MATLAB EXPO 2020

다수 무인기와 무인선을 이용한 조난자, 적조, 유출유 수색 알고리즘 개발

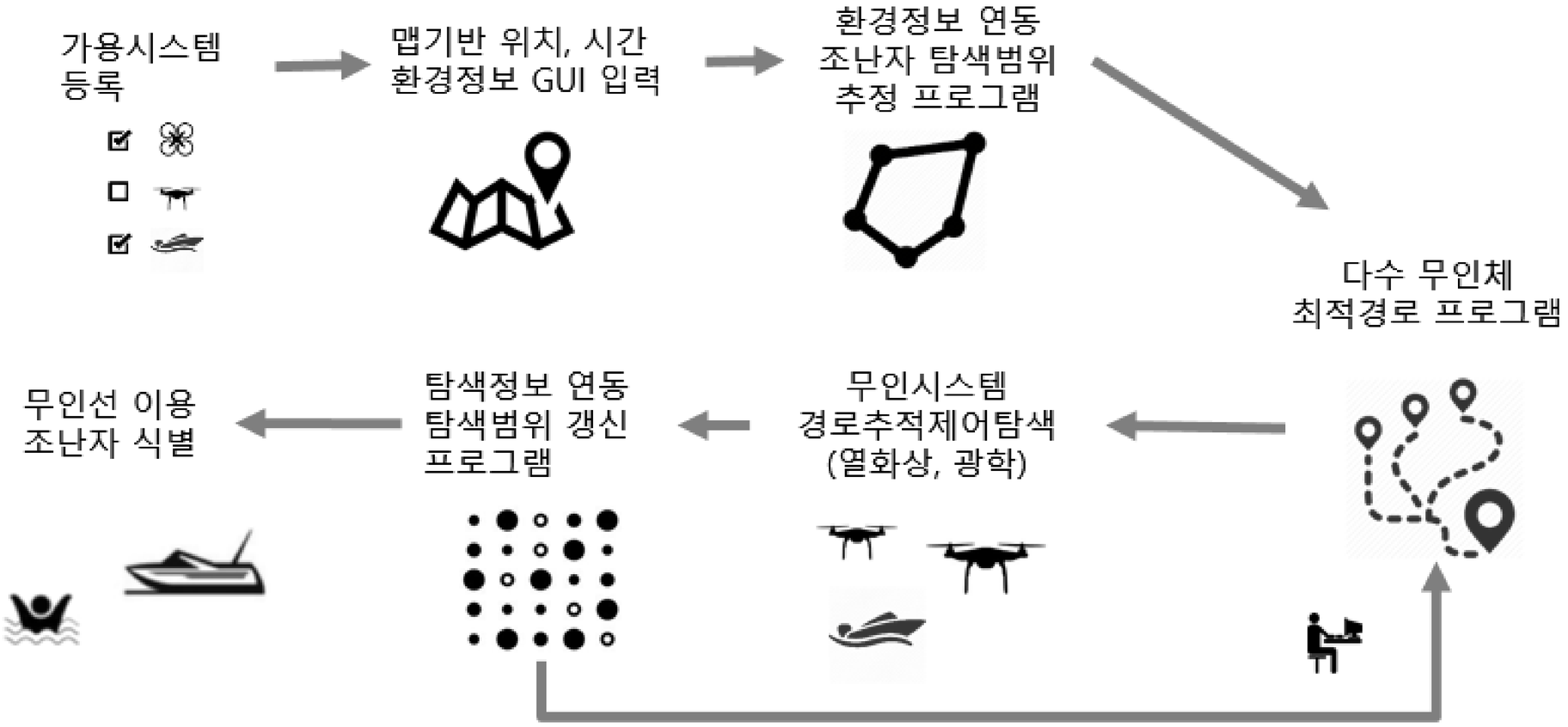
2020. 07. 02.

