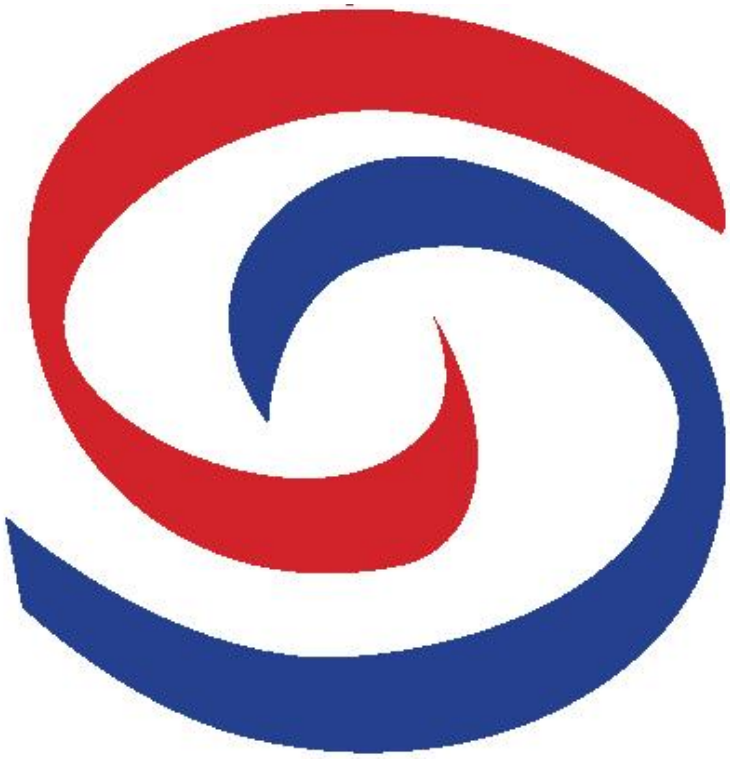


aParkigs-AI System

SenTerm 과 Jetson Nano 를 이용한 주차 공간 및
QR 코드를 통한 주차 위치 확인 시스템

김수빈, 이재혁

목차

1. 프로젝트 목표

aParkings-AI System은 SenTerm이 촬영한 영상을 Jetson Nano로 전달하여 객체 탐지 알고리즘(Yolo v3 알고리즘)을 통해 주차구역내 자동차의 유무를 파악 후 빈 주차 공간을 알려주는 시스템이다. 이 시스템의 부가적인 서비스로는 QR코드를 통해 주차한 위치를 대략적으로 알 수 있는 애플리케이션도 포함되어 있다.

이 프로젝트는 사용자의 편리한 주차 및 주차한 위치를 빠르게 찾을 수 있는 서비스를 제공하는 것이 목표이다.

2. 프로젝트 개발 환경

개발환경	구분	상세 내용
S/W 개발환경	OS	Windows 7, Ubuntu 18.04 LTS
	개발환경(IDE)	pyCharm, gedit,
	개발도구	OpenCV 4.0.1
	개발언어	C, Python3
	기타사항	Yolo v3
H/W 개발환경	디바이스	Jetson Nano, SenTerm
	통신	UDP Socket
	언어	C, Python3

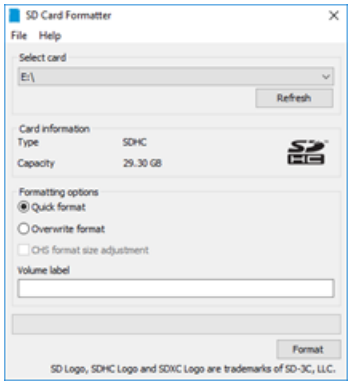
2.1. Jetson Nano 환경 설정

2.1.1. Insert Micro SD Card

1. Micro SD Card Format

SD Card Formatter를 다운로드하고 설치한다.

: https://www.sdcard.org/downloads/formatter_4/eula_windows/



SD 카드 드라이브를 선택한 후 Quick format을 선택한다. Volume label은 비워 두고 Format을 클릭한 후 Yes를 눌러 포맷을 시작한다.

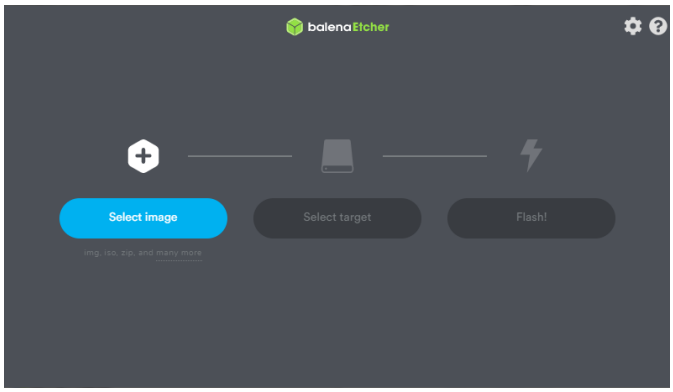
2. Write image file to Micro SD Card

이미지(JetPack)를 다운로드 한다.

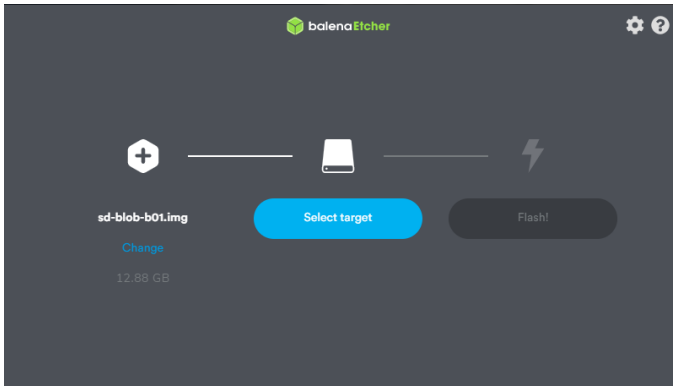
: https://developer.nvidia.com/jetson-nano-sd-card-image-r3231

Etcher를 다운로드하여 설치한다

: https://www.balena.io/etcher

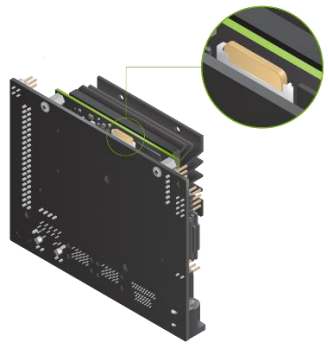


Select Image를 클릭해 다운로드한 이미지 압축파일을 선택한 후, PC에 Micro SD Card를 삽입한다.



Select Target을 클릭한 후 Micro SD Card에 해당하는 드라이브를 선택하고 Flash를 클릭한다. (약 10~30분 소요)

3. Insert Micro SD Card into Jetson Nano



2.1.2 Jetson Nano Start

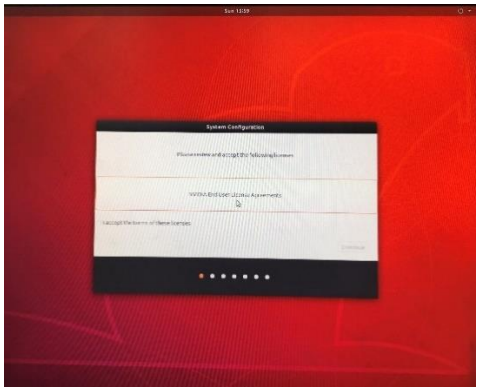
1. Jetson Nano Power Connection

i. Micro USB 단자를 통한 전원 (5V 2A, 저전력 모드)
ii. Barrel Jack 단자를 통한 전원 (5V 4A, j48핀에 점퍼핀 연결, 일반 모드)
iii. GPIO Header를 통한 전원 (2개의 5V 3A핀 존재, 총 6A)

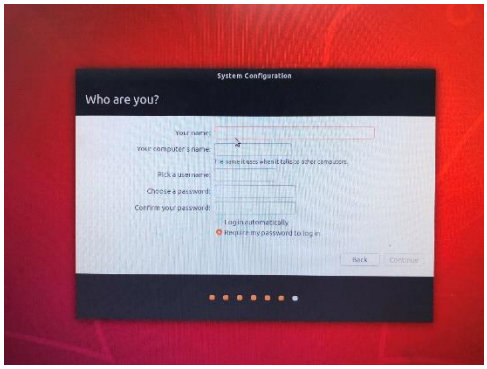
2. Connection Port

마우스, 키보드, 모니터, 전원선, LAN 선, J48핀 연결

3. Start

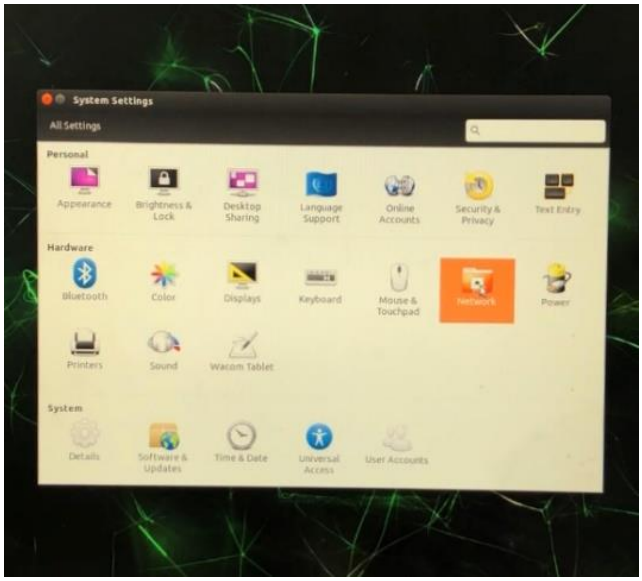


[Continue]를 누른다.

