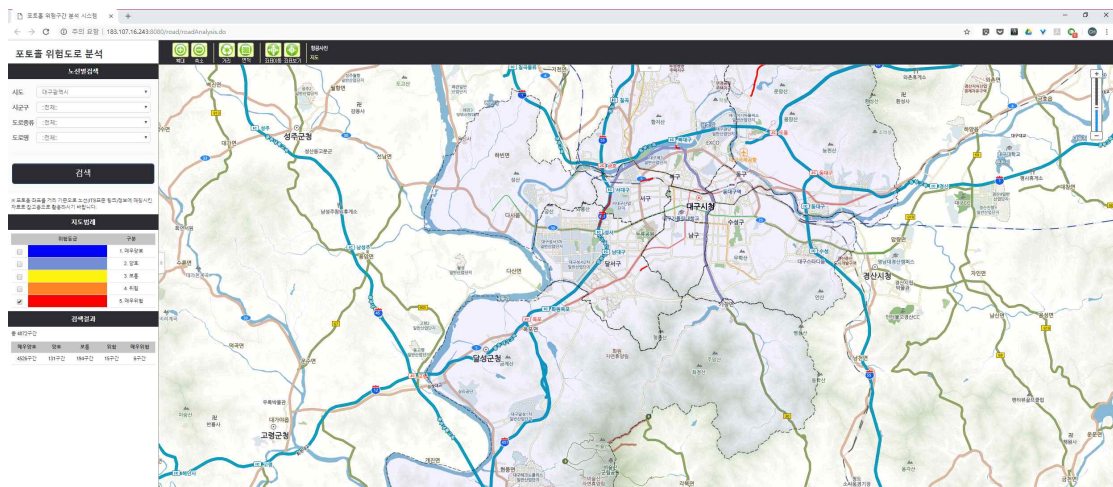
포트홀에 대한 위험도 등급(안) - 대구(시내부도로)

구분		포트홀 개수 (개/km)				
		9.5개 이하	9.5~10.7	10.7~13.0	13.0~14.1	14.1개 이상
평균 포트홀 크기 (1,000pixels/개)	3.1 이하	매우양호	매우양호	양호	양호	보통
	3.1~3.3	매우양호	양호	양호	보통	위험
	3.3~3.6	양호	양호	보통	위험	위험
	3.6~3.8	양호	보통	위험	위험	매우위험
	3.8 이상	보통	위험	위험	매우위험	매우위험

9. 포트홀 위치 정보 기반 위험도 가시화 UI (예시)

◦ 대구시 구간별 위험 도로 표출 UI

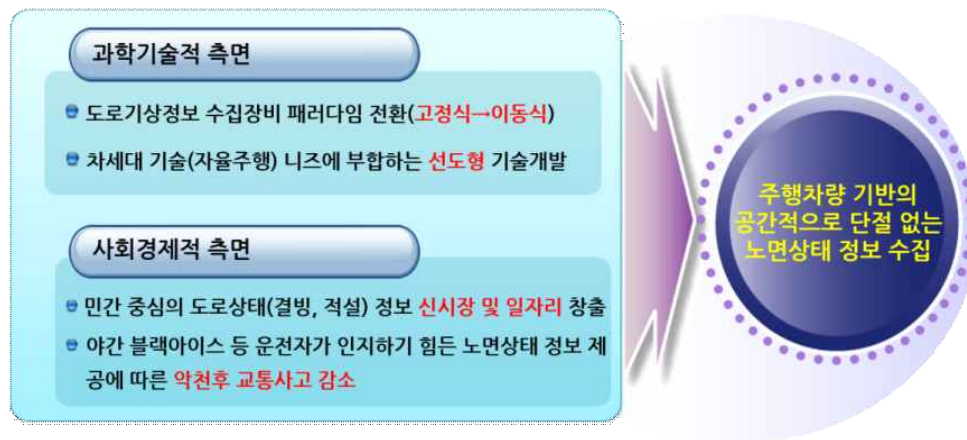
- 5등급으로 구분하여 매우 양호, 양호, 보통, 위험, 매우 위험 순으로 표시를 하며 해당 색상을 도로에 표출함
- 도로 전체 구간 중 등급별 해당 영역 구간에 대한 분석 전체 도로 위험 수위 파악 가능



도로 위험구간 분석 시스템

(1) 공공적 성과

- 본 기술개발 적용을 통해 민간에서 기 운영 중인 화물차량 스마트 차량 관제시스템을 이용하여 위험한 노면상태(결빙, 적설)를 수집할 경우, 수집된 정보는 스마트폰 기반의 내비게이션 업체 등에게 연계(판매) 되어 현재 실시간 소통정보 위주의 카내비게이션 서비스(T-map 등)에 악천후 노면상태 정보가 추가되어 도로교통 안전 관련 신시장 및 양질의 일자리 창출 가능
- 민간 플랫폼을 통해 수집되는 노면 상태 정보를 도로제설시스템 등 공공에서 활용할 경우 저비용으로 사업 추진 가능(※ 노면이 미끄러운 도로에 우선적으로 제설차량 투입)
- 본 기술 개발로 인해 특히 겨울철 야간 운전시 운전자가 가장 두려워하는 블랙아이스 등 노면 미끄럼 정보 제공이 가능하고, 제공된 정보로 인해 안전운전이 가능함에 따라 악천후 교통사고 감소 가능