선박해양플랜트연구소
해양시스템연구본부

김기훈

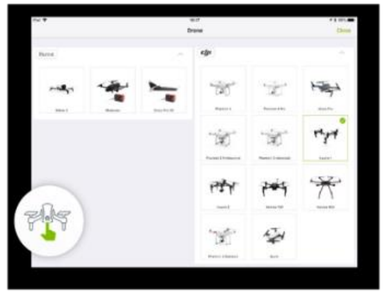
KRISO
선박해양플랜트연구소

다수무인체 운용 시나리오



무인기를 이용한 자동 수색 알고리즘 개념도

1



기체 선택

2

임무 계획 설정

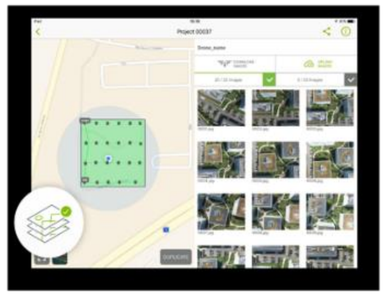
3

수색경로 설정 및 조정

4

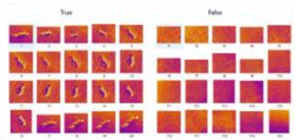
자동 비행 및 이미지 획득

5



결과 리뷰

6

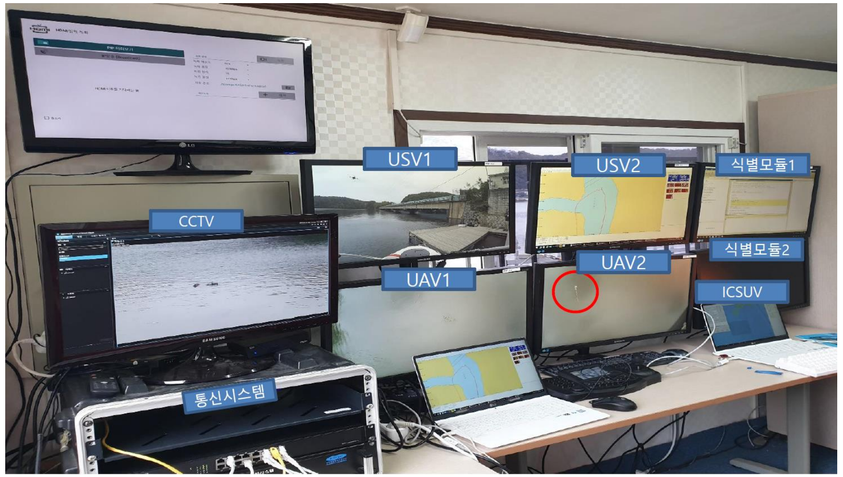


이미지 분석

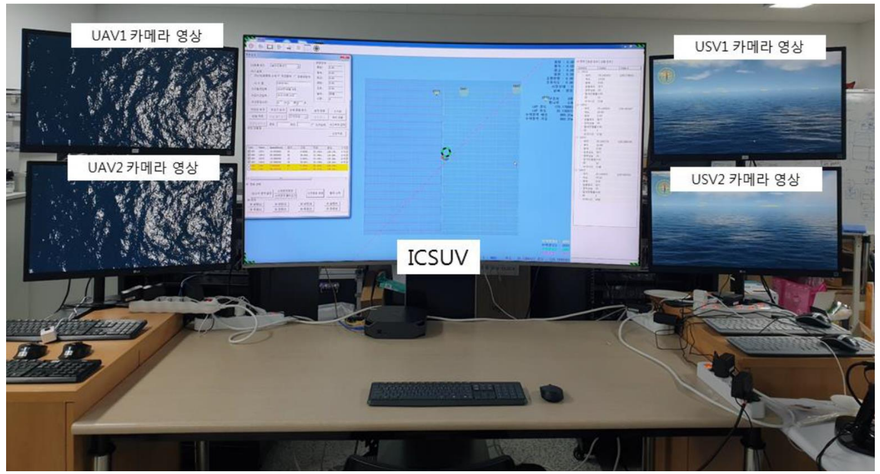
다수 무인기-무인선 이용 수색알고리즘 개발

무인기 1

무인기 2



통합 관제실



관제 시뮬레이터

KRISO USV

King fisher

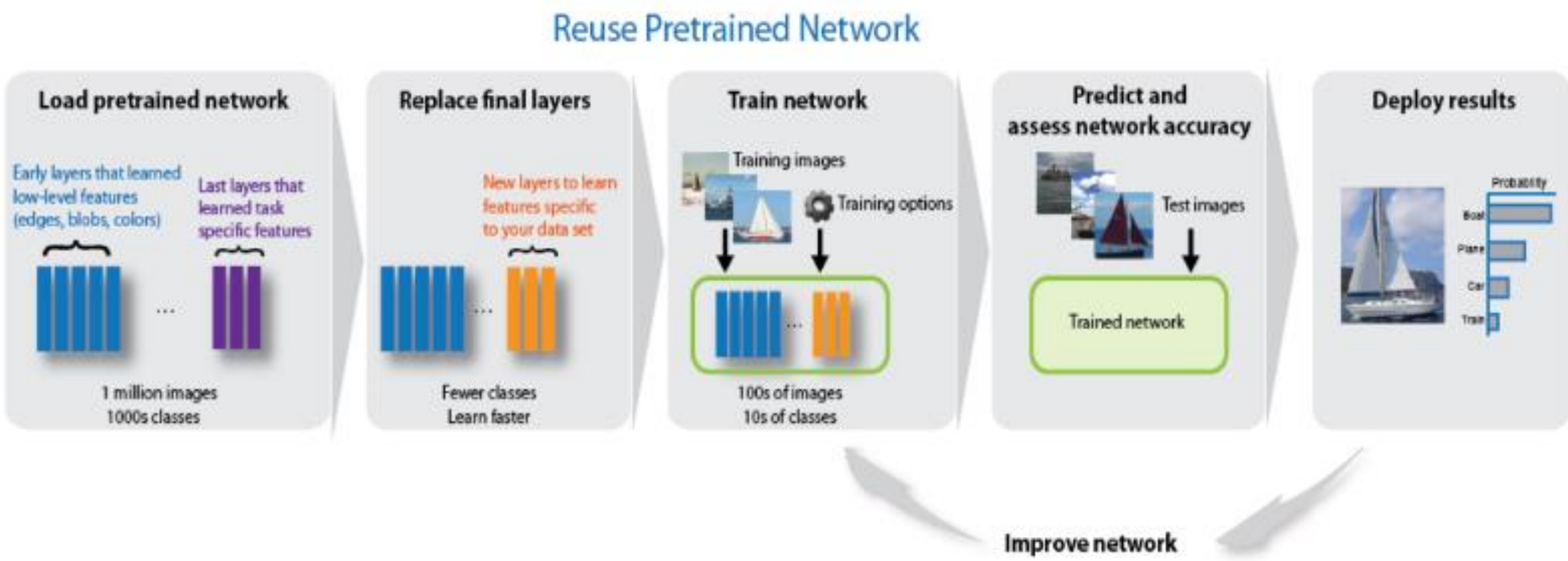
콤보호

Wam - V

조난자 역할 마네킹

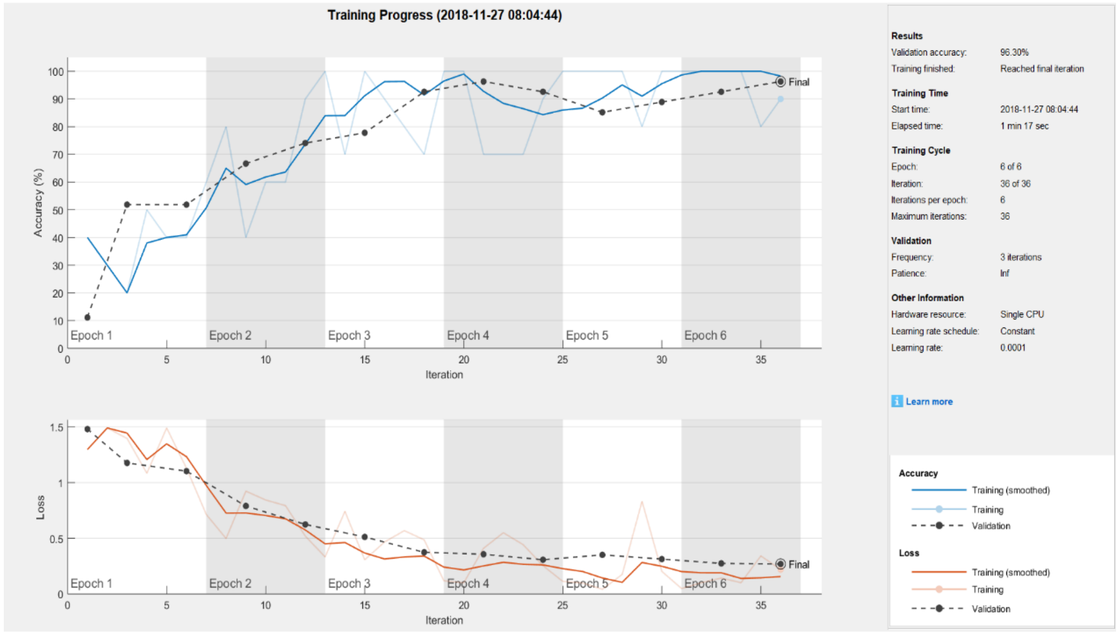
조난자/적조/유출유 식별알고리즘
Using Transfer Learning

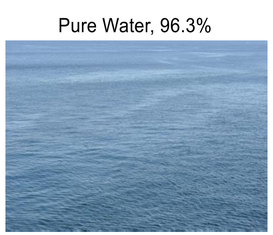
Transfer Learning을 이용한 조난자/적조/유출유 이미지 식별 알고리즘 개발



조난자/적조/유출유 식별 알고리즘 개발

GoogleNet Transfer Learning 학습 Process

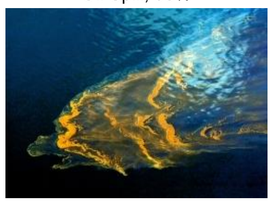



Pure Water, 96.3%


Red Tides, 87.5%


Oil Spill, 95.2%


Pure Water, 90.9%

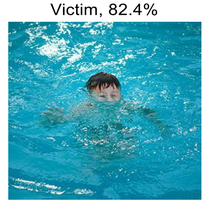

Oil Spill, 95%


Oil Spill, 93.9%

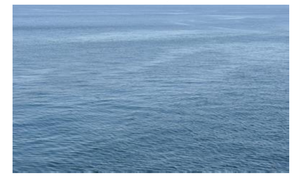

Oil Spill, 99.9%

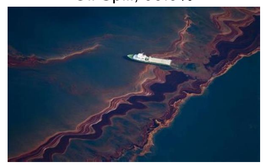

Red Tides, 85.5%


Red Tides, 86.5%


Victim, 82.4%


Oil Spill, 95.2%


Pure Water, 96.3%


Oil Spill, 99.9%

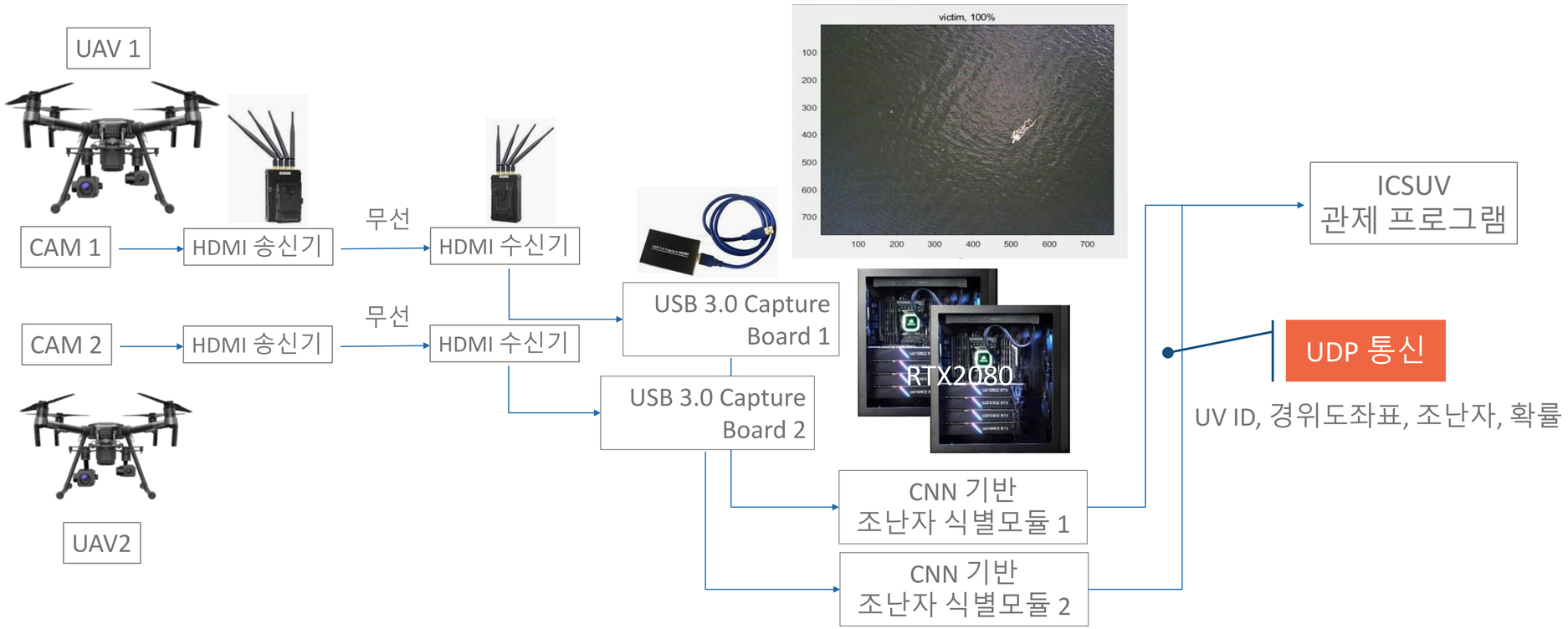

Red Tides, 87.5%


Victim, 90.1%


Oil Spill, 70.5%

딥러닝 기반 조난자 식별 알고리즘용 HW 구성 – 현장 실험, 시뮬레이터

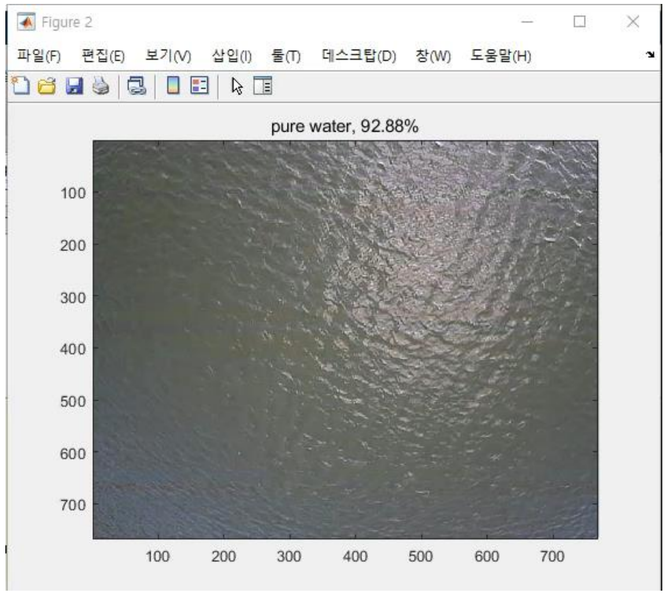
- ❖ 전년도 개발 전이학습 알고리즘에 비디오 하드웨어 연동 구현
- ❖ ICSUV(STR)과 UDP 통신으로 연결 OS, 코딩 언어 차이 극복