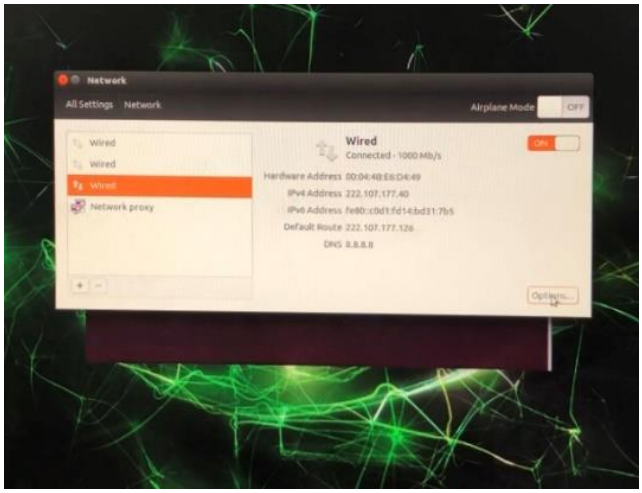


정보 입력 창이 뜨면 정보를 입력한다.

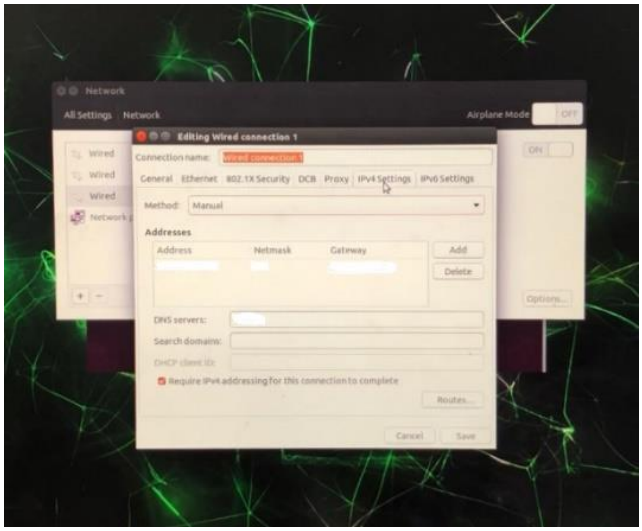
4. Assign IP



[Network] 폴더를 누른다.



인터넷이 연결되어 있는 이더넷(wired)을 선택한다.



[IPv4 Settings]를 누른 후 Method를 Manual로 설정하고 Address 정보를 기입한다.

2.1.3. Install OpenCV 4.0.1

1. Delete the previously saved OpenCV

```
$ pkg-config --modversion opencv
$ sudo apt-get remove libopencv*
$ sudo apt-get autoremove
```

2. Update & Upgrade

```
$ sudo apt-get update
$ sudo apt-get upgrade
```

3. Install package

```
$ sudo apt-get install build-essential cmake
$ sudo apt-get install pkg-config
$ sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev
$ sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libxvidcore-dev libx264-dev libxine2-dev
$ sudo apt-get install libv4l-dev v4l-utils
$ sudo apt-get install libgstreamer1.0-dev libgstreamer-plugins-base1.0-dev
$ sudo apt-get install libqt4-dev
$ sudo apt-get install mesa-utils libgl1-mesa-dri libqt4-opengl-dev
$ sudo apt-get install libatlas-base-dev gfortran libeigen3-dev
$ sudo apt-get install python2.7-dev python3-dev python-numpy python3-numpy
```

4. Create opencv directory

```
$ mkdir opencv
$ cd opencv
```

5. Download OpenCV 4.0.1

```
$ wget -O opencv.zip https://github.com/opencv/opencv/archive/4.0.1.zip
$ unzip opencv.zip
$ wget -O opencv_contrib.zip https://github.com/opencv/opencv_contrib/archive/4.0.1.zip
$ unzip opencv_contrib.zip
```

6. Create build directory

```
$ cd opencv-4.0.1/
$ mkdir build
$ cd build
```

7. Build configuration

```
$ cmake -D CMAKE_BUILD_TYPE=RELEASE ₩
-D CMAKE_INSTALL_PREFIX=/usr/local ₩
-D WITH_TBB=OFF ₩
-D WITH_IPP=OFF ₩
-D WITH_1394=OFF ₩
-D BUILD_WITH_DEBUG_INFO=OFF ₩
-D BUILD_DOCS=OFF ₩
-D INSTALL_C_EXAMPLES=ON ₩
-D INSTALL_PYTHON_EXAMPLES=ON ₩
-D BUILD_EXAMPLES=OFF ₩
```

```
-D BUILD_TESTS=OFF ₩
-D BUILD_PERF_TESTS=OFF ₩
-D WITH_QT=ON ₩
-D WITH_GTK=OFF ₩
-D WITH_OPENGL=ON ₩
-D OPENCV_EXTRA_MODULES_PATH=../../opencv_contrib-4.0.1/modules ₩
-D WITH_V4L=ON  ₩
-D WITH_FFMPEG=ON ₩
-D WITH_XINE=ON ₩
-D BUILD_NEW_PYTHON_SUPPORT=ON ₩
-D PYTHON2_INCLUDE_DIR=/usr/include/python2.7 ₩
-D PYTHON2_NUMPY_INCLUDE_DIRS=/usr/lib/python2.7/dist-packages/numpy/core/include/ ₩
-D PYTHON2_PACKAGES_PATH=/usr/lib/python2.7/dist-packages ₩
-D PYTHON2_LIBRARY=/usr/lib/x86_64-linux-gnu/libpython2.7.so ₩
-D PYTHON3_INCLUDE_DIR=/usr/include/python3.5m ₩
-D PYTHON3_NUMPY_INCLUDE_DIRS=/usr/lib/python3/dist-packages/numpy/core/include/  ₩
-D PYTHON3_PACKAGES_PATH=/usr/lib/python3/dist-packages ₩
-D PYTHON3_LIBRARY=/usr/lib/x86_64-linux-gnu/libpython3.5m.so ₩
../
```

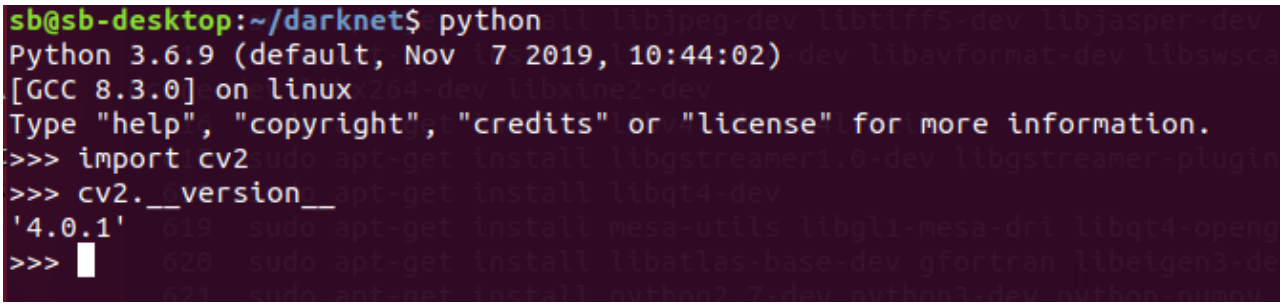
8. Compile

```
$ make -j4
```

9. Install OpenCV

```
$ sudo make install
$ sudo sh -c echo '/usr/local/lib/' > sudo /etc/ld.so.conf.d/opencv.conf
$ sudo ldconfig
```

10. Check OpenCV



2.1.4 Python Path Change (Ubuntu)

1. Check the python version

```
sb@sb-desktop:~$ python -V
Python 2.7.17
sb@sb-desktop:~$ python3 -V
Python 3.6.9
```

2. Check the location of python

```
sb@sb-desktop:~$ which python
/usr/bin/python
```

3. Check file pointing to `ls -al /usr/bin/python`

```
sb@sb-desktop:~$ ls -al /usr/bin/python
lrwxrwxrwx 1 root root 9  4월 16  2018 /usr/bin/python -> python2.7
```

4. Check for different versions of Python executable via `ls /usr/bin/ | grep python`

```
sb@sb-desktop:~$ ls /usr/bin/ | grep python
aarch64-linux-gnu-python2.7-config
aarch64-linux-gnu-python3.6-config
aarch64-linux-gnu-python3.6m-config
aarch64-linux-gnu-python3-config
aarch64-linux-gnu-python3m-config
aarch64-linux-gnu-python-config
dh_python2
dh_python3
python
python2
python2.7
python2.7-config
python2-config
python3
python3.6
python3.6-config
python3.6m
python3.6m-config
python3-config
python3m
python3m-config
python-config
```