(2) 기술적 성과

- 기존의 지점 검지방식의 AWS 및 RWIS 수집방식에서 벗어나 공간적으로 단절 없는 노면 상태(결빙, 적설) 정보수집이 가능한 주행차량 기반의 도로기상정보 수집기술 개발
- 차량에 장착된 센서(휠 펄스, 영상, GPS 등)를 이용하여 노면의 미끄럼, 도로 안개를 수집하는 기술로써, 현재 전 세계적으로 기술개발이 가장 활발하고 향후 100년간의 국부를 창출할 가능성이 있는 기술인 악천후 자율주행 기술수요에 부응하는 기술개발이기 때문에 전·후방 연관 산업 파급효과가 막대함
- 본 기술개발 결과의 사업화로 인해 수집되는 민간업체의 노면상태 정보와 기상청, 도로 관리청 등에서 고정식 기상장비를 통해 수집되는 도로기상정보와 융합할 경우, 시공간적으로 단절 없는 도로기상(노면상태) 정보 수집이 가능함

(3) 학술적 성과

- 직접촬영 및 오픈데이터 다운로드를 활용한 데이터셋 수집, 보수가 필요하다고 판단되는 프레임에 대하여 이미지 생성 및 파손된 도로노면 부위 Cropping 및 Segmentation으로 데이터 가공, 도로노면파손 부위 이미지 및 파손정보를 활용한 구조설계 가능
- 위와같은 방법을 통해 얻어진 데이터 셋으로 1차 성능검증을 위한 실험환경 설계완료

○ 성능검증용 시험대상 선정완료

- 시험대상 소프트웨어

구분	내용
개요	도로노면의 파손 부위(포트홀)를 탐지
개발 언어	Python 3.6.8
프레임워크	- Tensorflow 1.15.0 - CUDA 10.1
용량	12KB
실행 파일명	eval.py

- 인공지능 알고리즘

구분	내용
개요	촬영된 도로주행 영상으로부터 도로노면의 파손 부위(포트홀)를 학습
학습 네트워크	CNN 기반의 지도학습 수행
입력 데이터	- png: 도로노면 파손 부위 이미지 - npz: 도로노면 파손 정보
데이터 전처리	이미지 리사이징(960×128)
학습/검증 비율	5-fold cross validation 수행
학습 횟수	각 fold 당 100 epoch 반복
학습모델	각 fold 당 생성되는 모델을 앙상블하여 최종 결과값 도출 - pothole-segnet-v12-0-0038-0.7539.hdf5, 4.3MB - pothole-segnet-v12-1-0025-0.7519.hdf5, 4.3MB - pothole-segnet-v12-2-0031-0.7547.hdf5, 4.3MB - pothole-segnet-v12-3-0040-0.7556.hdf5, 4.3MB - pothole-segnet-v12-4-0043-0.7495.hdf5, 4.3MB
학습 성능지표	IOU (Intersection over union)

○ 성능항목구성

ID	시험항목	시험내용	목표치
TC1	도로노면 파손의 분할	도로주행 이미지에서 도로의 파손, 비파손 영역을 분할	mIOU 0.7 이상
TC2	도로노면 파손 탐지	도로주행 이미지에서 도로의 파손, 비파손 영역의 탐지 여부를 측정	탐지율 90% 이상

○ 성능 시험결과 : 만족

ID	구분	내용
TC1	항목명	도로노면 파손의 분할
	시험내용	도로주행 이미지에서 도로의 파손, 비파손 영역을 분할
	지표기준	도로의 파손, 비파손 영역의 mIOU(평균 IOU)를 산출
	평가목표	0.7 이상
	시험방법	1. 도로노면의 파손, 비파손 이미지를 학습한 프로그램을 실행하여, 평가용 데이터셋에 대한 파손, 비파손 영역에 대한 분할을 수행한다. 2. 평가용 데이터셋에 대한 영역 분할 후 각 영역에 대한 IOU를 측정한다. 3. 각 이미지의 파손 영역, 비파손 영역에 대한 mIOU(평균 IOU)를 구하고, 전체 이미지에 대한 IOU 평균이 0.7 이상인지 확인한다. ※ 결과값 확인: 0.7447158547638241
	시험결과	mIOU 0.74
	비고	※ IOU(Intersection over Union)는 이미지가 포함하고 있는 물체를 탐지하는 정도를 측정하기 위한 지표로 사용 - $IOU = (\text{Area of Intersection}) / (\text{Area of Union})$

ID	구분	내용
TC2	항목명	도로노면 파손 탐지
	시험내용	도로주행 이미지에서 도로의 파손, 비파손 영역의 탐지 여부를 측정
	지표기준	각 도로노면 이미지의 탐지율 측정 (기준: mIOU@0.5 이상)
	평가목표	90% 이상
	시험방법	1. 도로노면의 파손, 비파손 이미지를 학습한 프로그램을 실행하여, 평가용 데이터셋에 대한 파손, 비파손 영역에 대한 분할을 수행한다. 2. 평가용 데이터셋에 대한 영역 분할 후 각 영역에 대한 IOU를 측정한다. 3. 각 이미지의 파손 영역, 비파손 영역에 대한 mIOU(평균 IOU)를 구하고, 전체 이미지에 대한 평균 탐지율(기준: mIOU@0.5 이상)이 90% 이상인지 확인한다. 후 결과값 확인: 0.9090909090909091
	시험결과	탐지율 91%
	비고	-