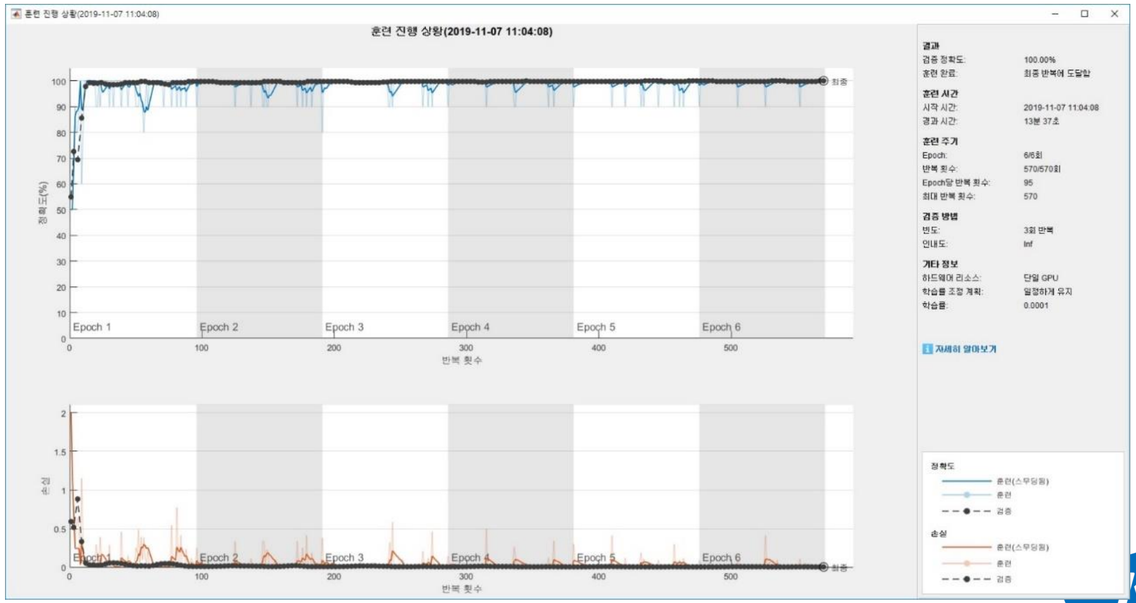
딥러닝 기반 조난자 식별 알고리즘용 SW 개발 – 현장 실험, 시뮬레이터용 2가지

- ❖ GoogleNet을 Base Network으로 하는 매트랩 기반 Transfer Learning(전이학습) 알고리즘 구현
- ❖ 전이학습의 장점 : 적은 수의 데이터로 우수한 성능 확보 가능, 학습 시간이 적게 걸림
- ❖ 일반 배경과 조난자 마네킹 라벨에 대한 이미지 학습(Pure Water: 1139장, 마네킹 : 610장)
- ❖ 학습데이터 70%, 검증데이터 30% 설정 후 Confusion Matrix 검증 정확도 98.3%
- ❖ 학습 시간 : RTX2060 Single GPU 기준 13분 소요
- ❖ 성능시험 결과 : 마네킹 4개 중 4개 모두 식별
- ❖ 날씨(조도, 바람) 영향에 따라 Pure water 식별 성능이 바뀜. 성능 향상에 노력.

식별 성능 결과 비디오

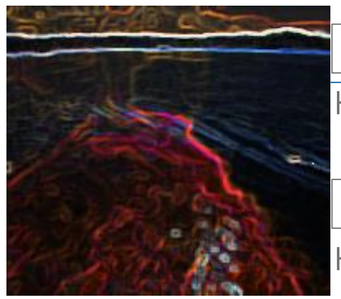


훈련진행상황



딥러닝 기반 적조, 유출유 식별알고리즘 HW 구성 - 시뮬레이터

- 적조, 유출유는 실해역 시연에 제약 조건 존재 → 시뮬레이터로 구현
- 2018년도 개발 전이학습 알고리즘에 시뮬레이터 이미지 연동
- ICSUV(STR)과 UDP 통신으로 연결으로 OS, 코딩 언어 차이 극복



적조, 유출유
시뮬레이터

UAV 1 CAM

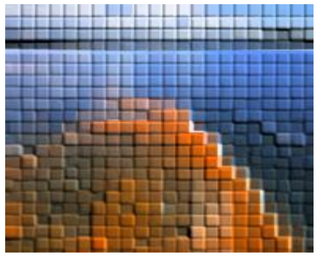
HDMI 유선

UAV 2 CAM

HDMI 유선

USB 3.0 Capture
Board 1

USB 3.0 Capture
Board 2



ICSUV
관제 프로그램

UDP 통신

UV ID, 위경도 좌표, 식별 종류(유출유, 적조), 확률

적조.유출유
식별모듈 1

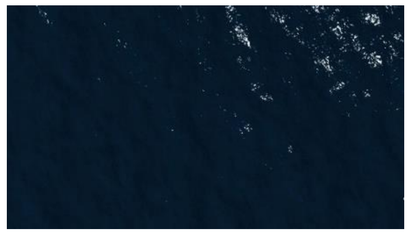
적조.유출유
식별모듈 2

Deep Learning으로
적조, 유출유 식별

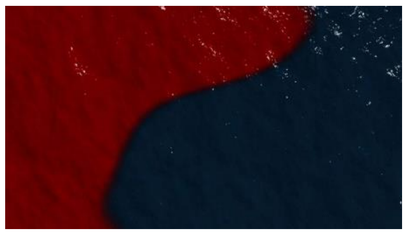
딥러닝 기반 적조, 유출유 식별 알고리즘용 SW 개발 – 시뮬레이터용 2가지

- GoogleNet을 Base Network으로 하는 매트랩 기반 Transfer Learning(전이학습) 알고리즘 구현
- 전이학습의 장점 : 작은 수의 데이터로 우수한 성능 확보 가능, 학습 시간이 적게 걸림
- 이미지 학습 데이터(Pure Water: 1083장, Half: 487, 유출유 : 126장)
- 이미지 학습 데이터(Pure Water: 154장, Half: 237, 적조 : 99장)
- 학습데이터 70%, 검증데이터 30% 설정 후 Confusion Matrix 검증 정확도 99.12%(유출유)
- 학습데이터 70%, 검증데이터 30% 설정 후 Confusion Matrix 검증 정확도 98.98%(적조)
- 유출유 학습 시간 : RTX2060 Single GPU 기준 11분 44초 소요
- 적조 학습 시간 : RTX2060 Single GPU 기준 4분 39초 소요