5. Python version change

```
sb@sb-desktop:~$ sudo update-alternatives --config python
update-alternatives: error: no alternatives for python
```

If nothing is set up like an error log, it means nothing is registered.

```
sb@sb-desktop:~$ sudo update-alternatives --install /usr/bin/python python /usr/bin/python2.7 1
update-alternatives: using /usr/bin/python2.7 to provide /usr/bin/python (python) in auto mode
sb@sb-desktop:~$ sudo update-alternatives --install /usr/bin/python python /usr/bin/python3.6 2
update-alternatives: using /usr/bin/python3.6 to provide /usr/bin/python (python) in auto mode
```

Register executable.

```
sb@sb-desktop:~$ sudo update-alternatives --config python
There are 2 choices for the alternative python (providing /usr/bin/python).

  Selection    Path                Priority   Status
------------------------------------------------------------
* 0            /usr/bin/python3.6   2         auto mode
  1            /usr/bin/python2.7   1         manual mode
  2            /usr/bin/python3.6   2         manual mode

Press <enter> to keep the current choice[*], or type selection number: 2
```

Enter `update-alternatives --config python` again to get a menu to select the registered Python version.

```
sb@sb-desktop:~$ python -V
Python 3.6.9
```

Enter the desired menu number and check the Python version.

```
sb@sb-desktop:~$ ls -al /usr/bin/python
lrwxrwxrwx 1 root root 24  1월 21 17:12 /usr/bin/python -> /etc/alternatives/python
sb@sb-desktop:~$ ls -al /etc/alternatives/python
lrwxrwxrwx 1 root root 18  1월 21 17:12 /etc/alternatives/python -> /usr/bin/python3.6
```

2.1.5. Install Yolo

1. Darknet git clone

```
$ git clone https://github.com/pjreddie/darknet
```

2. Change Makefile

```
$ cd darknet
$ sudo vi Makefile
```

OPENCV=1로 변경

```
1 GPU=0
2 CUDNN=0
3 OPENCV=1
4 OPENMP=0
5 DEBUG=0
6
7 ARCH= -gencode arch=compute_30,code=sm_30 \
8       -gencode arch=compute_35,code=sm_35 \
9       -gencode arch=compute_50,code=[sm_50,compute_50] \
10      -gencode arch=compute_52,code=[sm_52,compute_52]
11 #      -gencode arch=compute_20,code=[sm_20,sm_21] \ This one is deprecated?
12
13 # This is what I use, uncomment if you know your arch and want to specify
14 # ARCH= -gencode arch=compute_52,code=compute_52
15
16 VPATH=./src/:./examples
17 SLIB=libdarknet.so
18 ALIB=libdarknet.a
19 EXEC=darknet
20 OBJDIR=./obj/
21
22 CC=gcc
23 CPP=g++
Makefile" 105L, 3040C                                    1,1           Top
```

3. Compile

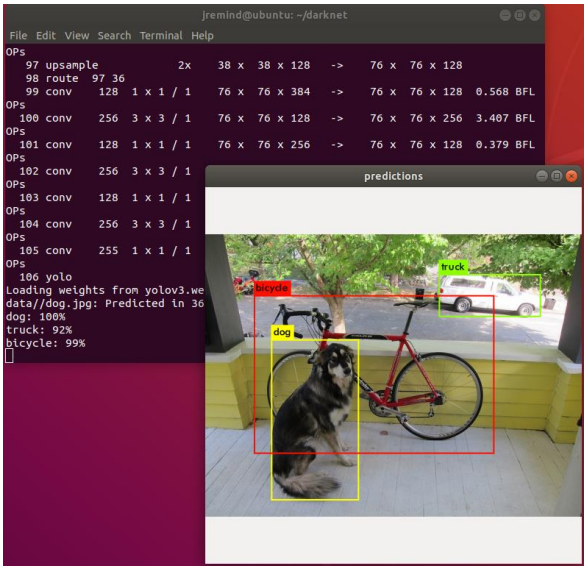
```
make
```

4. Download trained files from YOLO Darknet.

```
$ wget https://pjreddie.com/media/files/yolov3.weights
```

5. Run image example

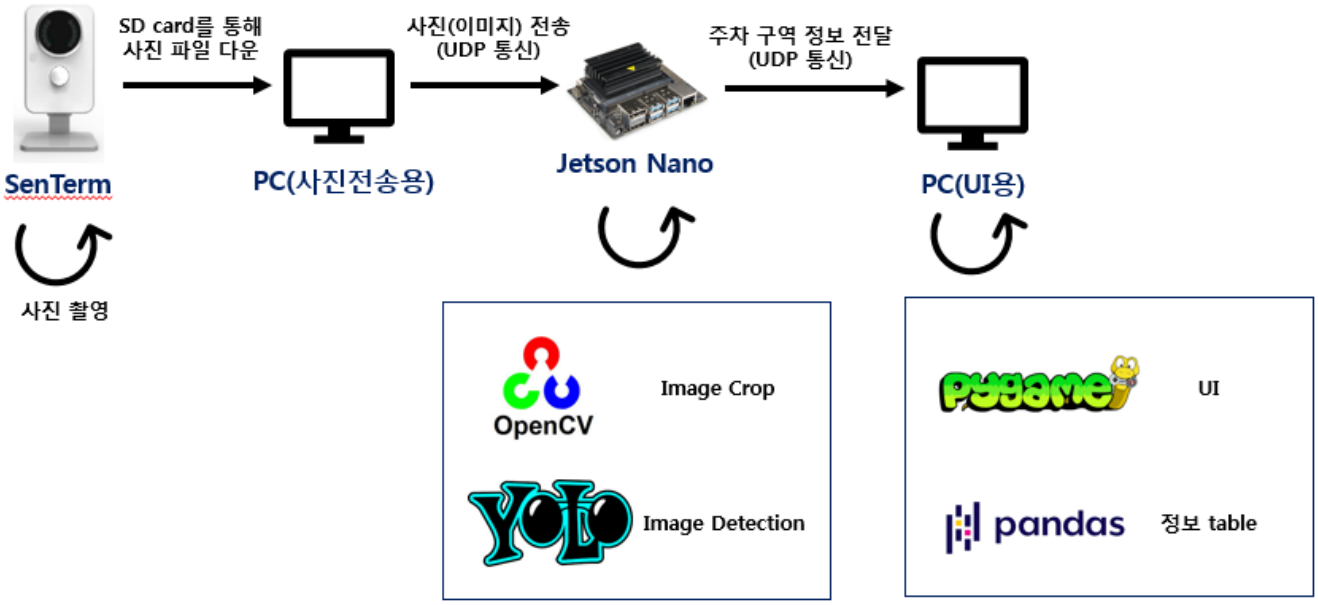
```
$ ./darknet detect cfg/yolov3.cfg yolov3.weights data/dog.jpg
```



2.2. Android Studio 환경 설정 (UI를 App으로 만들려고 했으나 완성 못함)

3. 프로젝트 적용기술

3.1. aParkings-AI 흐름도



- 원래 SenTerm에서 Server로 사진을 전송하고, Server에서 이미지를 특정 크기로 잘라 Jetson Nano로 전송하면, Jetson Nano에서 잘려진 이미지를 이용해 Yolo 수행 후 자동차 인식 (자동차 유무) 결과를 Server로 다시 전송하여 App에서 그 결과로 UI를 형성하는 방식으로 하는 게 맞음

- 현재 Server를 사용하기 어려울 것으로 판단하여 Jetson Nano가 Server 역할 및 Yolo 수행을 담당

3.2. 적용기술

3.2.1. UDP Socket 통신

- - PC → Jetson Nano
  · SenTerm에서 촬영한 이미지를 SD카드를 통해 PC에 저장 후 PC를 통해 Jetson Nano로 전송

- Jetson Nano → PC
  · 이미지 처리에 대한 결과(주차구역 내 주차유무에 대한 정보)를 PC로 전송

3.2.2. 이미지 처리

- OpenCV
  · 이미지의 특정 위치(주차구역)를 자른 이미지 파일 저장

- Yolov3
  · 자른 이미지 파일이 자동차인지 판단하여 자동차 유무 결정
  · 자동차 유무에 대한 리스트 저장

3.2.3. UI (User Interface)

- Pygame
  · 주차구역에 대한 정보(자동차 유무)가 표시된 주차 도면을 스크린으로 출력

- Pandas
  · 주차장 정보 테이블을 데이터 프레임으로 출력

3.2.4. QR code

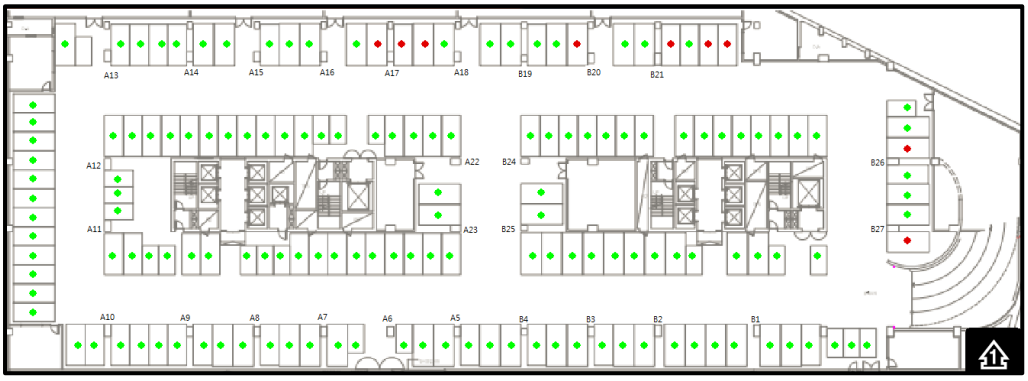
- QR code 생성을 도와주는 파이썬 모듈

4. 프로젝트 시나리오

4.1. aParkings-AI

1. 운전자(user)에게 주차장 도면(UI)을 통해 빈 주차 구역을 보여준다. (초록색 : 주차가능, 빨간색 : 주차불가능)

2. 운전자가 빈 자리에 주차하면 일정 시간 후 주차장 도면에 주차 유무에 대한 표시가 업데이트된다.