(4) 경제적 성과

○ 경제성 분석 개요

- 본 제안서에서 경제성 분석은 기술개발시 즉시 적용이 가능한 주관기업 운영 스마트관제 서비스 이용차량을 대상으로 함
- 본 경제성 분석결과는 2017년 1월 기준 스마트관제서비스 이용차량만을 대상으로 하기 때문에 향후 서비스 이용차량이 증가하거나 생성정보를 제3자(카내비게이션, 교통정보 사업자 등) 등에게 연계(판매)할 경우 편익은 크게 증가되므로, 본 경제성 분석 결과는 기술개발에 따른 최소편익을 이용하여 산정한 결과임

○ 분석 방법론

- 경제성 분석은 편익-비용(B/C) 분석 방법론을 적용하여, 편익은 본 기술개발에 따른 교통사고 비용 감소분을 적용하였고, 비용은 연구개발비용과 서비스 이용요금으로 산정함
- 분석기간은 DTG 및 블랙박스 내용연수인 10년으로 적용하였고, 할인율 6.5%(KDI), 경제성장률 3%(기획재정부)를 적용하여 분석함

○ 도로 파손 정보 서비스의 경제적 효과

- 포트홀 발생 위치의 데이터가 축적되지 않아 포트홀 유지보수에 있어 많은 민원과 사고가 일어나고 있으며, 포트홀로 발생하는 차량 파손 및 민원 등이 증가하고 있어 사회적 비용이 증대하고 있는 실정임
- 본 기술의 사업화를 통해 도로 파손 정보의 체계화, 적기의 도로노면 보수시기 결정, 과학적 상세진단 등의 기술적 지원으로 차량파손 예방, 보험민원 감소, 교통사고 감소 등의 사회적 이익 증대를 기대

[공공사업 적용에 따른 예상 편익]

연간 도로교통사고 감소 편익 = 24.044조원1) × 0.4%2) × 20%3) ≒ 192.3 억원

1) 2013년 기준, 도로교통사고비용 = 24조 444억 원

출처 : 도로교통공단(2014), 2013년 지역별 도로교통사고비용의 추계

2) 전체 교통사고 중 포트홀 관련사고 비율 = 0.4%

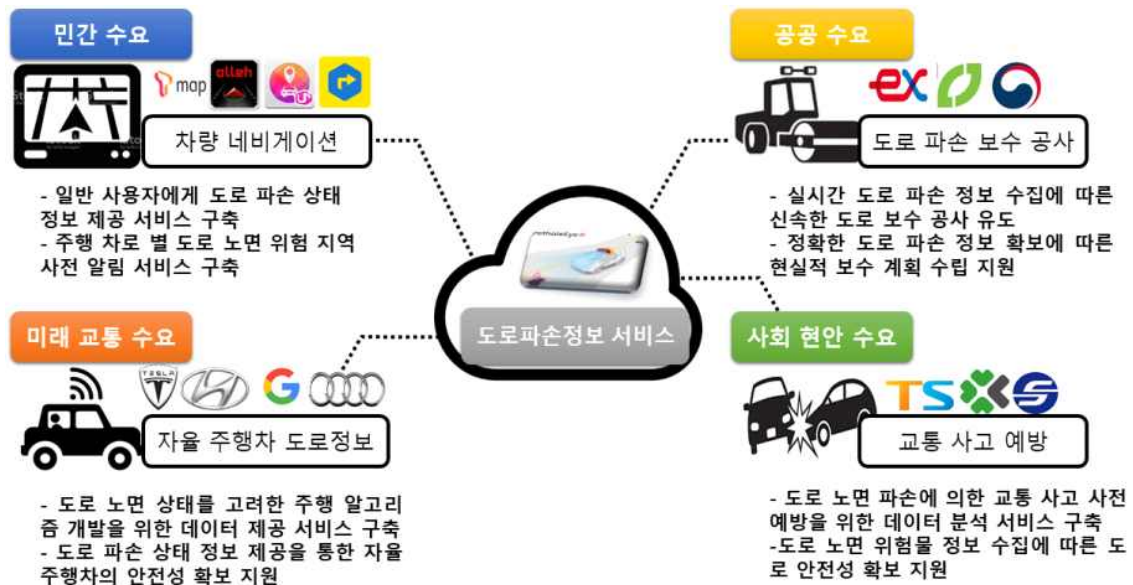
(최근 3년의 연간 포트홀 사고 861건, 교통사고 224,082건)

3) 포트홀 관련사고 감소 비율(추정치) = 20%

- 이 외에도 센터 플랫폼을 활용한 도로관리 시장, 자율주행자동차의 불량노면 알람 단말장치, 노면상태정보 콘텐츠 제공 서비스 등 다양한 시장으로의 비즈니스 모델을 발굴하면 상당한 시장규모로 성장할 수 있는 분야임