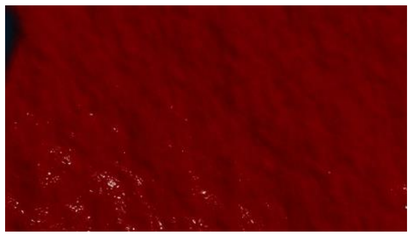
적조



Water : 154

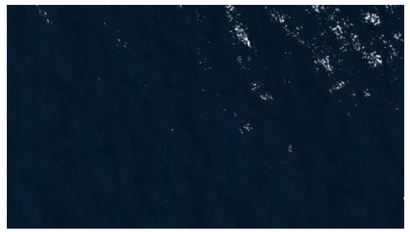


Half : 237



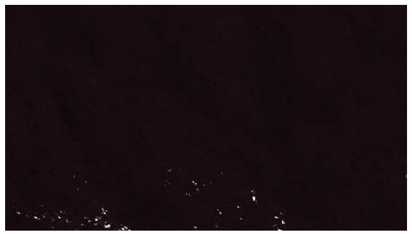
Red Tide : 99

유출유



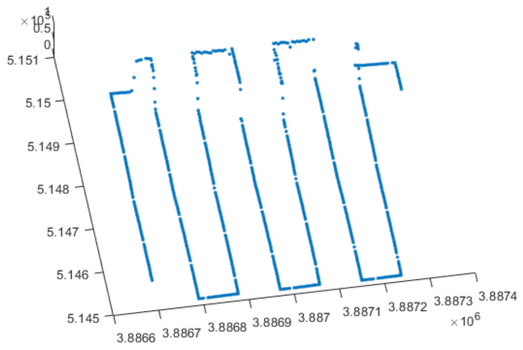
Water : 1083

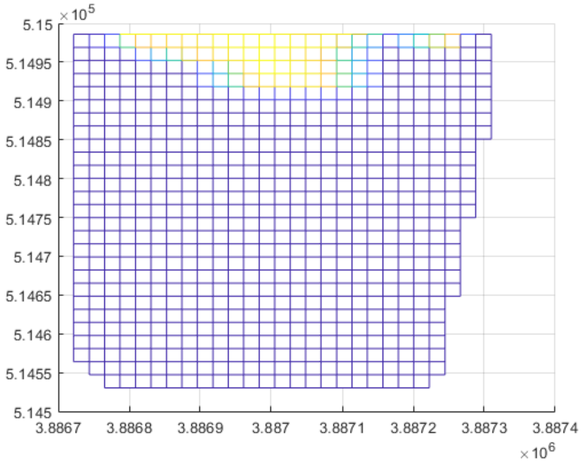
Half : 487

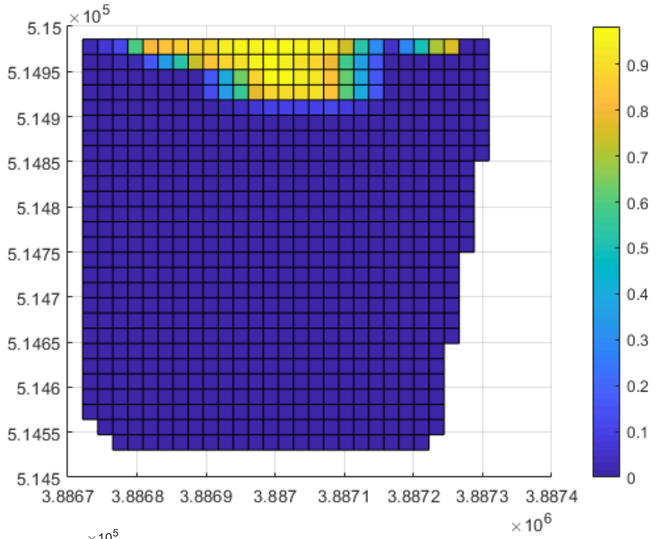


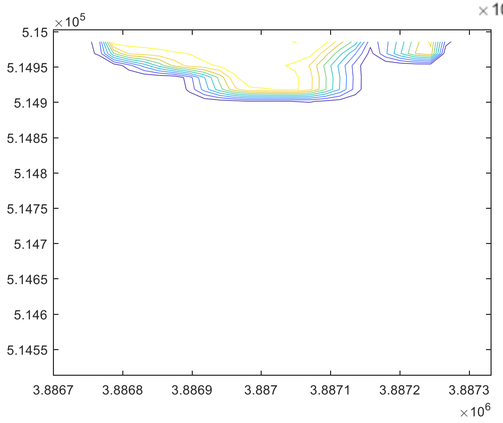
Oil :126

유출유 시뮬레이터 식별 결과

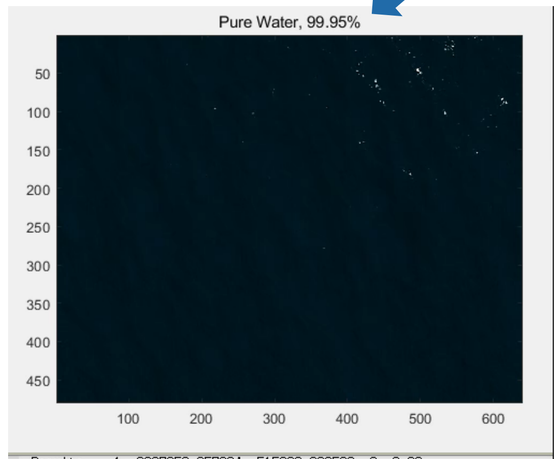








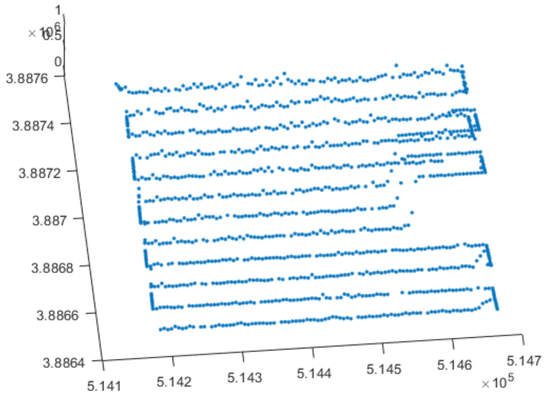
영상식별결과



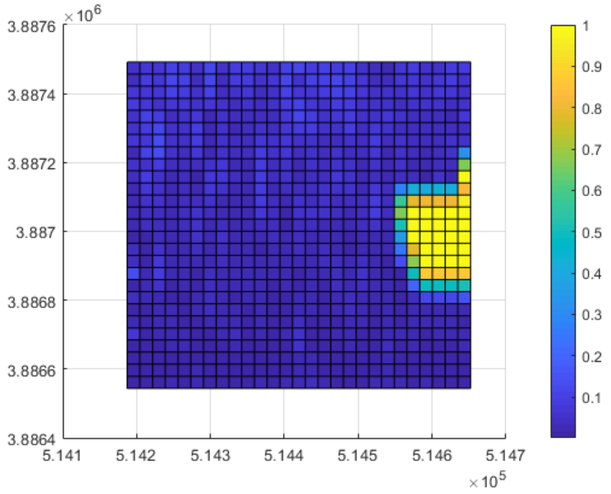

```
Results,   1, 3887258.957224, 515000.302530, 3, 0.00
Results,   1, 3887253.831218, 515000.510703, 3, 0.01
Results,   1, 3887248.705832, 515000.731762, 3, 0.01
Results,   1, 3887243.581519, 515000.973690, 3, 0.02
Results,   1, 3887243.565505, 514995.740071, 3, 0.62
Results,   1, 3887243.548036, 514990.506477, 3, 0.81
Results,   1, 3887243.508603, 514980.039361, 3, 0.93
Results,   1, 3887243.486567, 514974.805842, 3, 0.94
Results,   1, 3887243.462936, 514969.572351, 3, 0.92
Results,   1, 3887243.437670, 514964.338888, 3, 0.40
Results,   1, 3887243.410726, 514959.105456, 3, 0.16
Results,   1, 3887243.382062, 514953.872054, 3, 0.00
Results,   1, 3887243.351631, 514948.638686, 3, 0.00
Results,   1, 3887243.319385, 514943.405351, 3, 0.00
Results,   1, 3887243.285272, 514938.172053, 3, 0.00
Results,   1, 3887243.249237, 514932.938792, 3, 0.00
Results,   1, 3887243.211223, 514927.705570, 3, 0.00
```

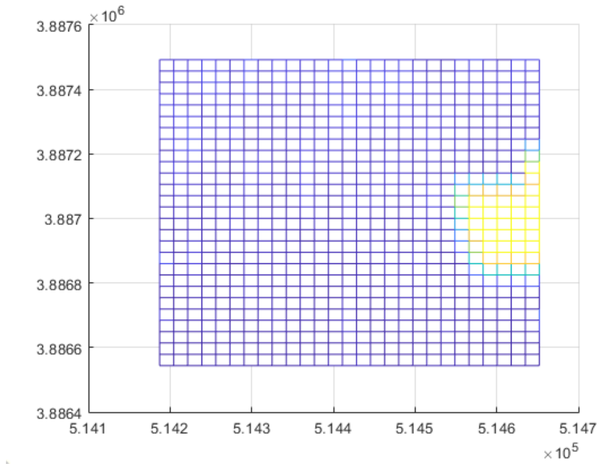
관제시스템과
통신 패킷

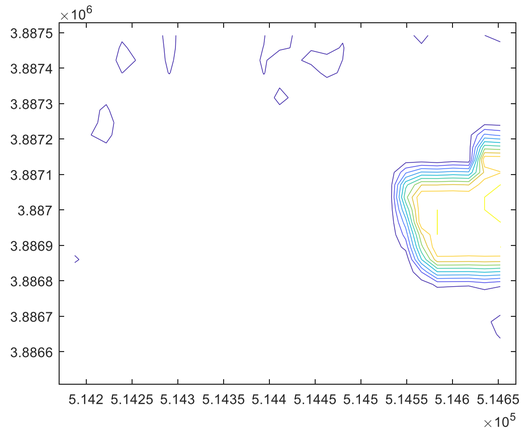
적조 시뮬레이터 식별 결과



무인기 이동경로

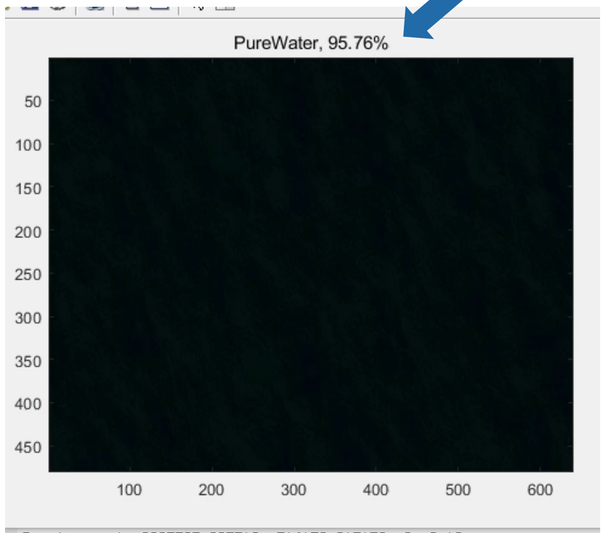






식별된 적조 매핑 결과

영상식별결과




```
Results,   1, 3887527.265510, 514170.017179, 2, 0.12
Results,   1, 3887517.406188, 514172.814452, 2, 0.12
Results,   1, 3887512.478199, 514174.218914, 2, 0.10
Results,   1, 3887507.551829, 514175.628986, 2, 0.09
Results,   1, 3887502.627959, 514177.047671, 2, 0.13
Results,   1, 3887502.623850, 514182.209002, 2, 0.16
Results,   1, 3887502.618318, 514187.370314, 2, 0.06
Results,   1, 3887502.611333, 514192.531605, 2, 0.04
Results,   1, 3887502.602862, 514197.692875, 2, 0.04
Results,   1, 3887502.592871, 514202.854123, 2, 0.05
Results,   1, 3887502.581325, 514208.015347, 2, 0.04
Results,   1, 3887502.568185, 514213.176548, 2, 0.03
Results,   1, 3887502.553414, 514218.337723, 2, 0.10
Results,   1, 3887502.536970, 514223.498872, 2, 0.06
Results,   1, 3887502.518810, 514228.659993, 2, 0.07
Results,   1, 3887502.498888, 514233.821085, 2, 0.04
```