(운전자가 차량을 빼면 빨간색은 다시 초록색으로 바뀐다.)

3. 주차장 도면에 자동차 유무가 반영되면 주차장 정보 테이블을 출력한다.

<table>
<tr><th>주차 가능한 차량 수</th><th>주차면 (주차구역으로 잘못 print함)</th></tr>
<tr><td>

주차 가능: 143 / 152

	주차구역	입차	출차
0	0	2020-02-25 14:46:15	2020-02-25 14:51:41
1	4	2020-02-25 14:46:15	0
2	7	2020-02-25 14:46:15	0
3	8	2020-02-25 14:46:15	2020-02-25 14:53:42
4	10	2020-02-25 14:46:15	2020-02-25 14:53:42
5	13	2020-02-25 14:46:15	2020-02-25 14:53:42
6	15	2020-02-25 14:46:15	2020-02-25 14:51:41
7	22	2020-02-25 14:46:15	2020-02-25 14:53:42
8	23	2020-02-25 14:46:15	0
9	1	2020-02-25 14:51:41	0
10	2	2020-02-25 14:51:41	0
11	5	2020-02-25 14:51:41	0
12	12	2020-02-25 14:51:41	2020-02-25 14:53:42
13	9	2020-02-25 14:53:42	0
14	11	2020-02-25 14:53:42	0
15	24	2020-02-25 14:53:42	0

</td><td>

주차 가능: 143 / 152

	주차구역	입차	출차
0	0	2020-02-25 14:46:15	2020-02-25 14:51:41
1	4	2020-02-25 14:46:15	0
2	7	2020-02-25 14:46:15	0
3	8	2020-02-25 14:46:15	2020-02-25 14:53:42
4	10	2020-02-25 14:46:15	2020-02-25 14:53:42
5	13	2020-02-25 14:46:15	2020-02-25 14:53:42
6	15	2020-02-25 14:46:15	2020-02-25 14:51:41
7	22	2020-02-25 14:46:15	2020-02-25 14:53:42
8	23	2020-02-25 14:46:15	0
9	1	2020-02-25 14:51:41	0
10	2	2020-02-25 14:51:41	0
11	5	2020-02-25 14:51:41	0
12	12	2020-02-25 14:51:41	2020-02-25 14:53:42
13	9	2020-02-25 14:53:42	0
14	11	2020-02-25 14:53:42	0
15	24	2020-02-25 14:53:42	0

</td></tr>
<tr><td>입차</td><td>출차</td></tr>
</table>

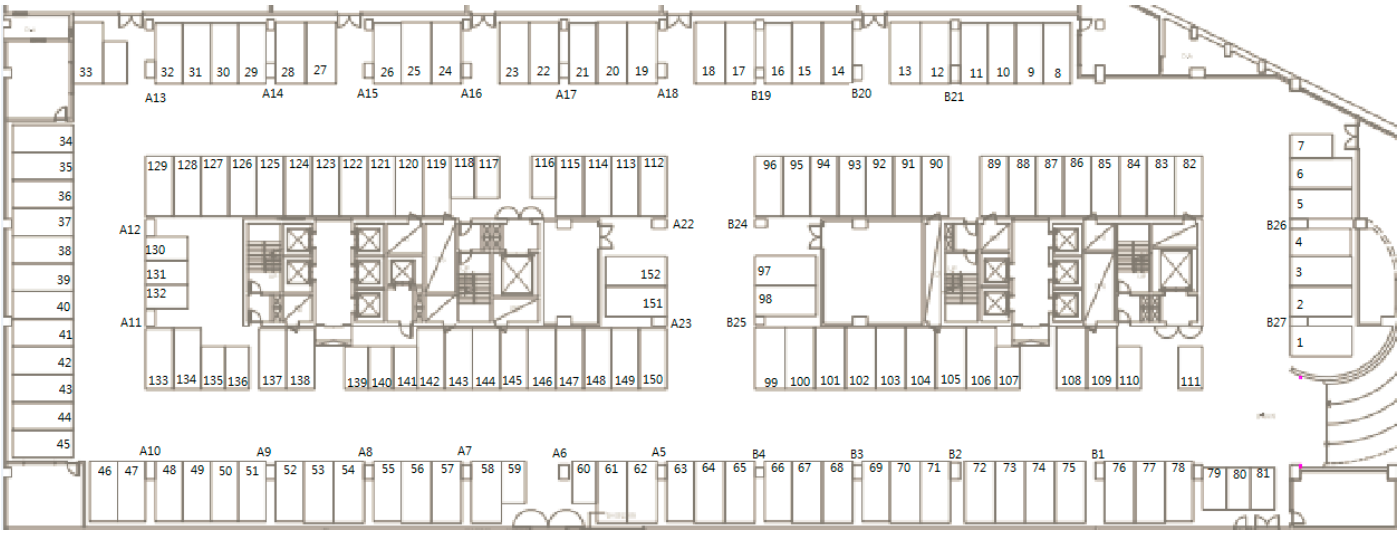
주차 가능: 143 / 152

	주차구역	입차	출차
0	0	2020-02-25 14:46:15	2020-02-25 14:51:41
1	4	2020-02-25 14:46:15	0
2	7	2020-02-25 14:46:15	0
3	8	2020-02-25 14:46:15	2020-02-25 14:53:42
4	10	2020-02-25 14:46:15	2020-02-25 14:53:42
5	13	2020-02-25 14:46:15	2020-02-25 14:53:42
6	15	2020-02-25 14:46:15	2020-02-25 14:51:41
7	22	2020-02-25 14:46:15	2020-02-25 14:53:42
8	23	2020-02-25 14:46:15	0
9	1	2020-02-25 14:51:41	0
10	2	2020-02-25 14:51:41	0
11	5	2020-02-25 14:51:41	0
12	12	2020-02-25 14:51:41	2020-02-25 14:53:42
13	9	2020-02-25 14:53:42	0
14	11	2020-02-25 14:53:42	0
15	24	2020-02-25 14:53:42	0

주차 가능: 143 / 152

	주차구역	입차	출차
0	0	2020-02-25 14:46:15	2020-02-25 14:51:41
1	4	2020-02-25 14:46:15	0
2	7	2020-02-25 14:46:15	0
3	8	2020-02-25 14:46:15	2020-02-25 14:53:42
4	10	2020-02-25 14:46:15	2020-02-25 14:53:42
5	13	2020-02-25 14:46:15	2020-02-25 14:53:42
6	15	2020-02-25 14:46:15	2020-02-25 14:51:41
7	22	2020-02-25 14:46:15	2020-02-25 14:53:42
8	23	2020-02-25 14:46:15	0
9	1	2020-02-25 14:51:41	0
10	2	2020-02-25 14:51:41	0
11	5	2020-02-25 14:51:41	0
12	12	2020-02-25 14:51:41	2020-02-25 14:53:42
13	9	2020-02-25 14:53:42	0
14	11	2020-02-25 14:53:42	0
15	24	2020-02-25 14:53:42	0

여기서 주차구역은 주차면으로 수정 필요



주차면은 도면에서 임의로 값을 주어 설정 여기서 도면에 숫자를 표시하는 방법은 그림판을 이용

4.2. QR code

1. 운전자가 주차 후 근처 기둥에 부착된 QR 코드를 휴대폰으로 촬영한다.

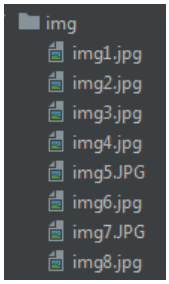
2. 운전자의 휴대폰에 주차 위치 정보를 담은 도면이 출력된다.

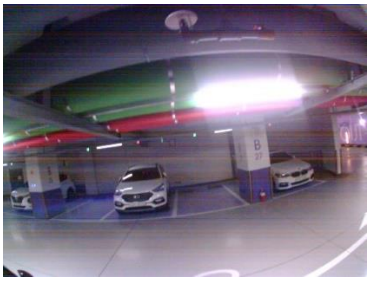
5. 프로젝트 결과

5.1. 문정동 주차장

5.1.1. 사진 전송

- SenTerm에서 촬영한 사진(이미지)를 SD card 이용해 PC에 저장한다.