3. 목표 달성도 및 관련 분야 기여도

3-1. 목표



<사업화 서비스 모델>

구분	내 용
최종 목표	교통사고 예방을 위한 도로 파손 정보 시스템 구축 및 사업화
세부 목표	- 민간 및 공공 기관을 대상으로 한 도로 노면 파손 제공 기술 사업화 * 실시간 도로 파손 정보 수집에 따른 신속한 도로 보수 공사 유도 * 정확한 도로 파손 정보 확보에 따른 현실적 보수 계획 수립 지원 - 차량 네비게이션을 활용한 도로 노면 파손 정보 제공 기술 사업화 * 일반 사용자에게 도로 파손 상태 정보 제공 기술 구축 * 주행 차로별 도로 노면 위험 지역 사전 알림 시스템 구축 - 초저지연 기술기반의 도로 노면 상태 정보 제공 서비스 구현 * 통신망과 연계한 인공지능 기술 기반의 도로 노면 상태 정보 제공 * 인공지능 기술을 활용하여 도로 노면 정보에 대한 데이터 확보 * 향후 지역 내 운전자에게 공유할 수 있는 교통 정보 플랫폼 제공 - 자율 주행차의 안전성 향상을 위한 도로 파손 정보 제공 기술 사업화 * 도로 노면 상태를 고려한 주행 알고리즘 개발을 위한 데이터 제공 기술 개발 * 도로 파손 상태 정보 제공을 통한 자율 주행차의 안전성 확보 지원
활용계획	- 다수의 수집단말기를 활용한 시공간적으로 단절 없는 도로 노면상태 정보 수집 - 지자체/도로관리공사에 정보 제공을 통한 신속 보수체계 구축 - 탐지된 노면 정보를 실시간으로 피드백하여 안전운전을 위한 보조서비스로 활용 - 도로 노면 파손 정보 제공을 통한 안전운전 유도로 교통사고 감소 가능

3-2. 목표 달성여부

구분(연도)	사업화 목표	사업화 전략 및 방법	성과물
1차 연도 (2018)	▪비즈니스 모델 개발	▪웰아이피 컨설팅 회사에 의뢰하여 비즈니스 모델 개발	▪BM 보고서 1부
	▪특허 출원	▪시스템 및 알고리즘 특허	▪특허 4건
	▪기존 기술개선 및 검증	▪2차년도 테스트베드 운영용 단말기의 데모용 시제품 개발	▪시제품 3대 (저체 테스트용)
2차 연도 (2019)	▪마케팅 전략 수립	▪웰아이피 컨설팅 회사에 의뢰하여 마케팅 전략 수립 ▪양산, 판로개척, IR자료 작성	▪마케팅전략 보고서 1부
	▪SW 인증	▪TTA에 의뢰하여 GS 인증	▪GS 인증 1건
	▪소프트웨어 품질 검증 (시스템 SW)	▪개발 소프트웨어의 상존하는 장애문제 및 기관운영에 필요한 프로그램 안전성 확보 ▪전문업체 소프트플로우에 의뢰하여 소프트웨어 품질검증	▪결과보고서 1부
	▪시스템 구축	▪테스트베드 운영시스템 구축	▪시스템 매뉴얼 1부
3차 연도 (2020)	▪시장진입 및 양산, 판로개척	▪양산 체계 구축 ▪양산, 판로개척, IR자료 작성 ▪제품 인증	▪KC 인증 2건 ▪프로그램 등록 3건
	▪소프트웨어 품질 검증 (단말기 SW)	▪전용 단말기 내소프트웨어의 상존하는 장애문제 및 기관운영에 필요한 프로그램 안전성 확보 ▪전문업체 소프트플로우에 의뢰하여 소프트웨어 품질검증	▪결과보고서 1부

3-3. 목표 미달성 시 원인(사유) 및 차후대책(후속연구의 필요성 등)

○ 3차년도 연구성과물 중 신기술인증 / 녹색기술인증 제한

- 신기술/녹색기술인증 관련사항은 당사가 제시한 성과목표는 아님.(세부사항 별도)
- 신(5.21)/녹색기술(7.29) 신청했으나 실패(인증기관 : koita)
 - 해당 기술이 정식 사업화가 실시되지 않았으며 기술의 특성상 본 기술은 해당 기술인증 항목에 적합하지 않음
 - 기술인증 시점은 사업화 시작후 2-3년 후 실적을 기준으로 검토 필요
- 기술인증 관련 KAIA 해당부서 미팅
 - 사업화에 대한 결과를 통해 적정시간이 지난 후 그 결과를 갖고 기술신청 요청
 - 과제사업부서와 인증부서간의 기술인증 시점에 대한 이견 있었음.
 - 현재 R&D 중간에 기술인증 받은 사례 없다고 함.
- 내부 조치사항으로 기술인증 컨설팅 추진 미팅을 통해 자문 중임
- 본 과제의 전문기관 부서 협의 후 “ICT 융합품질인증 “ 관련 준비 중

○ 달성된 목표의 질적 수준 검증

- 질적 수준 관련 3차년도 실적 보고 시 외부기관 인증 필요하다고 하였음.
- 주관기관의 시스템 관련하여 GS 인증 1등급 획득한 상황임
(인증 19-0499 / 시험 TGS-A-19-289)
- 정탐을 등에 대한 부분은 공동연구기관인 KICT에서 TTA로 외부시험 평가완료
(TTA_BT-A-20-0124)



○ 기술검증(외부전문기관)

- 정탐을 등에 대한 부분은 공동연구기관인 KICT에서 외부기관인 TTA로 검증완료
(문서 : TTA_BT-A-20-0124)