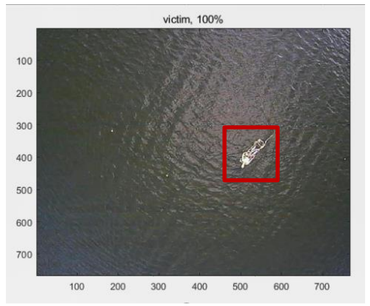
관제시스템과
통신 패킷

Object Detector Using YOLO v2

KRISO

2020년도 연구계획 – AI 기반 조난자 추적모듈

- ❖ 조난자 식별/추적 알고리즘(YOLO)을 통한 조난자 위치 정밀도 향상
- ❖ Edge AI 개발로 드론 탑재 조난자 탐지 알고리즘 개발 (노이즈에 강인)
- ❖ 6차년도 신규 적용 통신 방식 적용 영상 전송 솔루션 구현



조난자 Detector

Edge AI

ICSUV
관제 프로그램

통신

Data : 좌표, 조난자 여부, 확률, 조난자 정밀 좌표

목차

- MATLAB Deep Learning Toolbox를 이용한 Object Detector 알고리즘 구현(YOLO v2)
- 실무에서 실시간 영상 이미지 가져오기
- 통신을 통한 이종 언어, 이종 OS, 이종 시스템 극복
  - Serial 또는 UDP 통신

Object Detector Yolo v2 구현

1 데이터 준비하기

- data = load('victimDatasetGroundTruth.mat');
- victimDataset = data.victimDataset;
- victimDataset(1:4,:)

< 학습 데이터셋의 구조 >

ans = 4×2 table

	Var1	Var2
1	'C:\Users\K...	[79,47,234,...
2	'C:\Users\K...	[72,64,235,...
3	'C:\Users\K...	[59,87,241,...
4	'C:\Users\K...	[48,115,243...

victimDataset

510x2 table

1 Var1	2 Var2
'C:₩Users₩K.H.Kim₩Documents₩MATLAB₩Examples₩R2019b₩deeplearning_shared₩ObjectDetectionUsingYOLOV2DeepLearningExample₩victim_Images₩00001.jpg'	[79,47,234,288]
'C:₩Users₩K.H.Kim₩Documents₩MATLAB₩Examples₩R2019b₩deeplearning_shared₩ObjectDetectionUsingYOLOV2DeepLearningExample₩victim_Images₩00002.jpg'	[72,64,235,286]
'C:₩Users₩K.H.Kim₩Documents₩MATLAB₩Examples₩R2019b₩deeplearning_shared₩ObjectDetectionUsingYOLOV2DeepLearningExample₩victim_Images₩00003.jpg'	[59,87,241,292]
'C:₩Users₩K.H.Kim₩Documents₩MATLAB₩Examples₩R2019b₩deeplearning_shared₩ObjectDetectionUsingYOLOV2DeepLearningExample₩victim_Images₩00004.jpg'	[48,115,243,292]
'C:₩Users₩K.H.Kim₩Documents₩MATLAB₩Examples₩R2019b₩deeplearning_shared₩ObjectDetectionUsingYOLOV2DeepLearningExample₩victim_Images₩00005.jpg'	[46,127,245,294]
'C:₩Users₩K.H.Kim₩Documents₩MATLAB₩Examples₩R2019b₩deeplearning_shared₩ObjectDetectionUsingYOLOV2DeepLearningExample₩victim_Images₩00006.jpg'	[43,155,242,289]
'C:₩Users₩K.H.Kim₩Documents₩MATLAB₩Examples₩R2019b₩deeplearning_shared₩ObjectDetectionUsingYOLOV2DeepLearningExample₩victim_Images₩00007.jpg'	[40,167,241,287]
'C:₩Users₩K.H.Kim₩Documents₩MATLAB₩Examples₩R2019b₩deeplearning_shared₩ObjectDetectionUsingYOLOV2DeepLearningExample₩victim_Images₩00008.jpg'	[38,186,237,283]
'C:₩Users₩K.H.Kim₩Documents₩MATLAB₩Examples₩R2019b₩deeplearning_shared₩ObjectDetectionUsingYOLOV2DeepLearningExample₩victim_Images₩00009.jpg'	[31,199,240,276]
'C:₩Users₩K.H.Kim₩Documents₩MATLAB₩Examples₩R2019b₩deeplearning_shared₩ObjectDetectionUsingYOLOV2DeepLearningExample₩victim_Images₩00010.jpg'	[30,220,239,261]
'C:₩Users₩K.H.Kim₩Documents₩MATLAB₩Examples₩R2019b₩deeplearning_shared₩ObjectDetectionUsingYOLOV2DeepLearningExample₩victim_Images₩00011.jpg'	[35,220,236,260]
'C:₩Users₩K.H.Kim₩Documents₩MATLAB₩Examples₩R2019b₩deeplearning_shared₩ObjectDetectionUsingYOLOV2DeepLearningExample₩victim_Images₩00012.jpg'	[35,240,235,240]
'C:₩Users₩K.H.Kim₩Documents₩MATLAB₩Examples₩R2019b₩deeplearning_shared₩ObjectDetectionUsingYOLOV2DeepLearningExample₩victim_Images₩00013.jpg'	[39,249,234,232]
'C:₩Users₩K.H.Kim₩Documents₩MATLAB₩Examples₩R2019b₩deeplearning_shared₩ObjectDetectionUsingYOLOV2DeepLearningExample₩victim_Images₩00014.jpg'	[43,266,237,215]
'C:₩Users₩K.H.Kim₩Documents₩MATLAB₩Examples₩R2019b₩deeplearning_shared₩ObjectDetectionUsingYOLOV2DeepLearningExample₩victim_Images₩00015.jpg'	[48,273,224,208]
'C:₩Users₩K.H.Kim₩Documents₩MATLAB₩Examples₩R2019b₩deeplearning_shared₩ObjectDetectionUsingYOLOV2DeepLearningExample₩victim_Images₩00016.jpg'	[46,283,229,198]
'C:₩Users₩K.H.Kim₩Documents₩MATLAB₩Examples₩R2019b₩deeplearning_shared₩ObjectDetectionUsingYOLOV2DeepLearningExample₩victim_Images₩00017.jpg'	[56,301,212,180]
'C:₩Users₩K.H.Kim₩Documents₩MATLAB₩Examples₩R2019b₩deeplearning_shared₩ObjectDetectionUsingYOLOV2DeepLearningExample₩victim_Images₩00018.jpg'	[53,308,199,173]