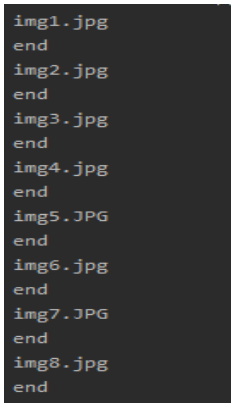




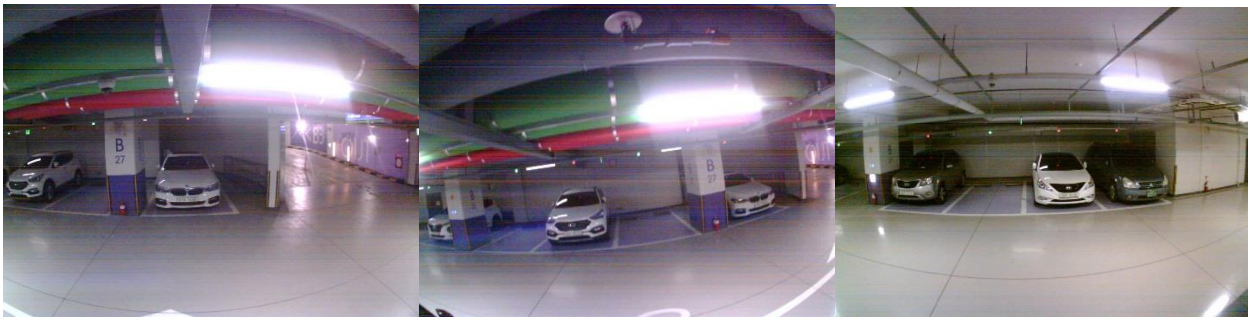
img1 img2 img3

여기서 img 파일은 주차 구역 + 뒤에 숫자는 몇 번 주차구역(임의로 지정)인지를 알려줍니다.

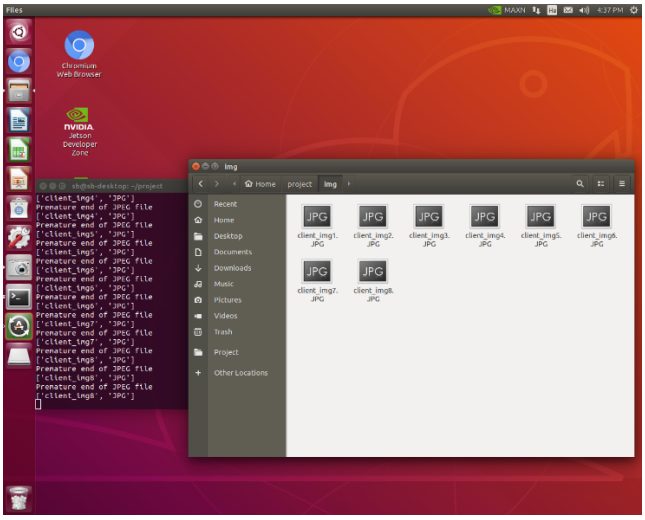
- Client(PC)가 Server(Jetson Nano)로 저장한 이미지를 전송한다.



여기서 img.JPG 파일도 마찬가지로 SenTerm으로 구역들을 지정해 놓고 찍었을 때 한구역을 의미



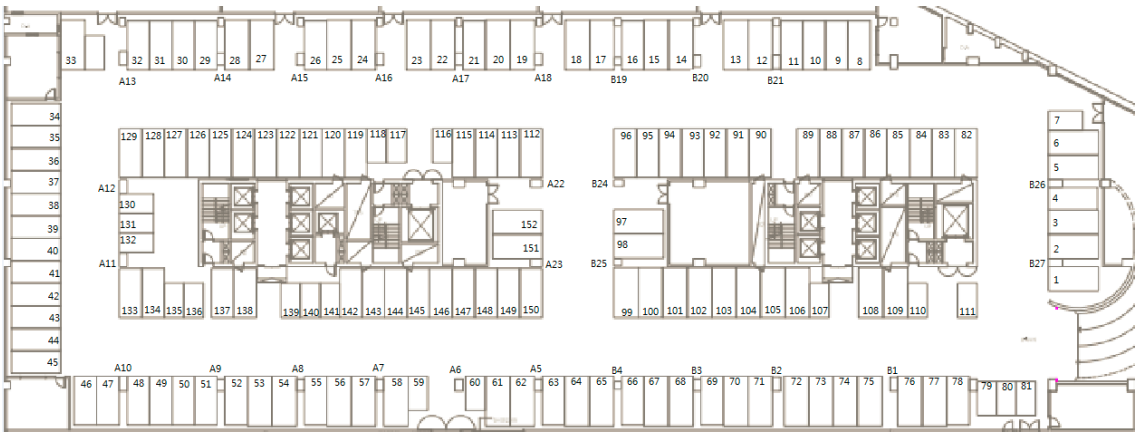
- Server(Jetson Nano)는 전송을 받은 이미지를 저장한다.



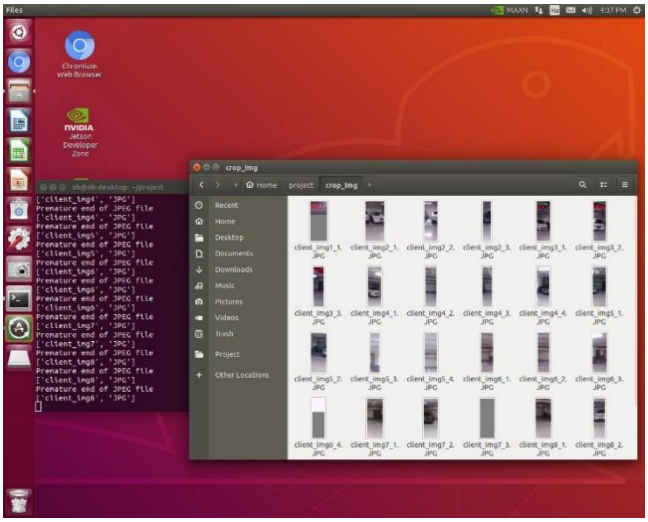
여기서 받은 이미지가 위에서 보낸 구역 사진

5.1.2. 이미지 처리

- Server(Jetson Nano)에서 이미지를 주차면 값(임의로 값을 지정해준)에 맞게 자른다.

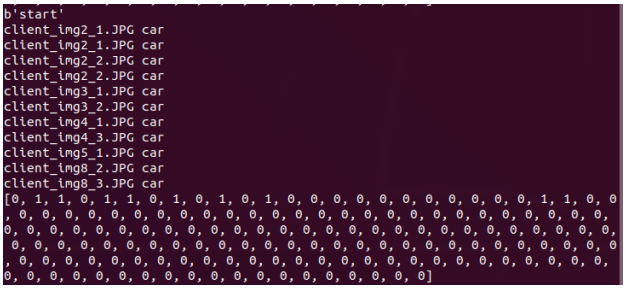


(주차 구역에 대한 좌표는 텍스트 파일에 저장되어 있다.)



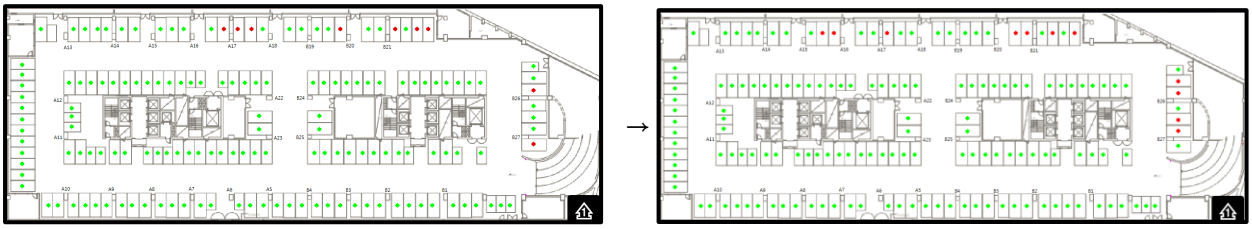
여기서 crop_img는 주차구역으로 받은 img directory의 image file에 주차면을 파악해 주차면 1개씩 자른 이미지 파일을 저장한 directory이다,

- UI Client(PC)가 'start'를 Server(Jetson Nano)로 보내면 Server(Jetson Nano)는 Yolo 수행 후 주차 상황에 대한 정보를 UI Client로 전송한다. (여기서 말하는 정보는 주차 구역 내 자동차 유무에 관한 것이다.)



5.1.3. UI 생성

- Server로부터 정보를 받은 UI Client(PC)는 pygame 모듈을 이용해 주차장 도면에 정보를 표시한다.
  (여기서 말하는 정보는 주차 구역 내 자동차 유무에 관한 것이다.)



- Server로부터 정보를 받은 UI Client(PC)는 pandas 모듈을 이용해 테이블 형태로 정보를 출력한다.

1) 첫번째 사진에서 주차되어 있는 자동차를 인식하고 도면에 표시 후 다음 사진에서 출차가 확인되면 순서대로 입차 출차가 나온다.