시험성적서		
한국정보통신기술협회 소프트웨어시험인증연구소 주소: 경기도 성남시 분당구 분당로 47 전화: 031-724-0218, Fax: 031-724-0189	결과서번호: BT-A-20-0124	
1. 의뢰자 · 회사(기관)명 : 한국건설기술연구원 · 주 소 : 경기도 고양시 일산서구 고양대로 283 · 계약일자 : 2020. 03. 27 2. 시험성적서의 용도 : 성능 검증 3. 제 품 명 : 포트를 탐지 시스템 4. 버 전 : v1.0 5. 시험기간 : 2020. 03. 16. 6. 시험방법 : 시험결과서 참조 7. 시험환경 : 시험결과서 참조 8. 시험결과 : 시험결과서(BT-A-20-0124) 불임		
확 인	작성 자 성 명 : 배 동 석 (서명)	승 인 자 직 위 : 관제크리팅장 성 명 : 신 준 호 (서명)
2020년 3월 30일 한국정보통신기술협회 회장 		

TPS-0071-1(00)

페이지 : (1) / (총 12)

○ 사업화 매출실적

구분	세부내용	수입		
		공급가	부가세	계
자치단체	자치단체 SI 업체	172,060,000	17,206,000	189,266,000
연구용역	5G MEC 시스템 구축	250,000,000	25,000,000	275,000,000
조달(납품)	수원국토 단말기 및 시스템연동	390,909,091	39,090,090	429,999,181
합 계		812,969,091	81,296,090	894,265,181

매출 전자세금계산서 명세

매출 전사세금계산서 명세						(단위:천원)	
총건수	3	총합계금액	264,000,000	총 공급가액	240,000,000	총 세액	24,000,000
일련번호	작성일자	승인번호	공급받는자 등록번호	상호	대표지명	전자세금계산서종류	
	발급일자	전송일자	합계금액	공급가액	세액	발급유형	
1	2020-12-23	20201223-41000008-000czra	409-86-15475	(주)그린이엔에스	이숙희	일반	
	2020-12-23	2020-12-24	44,000,000	40,000,000	4,000,000	ASP발급	
2	2020-11-20	20201120-41000008-000ce96k	409-86-15475	(주)그린이엔에스	이숙희	일반	
	2020-11-23	2020-11-24	88,000,000	80,000,000	8,000,000	ASP발급	
3	2020-10-19	20201019-41000008-000bt7au	409-86-15475	(주)그린이엔에스	이숙희	일반	
	2020-10-20	2020-10-21	132,000,000	120,000,000	12,000,000	ASP발급	

전자세금계산서						승인번호		20201228-42000105-g3799297			
공 급 자	등록 번호	305-81-36265		종사업장 번호		공 급 받 는 자	등록 번호	135-83-00236		종사업장 번호	
	상호 (법인명)	주식회사 케이웍스		성명	최종선		상호 (법인명)	국토교통부 서울지방국토관리청 수원국토관리사무소		성명	수원국토관리사무소장
	사업장 주소	대전광역시 유성구 도안북로 54-53 (용계동)					사업장 주소	경기도 용인시 기흥구 용구대로2291번길 12-0 (신갈동)			
	업태	서비스	종목	SW 개발 및 공급			업태		종목		
	이메일						이메일	jin1041@korea.kr			
작성일자		공급가액		세액		수정사유		비고			
2020-12-28		388,954,546		38,895,454		해당없음		공급받은 자의 세금계산서 번호: 20201228-42000105-g3799297 공급받은 자의 사업자번호: 135-83-00236 공급받은 자의 업태: 서비스 공급받은 자의 종목: SW 개발 및 공급			
월	일	품목		규격	수량	단가	공급가액		세액	비고	
12	28	통신서버소프트웨어		WebScan 1.0. ~ 100 User licenses 제어 플랫폼	36	11,884,722	388,954,546		38,895,454	10065025	
합계금액		현금		수표		여음		외상미수금		이 금액을 (청구) 함	
427,850,000		427,850,000									

본 인쇄물은 국세청 홈택스(www.hometax.go.kr)에서 발급 또는 전송 입력된 전자(세금)계산서입니다.

발급사실 확인은 상기 홈페이지의 "조회/발급>전자서금제산서> 제3자 발급사실 조회"를 이용하시기 바랍니다.

전자세금계산서					승인번호		20201014-41000008-000bqohi					
공 급 자	등록 번호	305-81-36265		종사업장 번호	공 급 받 는 자	등록 번호	314-86-21228		종사업장 번호			
	상호 (법인명)	(주)케이웍스		성명		최종선		상호 (법인명)	주식회사 와이티		성명	김용태,홍운혁
	사업장 주소	대전광역시 유성구 도안북로 54-53(용계동)				사업장 주소	대전광역시 유성구 가정북로 96 대전경제통상진흥원 703호					
	업태	서비스,제조 외	종목	S/W개발및공급외		업태	제조서비스	종목	소프트웨어자문,개발및공급			
	이메일	moog@kworks.co.kr				이메일	stellar8808@hanmail.net					
작성일자		공급가액		세액	수정사유		비고					
2020-10-14		14,400,000		1,440,000	해당없음							
월	일	품목		규격	수량	단가	공급가액	세액	비고			
10	14	시스템개발 용역					14,400,000	1,440,000				
합계금액		현금		수표	어음	외상미수금		이 금액을 (청구) 함				
15,840,000												

본 인쇄물은 국세청 홈택스(www.hometax.go.kr)에서 발급 또는 전송 입력된 전자(세금)계산서입니다.
 발급사실 확인은 상기 홈페이지의 "조회/발급"전자세금계산서> 제3자 발급사실 조회 "를 이용하시기 바랍니다.