KRISO

Image Labeler로 ROI 표시하기

1 데이터 준비하기 – ROI Label 정의

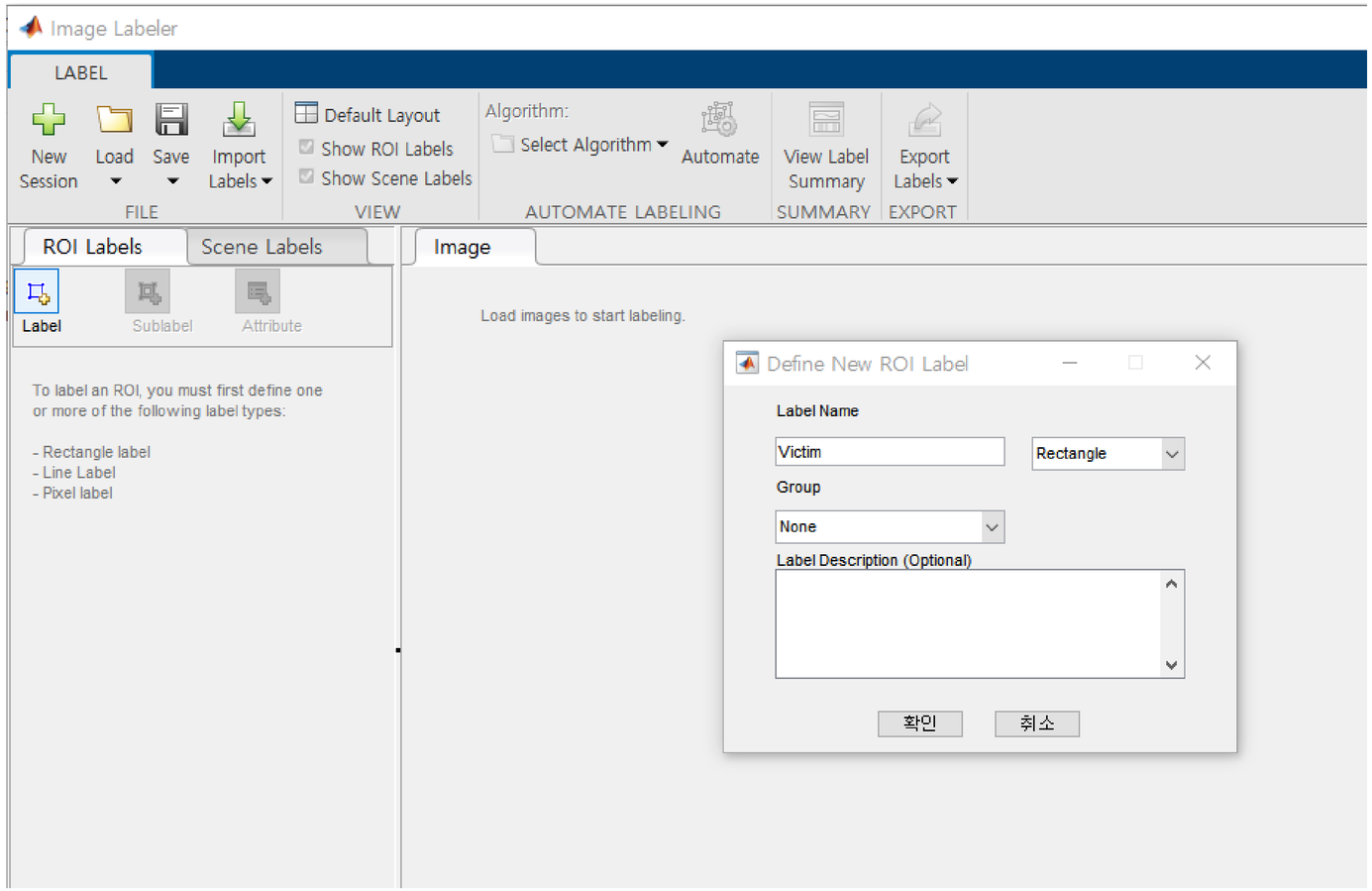


Image Labeler로 ROI 표시하기

- Load -> Add images from folder -> 전체 이미지 선택
- ROI 표시, 오른쪽 화살표 이용 다음 파일 표시
- ROI 표시가 다 끝나면 Export Labels => Workspace로 보내기, Table 선택

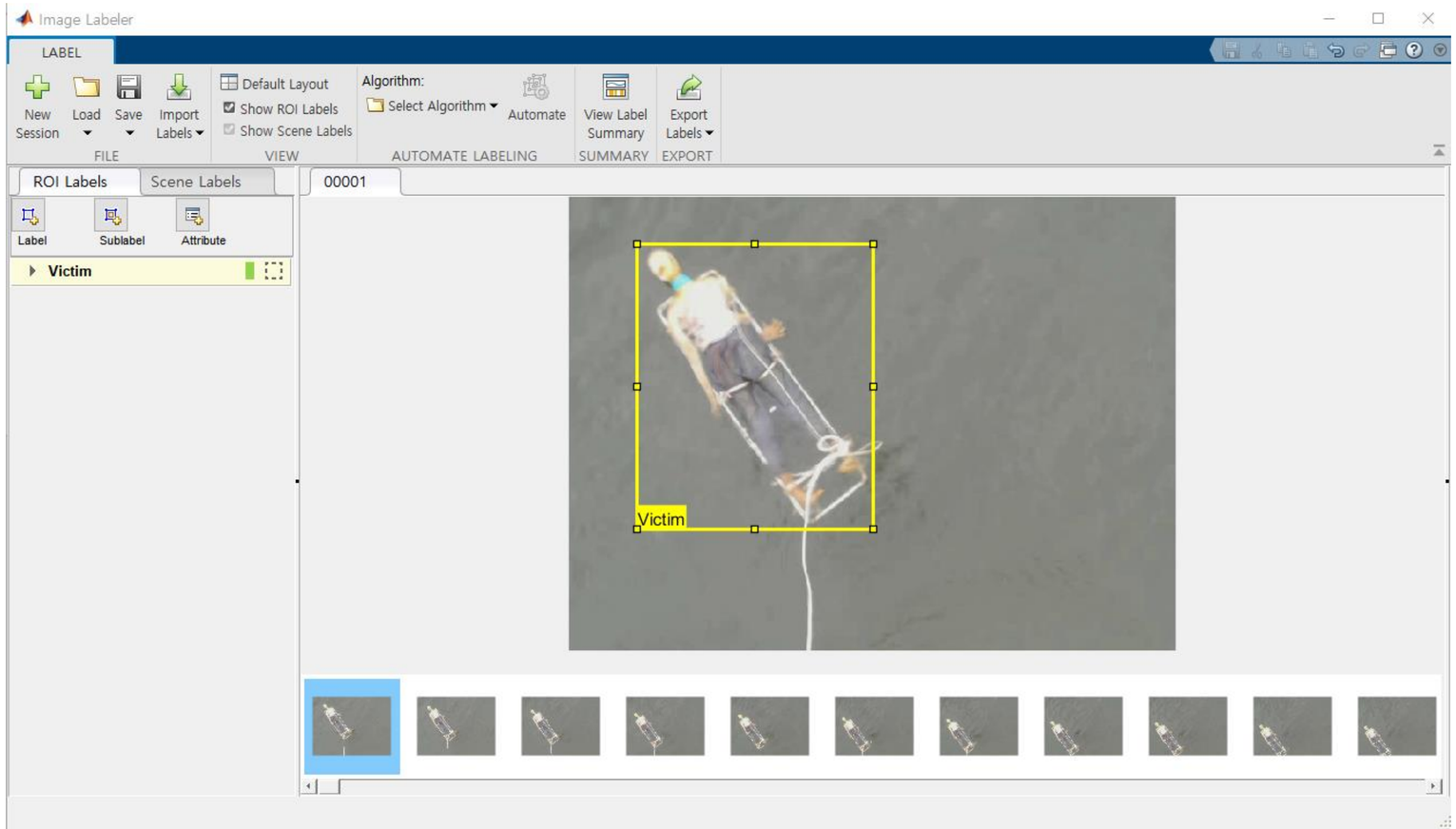


Image Labeler로 ROI 표시하기

작업공간에서 gTruth 더블 클릭

victimDataset.mat으로 저장

gTruth

510x2 table

	1 Var1	2 Var2
1	'C:₩Users₩K.H.Kim₩Documents₩MATLAB₩Examples₩R2019b₩deeplearning_shared₩ObjectDetectionUsingYOLOV2DeepLearningExample₩victim_Images₩00001.jpg'	[79,47,234,288]
2	'C:₩Users₩K.H.Kim₩Documents₩MATLAB₩Examples₩R2019b₩deeplearning_shared₩ObjectDetectionUsingYOLOV2DeepLearningExample₩victim_Images₩00002.jpg'	[72,64,235,286]
3	'C:₩Users₩K.H.Kim₩Documents₩MATLAB₩Examples₩R2019b₩deeplearning_shared₩ObjectDetectionUsingYOLOV2DeepLearningExample₩victim_Images₩00003.jpg'	[59,87,241,292]
4	'C:₩Users₩K.H.Kim₩Documents₩MATLAB₩Examples₩R2019b₩deeplearning_shared₩ObjectDetectionUsingYOLOV2DeepLearningExample₩victim_Images₩00004.jpg'	[48,115,243,292]
5	'C:₩Users₩K.H.Kim₩Documents₩MATLAB₩Examples₩R2019b₩deeplearning_shared₩ObjectDetectionUsingYOLOV2DeepLearningExample₩victim_Images₩00005.jpg'	[46,127,245,294]
6	'C:₩Users₩K.H.Kim₩Documents₩MATLAB₩Examples₩R2019b₩deeplearning_shared₩ObjectDetectionUsingYOLOV2DeepLearningExample₩victim_Images₩00006.jpg'	[43,155,242,289]
7	'C:₩Users₩K.H.Kim₩Documents₩MATLAB₩Examples₩R2019b₩deeplearning_shared₩ObjectDetectionUsingYOLOV2DeepLearningExample₩victim_Images₩00007.jpg'	[40,167,241,287]
8	'C:₩Users₩K.H.Kim₩Documents₩MATLAB₩Examples₩R2019b₩deeplearning_shared₩ObjectDetectionUsingYOLOV2DeepLearningExample₩victim_Images₩00008.jpg'	[38,186,237,283]
9	'C:₩Users₩K.H.Kim₩Documents₩MATLAB₩Examples₩R2019b₩deeplearning_shared₩ObjectDetectionUsingYOLOV2DeepLearningExample₩victim_Images₩00009.jpg'	[31,199,240,276]
10	'C:₩Users₩K.H.Kim₩Documents₩MATLAB₩Examples₩R2019b₩deeplearning_shared₩ObjectDetectionUsingYOLOV2DeepLearningExample₩victim_Images₩00010.jpg'	[30,220,239,261]
11	'C:₩Users₩K.H.Kim₩Documents₩MATLAB₩Examples₩R2019b₩deeplearning_shared₩ObjectDetectionUsingYOLOV2DeepLearningExample₩victim_Images₩00011.jpg'	[35,220,236,260]
12	'C:₩Users₩K.H.Kim₩Documents₩MATLAB₩Examples₩R2019b₩deeplearning_shared₩ObjectDetectionUsingYOLOV2DeepLearningExample₩victim_Images₩00012.jpg'	[35,240,235,240]
13	'C:₩Users₩K.H.Kim₩Documents₩MATLAB₩Examples₩R2019b₩deeplearning_shared₩ObjectDetectionUsingYOLOV2DeepLearningExample₩victim_Images₩00013.jpg'	[39,249,234,232]
14	'C:₩Users₩K.H.Kim₩Documents₩MATLAB₩Examples₩R2019b₩deeplearning_shared₩ObjectDetectionUsingYOLOV2DeepLearningExample₩victim_Images₩00014.jpg'	[43,266,237,215]
15	'C:₩Users₩K.H.Kim₩Documents₩MATLAB₩Examples₩R2019b₩deeplearning_shared₩ObjectDetectionUsingYOLOV2DeepLearningExample₩victim_Images₩00015.jpg'	[48,273,224,208]

학습 데이터 준비완료!!!