2) 첫번째 사진에서 주차되어 있는 자동차가 없고 다음 사진에서 입차가 된 것을 확인하면 테이블 밑에

정보를 추가한다.

=> 현재 주차 구역을 기준으로 순서대로 리스트에 삽입하는 형식인데, 사장님께서 원하시는 형식은 주차면(사진에는 주차구역이라고 표기되어 있음)에 대한 정보를 주차면을 기준으로 뽑기를 바라십니다. 현재는 간략하게 확인하기 위해 pandas 모듈을 사용했지만 사장님께서 원하시는 형식으로 출력하기 위해서는 dictionary 형식을 이용해 pandas를 출력하거나, database를 만드는 것이 더 좋을 것 같습니다. Database는 mysql과 excel을 이용하여 만드는 것이 쉬울 듯합니다. (python 모듈도 많고, 관련 정보도 찾기 쉬울 것입니다.) 현재 check list를 계속 이용하기 위해서는 그냥 새로운 기록 형식을 추가하는 것이 나을 것 같습니다.

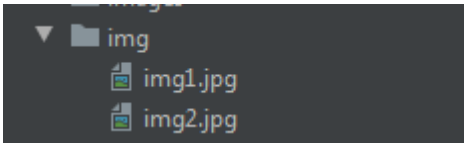
주차 가능: 143 / 152

	주차구역	입차	출차
0	0	2020-02-25 14:46:15	2020-02-25 14:51:41
1	4	2020-02-25 14:46:15	0
2	7	2020-02-25 14:46:15	0
3	8	2020-02-25 14:46:15	2020-02-25 14:53:42
4	10	2020-02-25 14:46:15	2020-02-25 14:53:42
5	13	2020-02-25 14:46:15	2020-02-25 14:53:42
6	15	2020-02-25 14:46:15	2020-02-25 14:51:41
7	22	2020-02-25 14:46:15	2020-02-25 14:53:42
8	23	2020-02-25 14:46:15	0
9	1	2020-02-25 14:51:41	0
10	2	2020-02-25 14:51:41	0
11	5	2020-02-25 14:51:41	0
12	12	2020-02-25 14:51:41	2020-02-25 14:53:42
13	9	2020-02-25 14:53:42	0
14	11	2020-02-25 14:53:42	0
15	24	2020-02-25 14:53:42	0

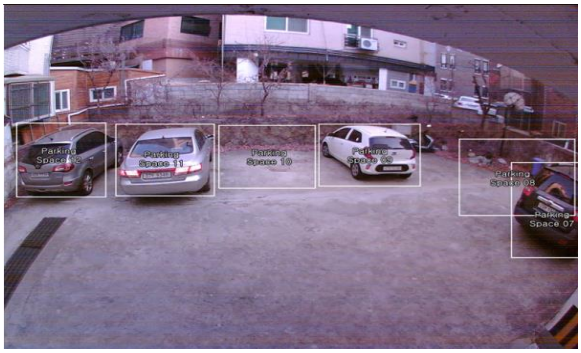
5.2. JWB 주차장

5.2.1 사진 전송

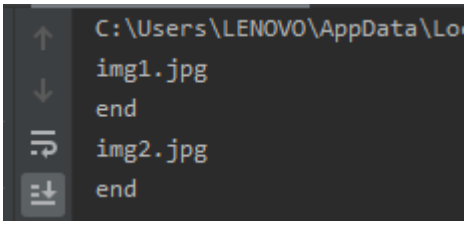
- SenTerm에서 촬영한 사진(이미지)를 SD card 이용해 PC에 저장한다.



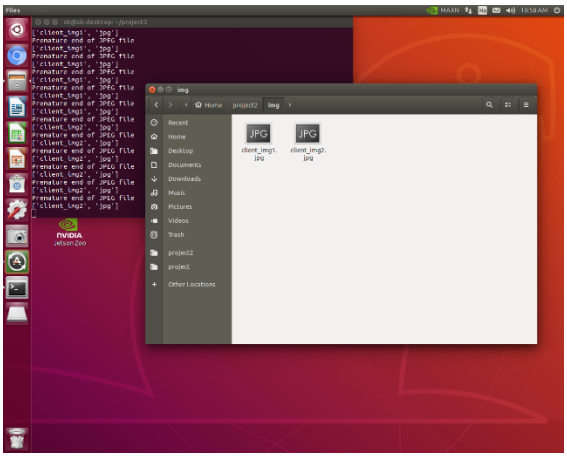
여기서 img는 마찬가지로 주차구역



- Client(PC)가 Server(Jetson Nano)로 저장한 이미지를 전송한다. (SenTerm이 촬영한 주차구역 1장)

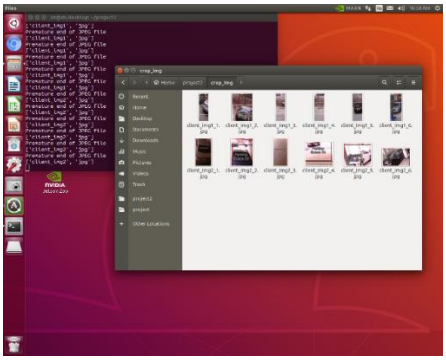


- Server(Jetson Nano)는 전송을 받은 이미지를 저장한다.



5.2.2 이미지 처리

- Server(Jetson Nano)에서 이미지를 주차면에 맞게 자른다.
  (주차면에 대한 좌표는 텍스트 파일에 저장되어 있다.)

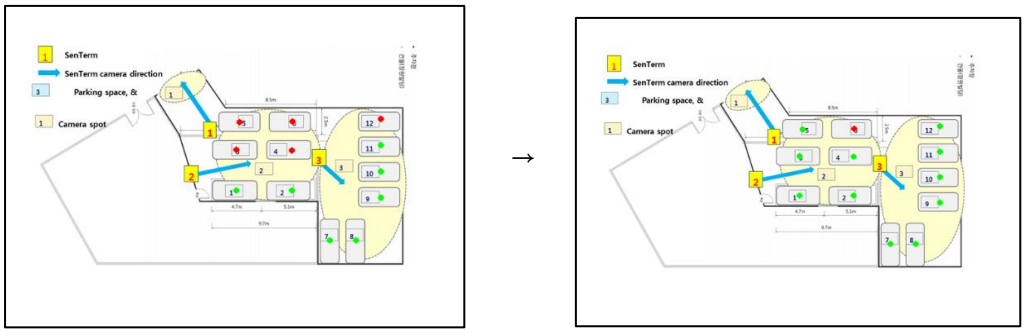


- UI Client(PC)가 'start'를 Server(Jetson Nano)로 보내면 Server(Jetson Nano)는 Yolo 수행 후 주차 상황에 대한 정보를 UI Client로 전송한다. (여기서 말하는 정보는 주차구역 내 사진에서 주차면을 기준으로 자른 crop_img의 자동차 유무에 관한 것이다.)

```
b'start'
client_img1_3.jpg car
client_img1_4.jpg car
client_img1_5.jpg car
client_img1_6.jpg car
client_img2_3.jpg refrigerator
client_img2_6.jpg car
[0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1]
```

5.2.3 UI 생성

- Server로부터 정보를 받은 UI Client(PC)는 pygame 모듈을 이용해 주차장 도면에 정보를 표시한다.
  (여기서 말하는 정보는 주차 구역 내 자동차 유무에 관한 것이다.)



1번 1번 2번 2번

야외 주차장 인식률 매우 저조 이부분에 대한 개선 필요

1) SenTerm 위치 변경

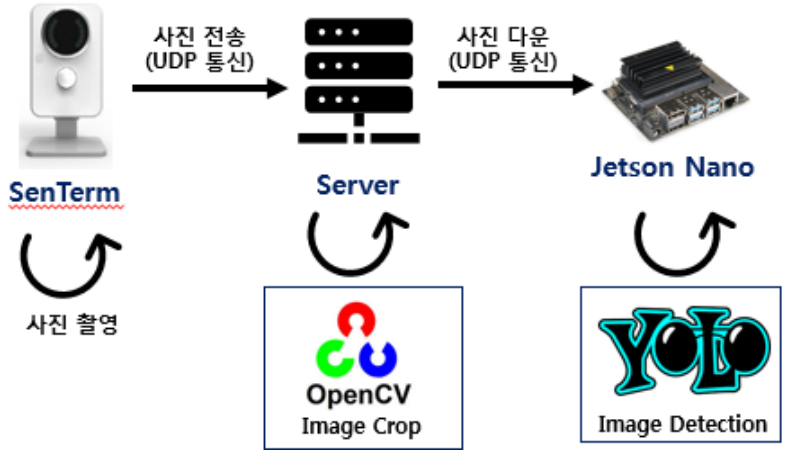
2) SenTerm으로 촬영한 주차구역으로 받은 이미지를 crop 할 때 좌표 값에 따른 인식률 실험

- Server로부터 정보를 받은 UI Client(PC)는 pandas 모듈을 이용해 테이블 형태로 정보를 출력한다.

```
주차 가능: 11 / 12
   주차구역                 입차                 출차
0     2  2020-02-27 11:05:37  2020-02-27 11:08:08
1     3  2020-02-27 11:05:37  2020-02-27 11:08:08
2     4  2020-02-27 11:05:37  2020-02-27 11:07:19
3     5  2020-02-27 11:05:37                    0
4    11  2020-02-27 11:05:37  2020-02-27 11:07:19
```

6. 프로젝트 개선사항

6.1. SenTerm과 Jetson Nano의 UDP 통신 구축



- 현재 SenTerm에서 촬영한 사진을 SD card를 통해 PC로 옮긴 후 전송

- SenTerm이 server로 사진을 전송하면 Jetson Nano는 server에서 사진을 전송을 받아 Yolo를 수행하는 방식으로 수정 필요

- SenTerm과 연결된 server는 C언어로 이루어져 있음

6.2. QR 코드 테이블

- QR 코드를 촬영하면 촬영한 순서대로 데이터를 기록할 수 있는 애플리케이션 제작이 필요하다.