전자세금계산서					승인번호		20201102-41000008-000c3p68				
공 급 자	등록 번호	305-81-36265		종사업장 번호	공 급 받 는 자	등록 번호	301-86-16490		종사업장 번호		
	상호 (법인명)	(주)케이웍스		성명		최종선	상호 (법인명)	(주)코아아이티		성명	신동용
	사업장 주소	대전광역시 유성구 도안북로 54-53(용계동)				사업장 주소	충청북도 청주시 청원구 오창읍 각리1길 97 (충청북도유 출렛폼운영센터 3층)				
	업태	서비스,제조 외	종목	S/W개발및공급외		업태	서비스업외	종목	소프트웨어개발및자문등		
	이메일	moog@kworks.co.kr				이메일	7777@coreit.co.kr				
작성일자		공급가액		세액	수정사유		비고				
2020-11-02		50,000,000		5,000,000	해당없음						
월	일	품목		규격	수량	단가	공급가액	세액	비고		
11	02	인공지능 솔루션 개발					50,000,000	5,000,000			
합계금액		현금		수표		어음		외상미수금		이 금액을 (청구) 함	
55,000,000											

본 인쇄물은 국세청 홈택스(www.hometax.go.kr)에서 발급 또는 전송 입력된 전자(세금)계산서입니다.
 발급사실 확인은 상기 홈페이지의 "조회/발급"전자세금계산서> 제3자 발급사실 조회 "를 이용하시기 바랍니다.

○ 홍보활동

◦ 전용 홍보페이지 제작 완료 <https://potholei.com/>

◦ 리플렛 제작 완료(국문 2건 / 영문1건)

◦ 전시회 취소로 인한 방송 송출 추진

YTN 극찬기업 1회(9. 7) YTN 황금나침반(11.14), 신영일의 비즈정보 플러스(11.20)
조영구



[조영구의 트렌드 핫 이슈] -케이웍스

조영구의 트렌드 핫 이슈 네이버TV

▶ 12 2020.12.17.



[신영일의 비즈정보 플러스] 케이웍스

신영일의 비즈정보 플러스 네이버TV

▶ 27 2020.11.24.

[조영구의 트렌드 핫 이슈] -케이웍스 (naver.com)

[신영일의 비즈정보 플러스] 케이웍스 (naver.com)



도로 위 지뢰를 찾아라 [극찬기업] 5-3 / YTN 사이언스

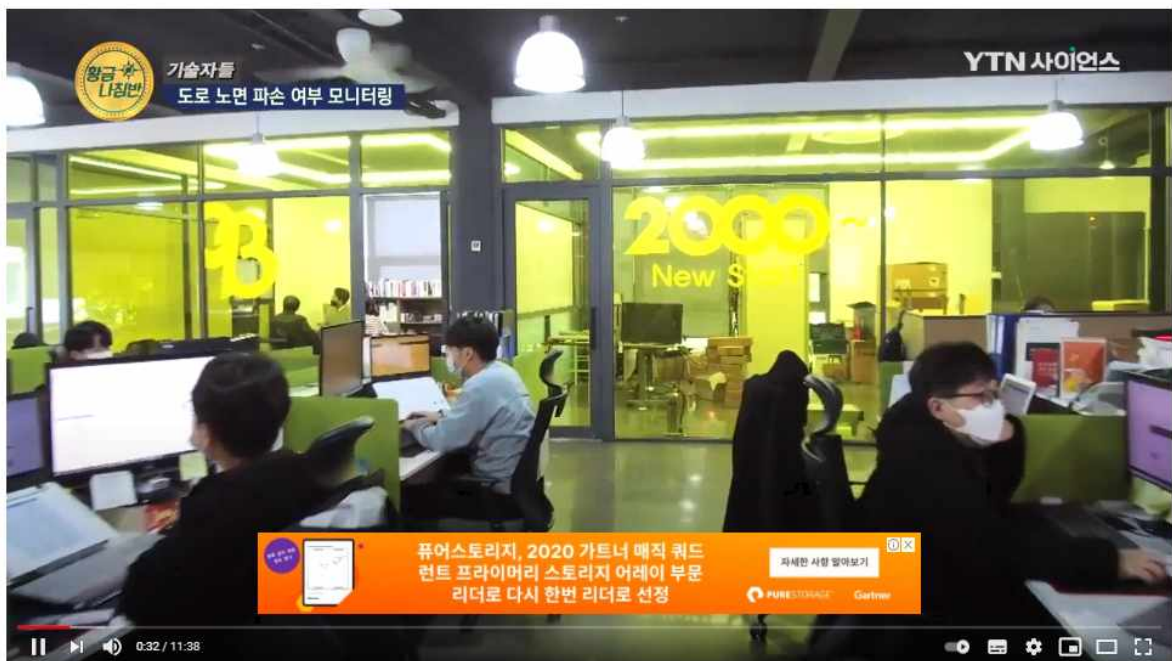
조회수 1,052회 · 2020. 9. 7.