KRISO

I9 Core, RTX2080 Ti
Batch Size 24, Epoch 20
8분 소요

```
*************************************************************************
Training a YOLO v2 Object Detector for the following object classes:

* Var2

단일 GPU에서 훈련합니다.
입력 데이터의 정규화를 초기화하는 중입니다.
|=========================================================================================|
|   Epoch   |  반복 횟수  |      경과 시간      |  미니 배치 RMSE  |  미니 배치 손실  |  기본 학습률  |
|           |             |    (hh:mm:ss)       |                  |                  |               |
|=========================================================================================|
|         1 |           1 |  00:00:03           |          9.12    |         83.2     |    0.0010     |
|         4 |          50 |  00:01:29           |          3.17    |         10.0     |    0.0010     |
|         8 |         100 |  00:02:59           |          3.97    |         15.8     |    0.0010     |
|        11 |         150 |  00:04:25           |          4.05    |         16.4     |    0.0010     |
|        15 |         200 |  00:05:54           |          4.05    |         16.4     |    0.0010     |
|        18 |         250 |  00:07:18           |          0.59    |          0.3     |    0.0010     |
|        20 |         280 |  00:08:09           |          0.54    |          0.3     |    0.0010     |
|=========================================================================================|
Detector training complete.
*************************************************************************
```

삼성 Flex 노트북 GPU MX250
Batch Size 12, Epoch 20
21분 소요

```
*************************************************************************
Training a YOLO v2 Object Detector for the following object classes:

* Var2

단일 GPU에서 훈련합니다.
입력 데이터의 정규화를 초기화하는 중입니다.
|=========================================================================================|
|   Epoch   |  반복 횟수  |      경과 시간      |  미니 배치 RMSE  |  미니 배치 손실  |  기본 학습률  |
|           |             |    (hh:mm:ss)       |                  |                  |               |
|=========================================================================================|
|         1 |           1 |  00:00:02           |          9.21    |         84.8     |    0.0010     |
|         2 |          50 |  00:01:09           |          1.14    |          1.3     |    0.0010     |
|         3 |         100 |  00:02:20           |          1.17    |          1.4     |    0.0010     |
|         4 |         150 |  00:03:31           |          1.12    |          1.3     |    0.0010     |
|         6 |         200 |  00:04:46           |          0.75    |          0.6     |    0.0010     |
|         7 |         250 |  00:05:57           |          0.70    |          0.5     |    0.0010     |
|         8 |         300 |  00:07:09           |          0.60    |          0.4     |    0.0010     |
|        10 |         350 |  00:08:23           |          0.74    |          0.6     |    0.0010     |
|        11 |         400 |  00:09:35           |          0.48    |          0.2     |    0.0010     |
|        12 |         450 |  00:10:50           |          0.66    |          0.4     |    0.0010     |
|        14 |         500 |  00:12:32           |          0.62    |          0.4     |    0.0010     |
|        15 |         550 |  00:14:15           |          0.42    |          0.2     |    0.0010     |
|        16 |         600 |  00:15:56           |          0.45    |          0.2     |    0.0010     |
|        18 |         650 |  00:17:17           |          0.37    |          0.1     |    0.0010     |
|        19 |         700 |  00:18:57           |          0.57    |          0.3     |    0.0010     |
|        20 |         750 |  00:20:38           |          0.42    |          0.2     |    0.0010     |
|        20 |         760 |  00:20:56           |          0.42    |          0.2     |    0.0010     |
|=========================================================================================|
Detector training complete.
*************************************************************************
```

I9 Core, RTX2080 Ti
Batch Size 32, Epoch 30
9분 15초 소요

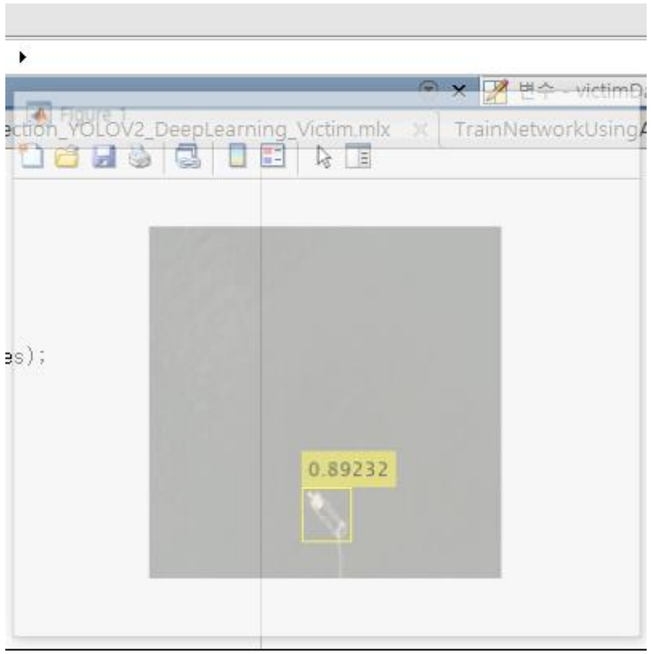
```
*************************************************************************
Training a YOLO v2 Object Detector for the following object classes:

* Var2

단일 GPU에서 훈련합니다.
입력 데이터의 정규화를 초기화하는 중입니다.
|=========================================================================================|
|   Epoch   |  반복 횟수  |      경과 시간      |  미니 배치 RMSE  |  미니 배치 손실  |  기본 학습률  |
|           |             |    (hh:mm:ss)       |                  |                  |               |
|=========================================================================================|
|         1 |           1 |  00:00:05           |          7.37    |         54.3     |    0.0010     |
|         8 |          50 |  00:02:16           |          1.59    |          2.5     |    0.0010     |
|        15 |         100 |  00:04:28           |          0.74    |          0.5     |    0.0010     |
|        22 |         150 |  00:06:41           |          0.56    |          0.3     |    0.0010     |
|        29 |         200 |  00:08:52           |          0.28    |       7.9e-02    |    0.0010     |
|        30 |         210 |  00:09:15           |          0.34    |          0.1     |    0.0010     |
|=========================================================================================|
Detector training complete.
*************************************************************************
```

I9 Core, RTX2080 Ti
Batch Size 32, Epoch 50
15분 28초 소요

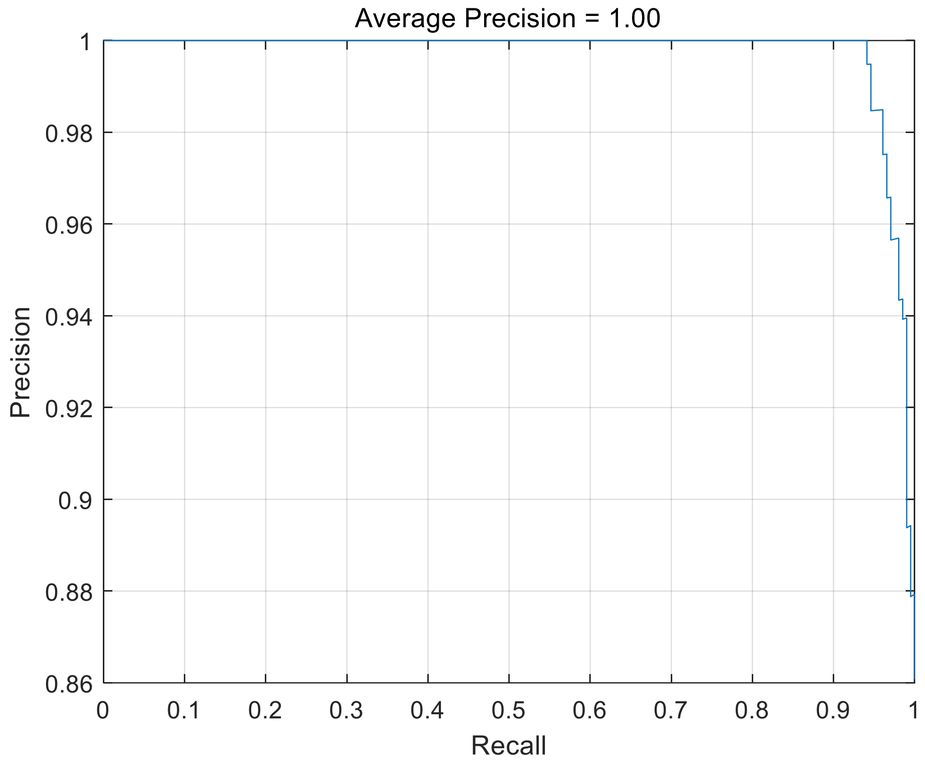
```
*************************************************************************
Training a YOLO v2 Object Detector for the following object classes:

* Var2

단일 GPU에서 훈련합니다.
입력 데이터의 정규화를 초기화하는 중입니다.
|=========================================================================================|
|   Epoch   |  반복 횟수  |      경과 시간      |  미니 배치 RMSE  |  미니 배치 손실  |  기본 학습률  |
|           |             |    (hh:mm:ss)       |                  |                  |               |
|=========================================================================================|
|         1 |           1 |  00:00:05           |          7.37    |         54.3     |    0.0010     |
|         8 |          50 |  00:02:17           |          1.57    |          2.5     |    0.0010     |
|        15 |         100 |  00:04:29           |          0.74    |          0.5     |    0.0010     |
|        22 |         150 |  00:06:41           |          0.58    |          0.3     |    0.0010     |
|        29 |         200 |  00:08:53           |          0.29    |       8.1e-02    |    0.0010     |
|        36 |         250 |  00:11:05           |          0.34    |          0.1     |    0.0010     |
|        43 |         300 |  00:13:16           |          0.20    |       3.9e-02    |    0.0010     |
|        50 |         350 |  00:15:28           |          0.20    |       4.0e-02    |    0.0010     |
|=========================================================================================|
Detector training complete.
*************************************************************************
```