- 데이터 기록 방식은 Queue 형식으로, 일정 기간이 지난 오래된 데이터나 저장된 데이터 수가 많은 경우 가장 오래된 데이터부터 저장하는 형식이다.

6.3. Yolov3의 이미지 처리 속도 향상

- 이미지 처리 속도가 Raspberry pi camera로 촬영하여 실시간으로 처리하는 속도보다 느리다.

- SenTerm 보드에 OpenCV를 설치할 수 없기 때문에 영상에서 Yolo 수행이 불가능하므로 다른 방안을 모색해야 할 것 같다.

7. 코드

7.1. SenTerm에서 촬영한 사진을 저장한 PC(client)에서 Jetson Nano(server)로 이미지(사진) 전송

- client_Senterm.py (server_Senterm.py가 실행되어 있어야 함)

```
import socket
import os
import time

UDP_IP = '222.107.177.42' # server IP num
UDP_PORT = 6667 # server port num

path = './img' # SenTerm 에서 촬영한 사진이 저장된 directory 경로
img_list = os.listdir(path) # path 에 저장된 file list 를 저장 (list 형으로)
img_list.sort() # 받아온 이미지를 정렬

clientSock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

for file_name in img_list: # 이미지 리스트가 저장된 img_list 의 이름을 file_name 이라는 변수로 받음
    clientSock.sendto(file_name.encode(), (UDP_IP, UDP_PORT))
    # server 로 file(image) name 전송, client 의 IP, PORT 를 실어서 전송함.
```

```
    data_transferred = 0
    with open('./img/' + file_name, 'rb') as f: # file(image) open
        print(file_name) # file name 확인용
        try:
            data = f.read(1024) # file(image)에 적힌 내용 read 하여 data 변수로 저장
            while data: # read 한 data 가 없을 때까지 data read 반복
                data_transferred += clientSock.sendto(data, (UDP_IP, UDP_PORT))
                data = f.read(1024)
                time.sleep(0.0001) # data 가 중간에 손실되는 것을 막기 위해 약간의 delay 를 넣음
        except Exception as e:
            print(e)
    clientSock.sendto('end'.encode(), (UDP_IP, UDP_PORT))
    # client 는 server 로 image 를 다 보내면 'end'를 전송하여 image 하나를 다 보냄을 알려줌
    print('end') # image 하나 전송 완료 확인용
```

7.2. Jetson Nano는 Client(PC)로부터 전송을 받은 이미지 처리

- server_SenTerm.py

```
import socket
import imgCrop # 같은 directory 내에 있는 imgCrop.py 를 import

UDP_IP = '222.107.177.42' # server IP num
UDP_PORT = 6667 # server port num

serverSock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
serverSock.bind((UDP_IP, UDP_PORT))
crop = imgCrop.CROP() # imgCrop.py 의 CROP class

def s():
   while True:
      data_transferred = 0
      title, addr = serverSock.recvfrom(1024) # client 에게 data 및 주소(IP, PORT)를 받음
      print ("Messsage: ", title) # 처음 오는 데이터는 file(image) name
      if not title:
         print("Error") # title 이 없는 경우(전송을 받지 못한 경우) error
      else :
         data, addr = serverSock.recvfrom(1024) # 여기부터는 image 에 대한 정보를 받음
         with open('./img/'+ title.decode(), 'wb') as f: # title 로 file 을 생성하여 open
            try:
               while data:
                  f.write(data) # file 에 읽어온 data 를 기록
                  data_transferred += len(data)
                  data, addr = serverSock.recvfrom(1024)
                  if data == b'end': # client 에서 'end'를 전송하면 이미지 하나를 다 보냈다는 의미
                     print('end')
                     crop.main() # CROP class 의 main 함수를 실행하여 image crop
                     s() # 함수를 처음부터 다시 실행하여 다음 image 를 받아옴
               print("while end")
            except Exception as e:
               print(e)

s() # server 에 대한 함수 실행
```

- imgCrop.py

```
import os


class CROP:
    def main(self):
        # Color
        red = (0, 0, 255)

        # Read image file list in directory
        path_img = './img'  # Set image directory path
        img_file_list = os.listdir(path_img)  # Read file list in path
        img_file_list.sort()

        # Store light status (0:Green, 1:Red)
        light_list = [0] * 3

        cnt = 0 # 좌표용 count

        # Open vector_file_list
        vector_list = []
        f = open("./vector/vec.txt", "r")  # 주차구역에 대한 좌표가 기록된 vec.txt open

        line = f.readline() # file 에 대한 정보를 읽어 옴

        vector_list = line.split() # 공백을 기준으로 쪼개서 vector_list 에 저장
        vector_list = [int(i) for i in vector_list] # 저장된 좌표가 str 형이므로 int 형으로 변경

        def extractor(img, cnt): # image crop 한 후 return
            cropped_img = img[vector_list[cnt + 1]:vector_list[cnt + 3],
vector_list[cnt]:vector_list[cnt + 2]]
            return cropped_img

        for j in range(0, len(img_file_list)): # img directory 에 저장된 image 이름을 불러옴
            count = 0 # 특정 image 의 크롭된 이미지들에게 구별된 이름을 주기 위해 사용
            while 1:
                count += 1
                if cnt > len(vector_list) - 1:
                # cnt 가 vector list 보다 큰 경우 더 이상 좌표 값이 없으므로 while 문 break
                    break
                if vector_list[cnt] == -1:
                # -1 이 나온 경우 한 이미지에 대한 좌표 값을 다 읽은 것으로 보고 while 문 break
                    cnt += 1
                    break
                image = ("./img/" + str(img_file_list[j])) # image 에 대한 path 설정
                img = cv.imread(image, cv.IMREAD_COLOR) # OpenCV 를 이용해 image 를 읽어 옴
                cv.rectangle(img, (vector_list[cnt], vector_list[cnt + 1]),
                             (vector_list[cnt + 2], vector_list[cnt + 3]), red, 3)
                # cv.rectangle 을 이용해 이미지에 사각형을 그려 줌
                img = extractor(img, cnt) # image crop
                file_name = img_file_list[j].split('.')
                print(file_name)
                file_name_path = './crop_img/' + str(file_name[0]) + '_' + str(count) + '.' +
str(file_name[-1]) # 크롭된 이미지는 crop_img directory 에 저장해야하므로 path 및 file name 설정
                cv.imwrite(file_name_path, img) # 크롭된 이미지 저장
                cnt = cnt + 4
```

```
if __name__ == '__main__':
    crop = CROP() # CROP class 사용 표시
    crop.main() # CROP class의 main 함수 호출
```

7.3. UI Client(PC)가 Server(Jetson Nano)로 'start'를 전송하면 Server에서 Yolo 수행 후 결과를 UI Client에게 전송

- server_UI.py

```
import socket
import yolo_object_detection

UDP_IP = '222.107.177.42' # server IP num
UDP_PORT = 6668 # server port num
                # UI용 server는 SenTerm 사진용과 port를 다르게 줘서 동시에 실행 가능하도록 만듦

serverSock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
serverSock.bind((UDP_IP, UDP_PORT))
detection = yolo_object_detection.DETECT() # yolo_object_detection의 DETECT class 호출

while True:
    data, addr = serverSock.recvfrom(1024) # data, address(IP, PORT)를 전송 받음
    if data == b'start': # 전송 받은 data가 start인지 확인 (bit 단위 전송이므로 b'start' 라고 씀)
        parkingList = detection.main()#start를 받은 경우 DETECT class의 main 함수 수행(yolo 수행)
        message = ''
        for i in parkingList: # main 함수를 통해 받은 car 유무에 대한 list를
            message += str(i) # 전송을 위해 string으로 형 변환
        serverSock.sendto(message.encode(), addr) # Client 주소로 인코딩하여 전송
```