19 1 공유 저장 ...



*차를 타면 대한민국 어디든 못 갈 곳이 없다
하지만 '유전자'들이 쉽게 예상하지 못한면 사고 요소가 인류에

도로 위 지뢰를 찾아라! [극찬기업] 5-3 / YTN 사이언스 - YouTube



[기술자들] 도로 노면 파손 여부 모니터링 / YTN 사이언스

조회수 1,501회 · 2020. 11. 15.

16 3 공유 저장 ...



도로 노면 파손 여부를 확인할 수 있는 첨단 기술부터
어린이 안전을 위해 공중부양하는 횡단보도까지

[기술자들] 도로 노면 파손 여부 모니터링 / YTN 사이언스 - YouTube

4. 연구개발성과의 활용 계획 등

○ 서비스 제공자 및 사용자 제공·활용과 관련된 프로세스

- 국내 최초 개발한 도로노면에 대한 교통정보서비스 플랫폼 구축완료
- 경기도(고양), 대구광역시(스마트시티센터), 충청북도(청주시) 등 관리적 운영에서 대국민서비스로의 전향을 위해 광주광역시에서의 서비스 실증을 위한 테스트베드 확대
- 국가 교통체계를 담당하고 있는 중앙부처 및 산하기관으로의 서비스 확대
- 센터 플랫폼을 활용한 도로 교통 시장, 자율주행 자동차 단말 장치, 도로 상태 정보 콘텐츠 제공 서비스 등 다양한 시장과의 융합서비스
- 인공지능 기반의 연동 인터페이스 및 서비스 시나리오 표준화
- 국내외 자율 주행 차량 시장에서의 도로 노면 교통 정보 분석 SW 컴팩트 버전 개발을 통해 해당장비의 칩셋 시장 진입 및 분석 솔루션 제공
- 공공기관 차량 및 대중교통 차량 탑재를 통한 데이터 수집 및 분석서비스 확대
- 본 과제를 통하여 개발한 수정 교통 수집 장치 단말기를 보다 많은 수량의 차량 적용을 위하여 다수의 차량 운행이 진행되고 있는 대중교통차량(버스 및 택시)에 적용하여 실시간 교통 위험정보 수집 적용 지역을 넓히고, 대량 데이터를 수집하여 보다 효율적인 서비스 제공을 추진한다. 광주광역시 뿐만 아니라 다수의 지자체 및 대중교통 운영 기관으로의 제안 및 시범서비스, 사용서비스를 추진
- 인공지능 인식성능 확대를 위한 인공지능 플랫폼 확대
- 현재 수행하고 있는 교통정보수집장치를 통한 교통정보 및 위험정보 인식 기능을 수집된 데이터 기반으로 클라우드 내에서 이중 수행하여 보다 높은 품질의 인식 성능 확보 및 포트홀 이외의 다양한 인공지능 서비스를 개발한다. 고성능, 고품질의 서비스를 통해 보다 많은 수요처를 확보하여 서비스를 확산
- 사후 계획으로 티맵, 네이버지도, 카카오네비 등 대국민서비스에 오픈소스 연동을 통한 포트홀 탐지 위험구간 알림 및 사용구간 확대

○ 사업화

- 총 4단계로 구성하여 사업화 실시
- 시범사업 기반의 1단계 사업화는 기존 시제 모델로 개발한 스마트폰으로 개발된 분석 알고리즘을 적용하여 대구광역시 - 대구시설관리공단, 충청북도 - 청주시청 등 운영
- 2단계 사업화는 개발된 전용 단말기를 기반으로 공공기관을 대상으로 사업화

구 분	자치단체(시군구)	국토관리청	한국도로공사	합계
단체수	226	18	310	554
보수차량현황(평균)	5	20	10	-
계	1,130	360	3,100	4,590

* 자치단체 도로보수 용역업체 차량 산정불가로 미계상

- 국내외 출품을 위한 3단계 사업화는 국내 1회(안전산업박람회), 국외 1회 (CES 또는 MWC) 출품
- 해외사업화 중심의 4단계의 경우 (주)케이웍스는 KOTRA와 협의하며 SK텔레콤 글로벌TF 협업을 통해 해외지사 기반의 공공영업 추진
- * (주)케이웍스는 SK텔레콤 주식회사 IoT 부문 5대 비즈니스 파트너사임

▷ 2021년도 사업화 예정 계획

단체명	단말수	비고
서울 은평구청	3	각 지자체별 의견 조율중
대구시청	30	
광주시청	64	
경기도 화성시청	1	
경상남도 김해시청	30	
계	128	

○ 양산사업화

- 국내 전용단말기
- 시장기준 양산 추정소요 : 48,000 여대

<공공기관>

구 분	자치단체(시군구)	국토관리청	한국도로공사	합계
단체수	226	18	310	554
보수보유현황(평균)	5	20	10	-
계	1,130	360	3,100	4,590

□ 포트홀 탐지 데이터 현황

탐지정보 - 사진 모아보기

총 : 20/300건

단말기선택

2021-02-02

2021-02-02

검색

발생

검수

보수완료

삭제

리스트 보기

시간순

전체

[시간 : 2021-02-02 15:11:09.480] [갯수 : 1] [상태 : 발생]

[시간 : 2021-02-02 15:10:12.955] [갯수 : 1] [상태 : 발생]

[시간 : 2021-02-02 15:09:05.097] [갯수 : 3] [상태 : 발생]

[시간 : 2021-02-02 15:08:31.398] [갯수 : 1] [상태 : 발생]

[시간 : 2021-02-02 15:07:30.877] [갯수 : 1] [상태 : 발생]

[시간 : 2021-02-02 15:02:48.012] [갯수 : 1] [상태 : 발생]

[시간 : 2021-02-02 14:59:51.222] [갯수 : 1] [상태 : 발생]

[시간 : 2021-02-02 14:55:50.666] [갯수 : 2] [상태 : 발생]

[시간 : 2021-02-02 14:55:06.414] [갯수 : 1] [상태 : 발생]

[시간 : 2021-02-02 14:52:05.894] [갯수 : 1] [상태 : 발생]

[시간 : 2021-02-02 14:51:52.064] [갯수 : 1] [상태 : 발생]

[시간 : 2021-02-02 14:45:32.383] [갯수 : 1] [상태 : 발생]

[시간 : 2021-02-02 14:43:37.033] [갯수 : 1] [상태 : 발생]

[시간 : 2021-02-02 14:39:34.276] [갯수 : 2] [상태 : 발생]

[시간 : 2021-02-02 14:21:19.733] [갯수 : 1] [상태 : 발생]

[시간 : 2021-02-02 14:14:03.795] [갯수 : 2] [상태 : 발생]

[시간 : 2021-02-02 14:13:48.027] [갯수 : 4] [상태 : 발생]

[시간 : 2021-02-02 14:11:27.143] [갯수 : 1] [상태 : 발생]

<민간기업>

구 분	시내버스	농어촌버스	시외버스	고속버스	합계
업체수	352	84	78	11	525
보수대수	33,689	1,826	7,676	2,086	45,277

* 국토교통부(2016년 기준) 대중교통현황조사 : e-나라지표

· 실제 양산 및 판매 소요 : 5개년 기준 납품 / 40,000대

구 분	2019	2020	2021	2022	2023	계
국내 납품 기준	1,000	3,000	6,000	10,000	20,000	40,000

- 해외 전용단말기 : 미계상

· 2020년 사업종료시 해외 전용단말기 개발 후 kotra를 통한 시장수요 확인 후 양산

· SK텔레콤(글로벌TF) 시장수요조사 후 사업화 협의(2020년)

○ 사업화 매출 기대효과

구 분		2020년	2021년	2022년	2023년
사업화 기술(제품, 공법)		단말기	단말기, 솔루션	서비스	서비스
투자계획(백만원)		490	1,397	2,646	3,920
판매 계획 (백만원)	내 수	1,420	4,980	10,680	19,240
	직접수출	-	-	-	-
	간접수출			1,068	1,924
	계	1,420	4,980	11,748	21,164
비용절감(백만원)		-	10,508	86,864	199,864
수입대체(백만원)		-	-	-	-

<판매계획 : 내수기준>

연도	항목	수량	단가	기간(횟수)	매출(원)		비고
					완전판매시	재고율(15%)	
2020	단말매출	1,000	700,000	1	700,000,000		
	연통신료	1,000	60,000	12	720,000,000		
	계	0			1,420,000,000	1,207,000,000	
2021	단말매출	3,000	700,000	1	2,100,000,000		
	연통신료	1,000	60,000	12	720,000,000		
		3,000	60,000	12	2,160,000,000		
	계	3,000			4,980,000,000	4,341,000,000	
2022	단말매출	6,000	700,000	1	4,200,000,000		
	연통신료		60,000	12	-		20년 가입단말교체
		3,000	60,000	12	2,160,000,000		
		6,000	60,000	12	4,320,000,000		
	계	9,000			10,680,000,000	9,402,000,000	
2023	단말매출	10,000	700,000	1	7,000,000,000		
	연통신료	1,000	60,000	12	720,000,000		
			60,000	12	-		21년 가입단말교체
		6,000	60,000	12	4,320,000,000		
		10,000	60,000	12	7,200,000,000		
	계	20,000			19,240,000,000	17,110,000,000	
2024	단말매출	20,000	700,000	1	14,000,000,000		
	연통신료		60,000	12	-		20년 가입단말교체
		3,000	60,000	12	2,160,000,000		
			60,000	12	-		22년 가입단말교체
		10,000	60,000	12	7,200,000,000		
		20,000	60,000	12	14,400,000,000		
	계	43,000			37,760,000,000	33,500,000,000	

○ 고용현황

구 분	2017년 (기술개발 전년)	2018년 (개발종료 해당년)	2019년 (개발종료 후 1년)	2020년 (개발종료 후 2년)
신규고용(명)	-	3 (개발1, 사업화2)	2 (개발1, 사업화1)	1 (사업화1)
상시고용(명)	25	28	30	31

주 의

1. 이 보고서는 국토교통부에서 시행한 국토교통기술사업화지원사업의 연구보고서입니다.
2. 이 보고서 내용을 발표하는 때에는 반드시 국토교통부에서 시행한 국토교통기술사업화지원사업의 연구결과임을 밝혀야 합니다.
3. 국가과학기술 기밀 유지에 필요한 내용은 대외적으로 발표 또는 공개하여서는 안 됩니다.