- yolo_object_detection.py

```
import cv2
import numpy as np
import os

class DETECT:
    def main(self):
        # Load Yolo
        net = cv2.dnn.readNet("yolov3.weights", "yolov3.cfg")
        classes = []
        with open("coco.names", "r") as f:
            classes = [line.strip() for line in f.readlines()]
        layer_names = net.getLayerNames()
        output_layers = [layer_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]
        colors = np.random.uniform(0, 255, size=(len(classes), 3))

        # Loading image
        parking_list = [0]*152 # list 길이는 주차 공간 수와 같아야 함.
        path_crop_img = './crop_img'  # Set image directory path
        img_list = os.listdir(path_crop_img)  # Read file list in path
        img_list.sort()
        for l in range(len(img_list)):
            img = cv2.imread('./crop_img/'+img_list[l]) # image read
            #img = cv2.resize(img, None, fx=0.4, fy=0.4)
            height, width, channels = img.shape
```

```
            # Detecting objects
            blob = cv2.dnn.blobFromImage(img, 0.00392, (416, 416), (0, 0, 0), True, crop=False)
            net.setInput(blob)
            outs = net.forward(output_layers)
            # Showing informations on the screen
            class_ids = []
            confidences = []
            boxes = []
            for out in outs:
                for detection in out:
                    scores = detection[5:]
                    class_id = np.argmax(scores)
                    confidence = scores[class_id]
                    if confidence > 0.5:
                        # Object detected
                        center_x = int(detection[0] * width)
                        center_y = int(detection[1] * height)
                        w = int(detection[2] * width)
                        h = int(detection[3] * height)
                        # Rectangle coordinates
                        x = int(center_x - w / 2)
                        y = int(center_y - h / 2)
                        boxes.append([x, y, w, h])
                        confidences.append(float(confidence))
                        class_ids.append(class_id)
            indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)
            #print(indexes)
            font = cv2.FONT_HERSHEY_PLAIN
            for i in range(len(boxes)):
                if i in indexes:
                    x, y, w, h = boxes[i]
                    label = str(classes[class_ids[i]]) # label 불러옴
                    print(img_list[l], label)
                    if label == 'car': # label 이 car 인 경우
                        parking_list[l] = 1 # parking_list 에 1 넣음
                    # color = colors[i]
                    # cv2.rectangle(img, (x, y), (x + w, y + h), color, 2)
                    # cv2.putText(img, label, (x, y + 30), font, 3, color, 3)
        print(parking_list)
        return parking_list # parking_list 전송 (car 인 경우 1, 아닌 경우 0)


        # cv2.imshow(img_list[l], img)
        # cv2.waitKey(0)
        # cv2.destroyAllWindows()

if __name__ == '__main__':
    detect = DETECT()
    detect.main()
```

7.4. Server(Jetson Nano)에서 Yolo 수행 결과값을 받은 UI Client(PC)는 pygame과 pandas를 이용해 UI 생성

- client_UI.py

```
import pandas as pd
import datetime
import time
```

```
import pygame
from pygame import *
from pygame.locals import *
import numpy
import socket


UDP_IP = '222.107.177.42' # server IP num
UDP_PORT = 6668 # server port num
clientSock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

# 주차 도면 위의 circle 생성 좌표
parkingList = [(1180,303), (1180,269), (1180,244), (1180,218), (1180,183), (1180,156),
(1180,129), (944,46), (919,46), (894,46), (870, 46), (836,46), (810,46), (747,46), (720,46),
(695,46), (660,46), (633,46), (573,46), (548,46), 517,46), (486,46), (458,46), (396,46),
(370,46), (344,46), (288,46), (257,46), (222,46), (203,46), (175,46),(149,46), (76,46),
(34,126), (34,148), (34,172), (34,198), (34,222), (34,248),(34,273), (34,297), (34,323),
(34,347), (34,372), (34,397), (93,440), (114,440), (149,440), (176,440),(199,440), 223,440),
(258,440), (280,440), (309,440), (345,440), (370,440), (397,440), (434,440),
(457,440),(520,440), (545,440), (575,440), (605,440), (633,440), (661,440), (694,440),
(719,440), (747,440), (780,440),(807,440), (835,440), (875,440), (901,440), (928,440),
(953,440), (1000,440), (1026,440), (1051,440), (1085,440),(1107,440), (1127,440), (1061,166),
(1036,166), (1011,166), (985,166), (962,166), (937,166), (914,166), (887,166),(835,166),
(809,166), (785,166), (761,166), (736,166), (710,166), (686,166), (700, 240), (700,
270),(689,323),(712,323), (741,323), (766,323), (797,323), (822,323), (849,323), (878,323),
(898,323), (956,323), (984,323),(1009,323), (1062,323), (582,166), (558,166), (533,166),
(505,166), (483,166), (434,166), (412,166), (390,166),(364,166), (339,166), (315,166),
(290,166), (265,166), (238,166), (214,166), (188,166), (164,166), (139,166),(145,223),
145,241), (145,264), (138,323), (164,323), (186,323), (210,323), (242,323), (264,323),
(314,323),(338,323), (358,323), (382,323), (408,323), (429,323), (458,323), (479,323),
(507,323), (529,323), (555,323),(580,323), 565,240), (565,270)]

screen = pygame.display.set_mode((1332,477), pygame.DOUBLEBUF) # screen 에 대한 설정

class PROCESSING:
    def main(self):
        pygame.init()
        img = pygame.image.load("./parkingLotImg/MoonParkingLot.png") # 배경 이미지 생성
        img = pygame.transform.scale(img, (1332, 477)) # 배경 이미지 크기 변경 (screen 에 맞게)
        screen.blit(img, (0, 0)) # screen 에 배경 이미지 삽입
        for i in range(len(parkingList)): # parking list 길이만큼 circle 삽입
            pygame.draw.circle(screen, (0, 255, 0), parkingList[i], 5)
        pygame.display.flip() # 지금까지 변한 모든 설정을 screen 위에 반영
        carTable = [] # 주차면, 입차 시간, 출차 시간에 대한 리스트
        carArray = numpy.array(carTable) # 특정 열을 받아 오기 위해 numpy.array 사용

        running = True # while 문의 수행 상태 변화를 위해 running 변수 사용

        while running:
            for event in pygame.event.get(): # 종료에 관한 입력 event 를 받아 옴
                if event.type == pygame.QUIT: running = False # 마우스로 창 닫기를 클릭하면 종료
                if event.type == pygame.KEYDOWN: # 키보드를 누르는 경우
                    if (event.key == pygame.K_ESCAPE): running = False # 누른 키보드가 esc 면 종료
                pygame.display.flip() # 지금까지 변한 모든 설정을 screen 위에 반영
                clientSock.sendto("start".encode(), (UDP_IP, UDP_PORT)) # car 유무에 대한 list 를
받기 위해 server 로 'start'를 인코딩하여 전송
                data, addr = clientSock.recvfrom(1024) # server 가 전송한 data 를 받음
```

```
            yolo_detection = data.decode() # yolo 수행 결과로 보낸 data 디코딩
            yolo_detection = [int(i) for i in yolo_detection] # list 구조 및 int 형으로 변경
            check = [0] * len(yolo_detection) # in, not in check
            print("주차 가능:", len(parkingList) - sum(yolo_detection), "/",
len(parkingList)) # 현재 주차 가능한 수에 대해 출력 (sum(yolo_detection)은 리스트의 숫자 합을 의미)

            for i in range(len(yolo_detection)):
                carArray = numpy.array(carTable)
                if yolo_detection[i]: # yolo_detection[i] == 1 인 경우 (car 인 경우)
                    if len(carTable) == 0:
                        check[i] = 1 # check 가 1 이면 내용 기록이 필요함을 의미
                    else:
                        for j in range(len(carTable)):
                            carArray = numpy.array(carTable)
                            if str(i) not in carArray[:, 0]: # 기록된 주차면이 아닌 경우
                                check[i] = 1 # 새로 들어온 차이므로 기록 필요
                            elif carArray[:, 0][j] == str(i) and carArray[:, 2][j] == str(0):
                                # 해당 주차면의 출차 기록이 없는 경우
                                check[i] = 0 # 기존에 있던 차이므로 기록 필요 X
                            elif carArray[:, 0][j] == str(i) and carArray[:, 2][j] != str(0):
                                # 해당 주차면의 출차 기록이 있는 경우
                                check[i] = 1 # 새로 들어온 차이므로 기록 필요
                else: # yolo_detection[i] == 0 인 경우 (car 가 아닌 경우)
                    for j in range(len(carTable)):
                        if carTable[j][0] == i and carTable[j][2] == 0: # 입차 기록이 있는 경우
                            now = datetime.datetime.now()
                            carTable[j][2] = now.strftime('%Y-%m-%d %H:%M:%S') # 출차 시간 기록
                    pygame.draw.circle(screen, (0, 255, 0), parkingList[i], 5) # Create green
                    # cv2.circle(img, parkingList[i], 5, (0, 255, 0), -1)  # Create green
circle
            for i in range(len(check)): # check list 확인을 위한 반복문
                if check[i]: # check[i] == 1 이면
                    now = datetime.datetime.now()
                    carTable.append([i, now.strftime('%Y-%m-%d %H:%M:%S'), 0])
                    #주차면, 입차 시간 기록
                    pygame.draw.circle(screen, (255, 0, 0), parkingList[i], 5) # Create rea
            df = pd.DataFrame(carTable, columns=['주차면', '입차', '출차']) #DataFrame Update
            print(df) # 테이블 형식으로 내용 출력
        pygame.quit() # running == False 인 경우 while 문을 종료하면서 프로그램 종료

if __name__ == '__main__':
    processing = PROCESSING()
    processing.main()
```