KRISO

Object Detection 성능



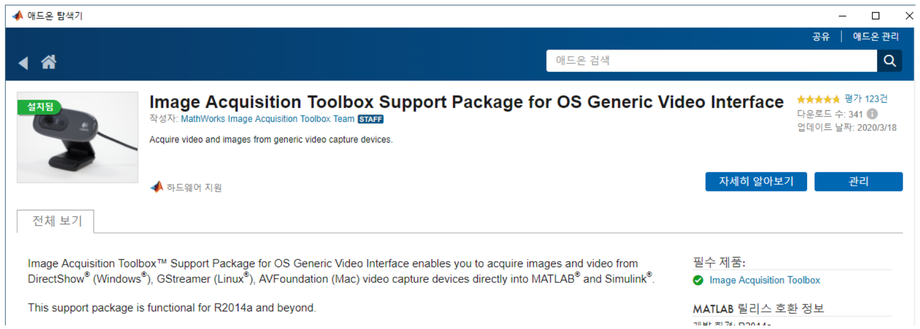
Precision vs. Recall



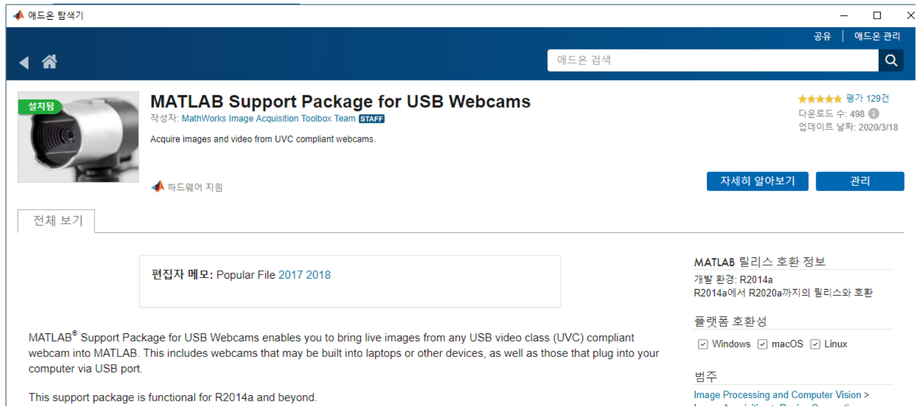
ap = 0.9974

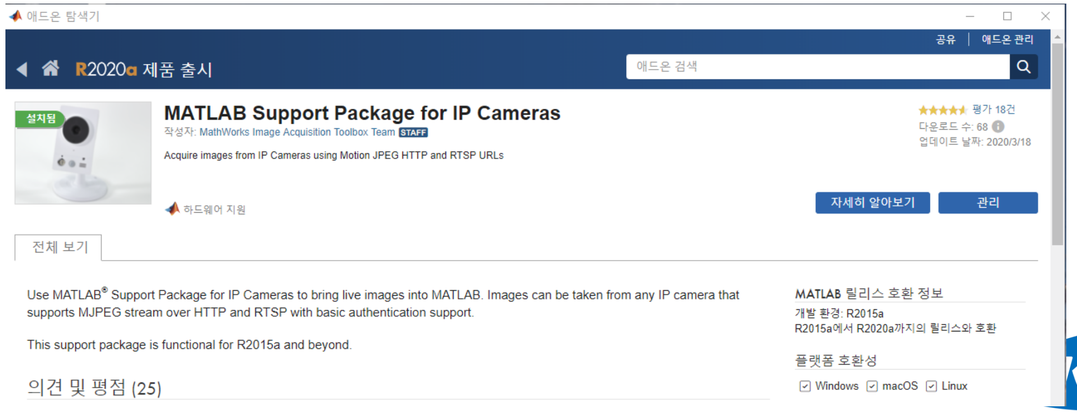
실제 프로젝트에서 실시간 영상 연동 준비

- USB Webcam (실시간)
- HDMI2USB 캡쳐 보드 (실시간)
- IP Camera - Mjpeg, RTSP (약간의 Delay)



- 애드온 탐색기에서 Webcam Support Package 찾아서 설치
- HDMI영상 HDMI2USB 캡쳐보드 사용 Webcam으로 인식 가능
- 모든 PC화면 HDMI로 처리 가능
  - 각종 SDK 없는 응용프로그램, 시뮬레이터 이미지 기반 인공지능 데이터 처리 가능





KRISO

Webcam/HDMI에서 영상 받아 분석/통신 예제


```
cam = webcam(1);
cam.resolution = '640x480';

preview(cam); %하드웨어 문제없는지 이때 확인

load yolov2ResNet50Victim.mat;  %기학습된 detector 로딩, YOLO 약 100Mb

inputSize = [448 448 3]; %원래는 224x224로 학습
```

```
h1 = figure(1);

while ishandle(h1)
% AI모듈 호출
AI_UAV1

% 통신 모듈 호출
STR_UDP_UAV1 % UDP 통신 수신, 파싱, 결과 전송 모듈
end
```

IP Camera에서 영상 받아 분석/통신 예제

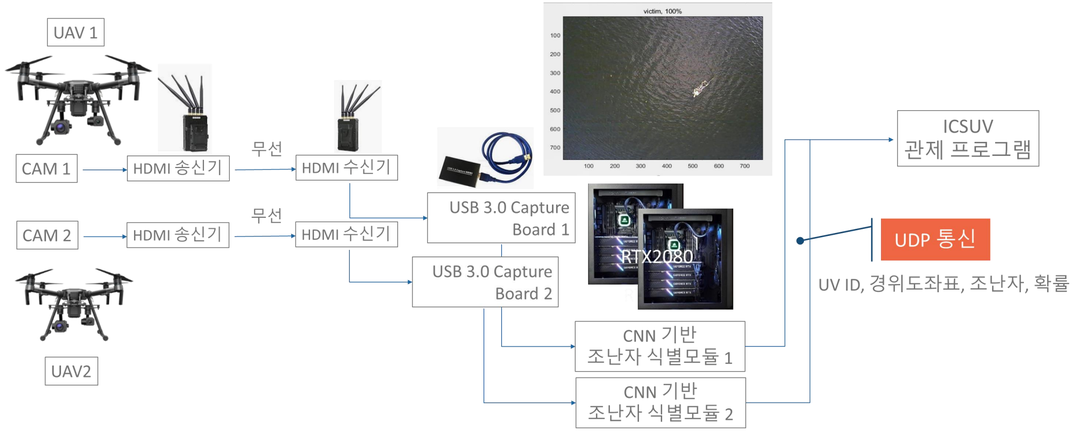

```
cam = ipcam('http://172.28.17.193/video.mjpeg', 'admin', 'myfoscam')

cam.resolution = '640x480';

preview(cam); %하드웨어 문제없는지 이때 확인

load yolov2ResNet50Victim.mat;  %기학습된 detector 로딩, YOLO 약 100Mb

inputSize = [448 448 3];
```

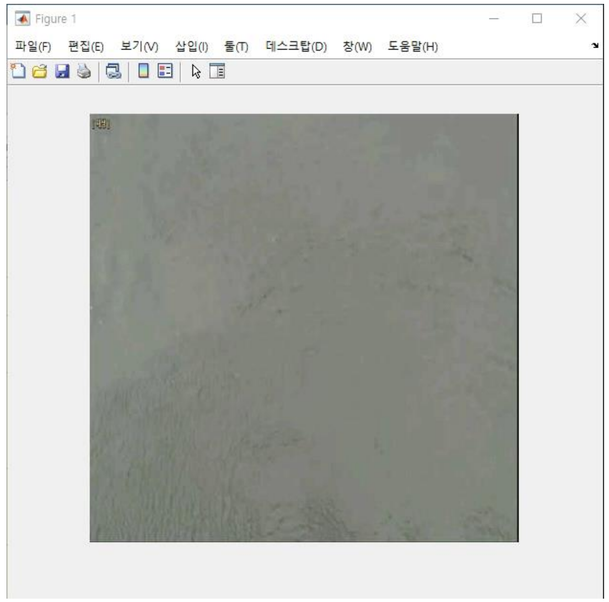
```
h1 = figure(1);

while ishandle(h1)
% AI모듈 호출
AI_UAV1

% 통신 모듈 호출
STR_UDP_UAV1 % UDP 통신 수신, 파싱, 결과 전송 모듈
end
```

AI UAV1 모듈

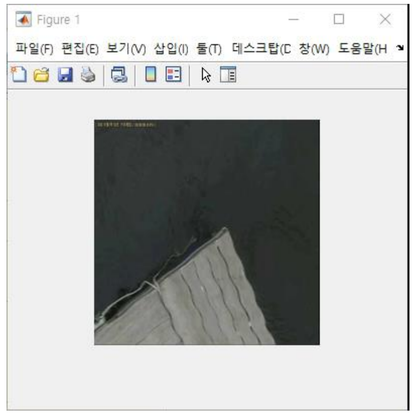
```
im = snapshot(cam);
%im = getsnapshot(vid1);  %Image Acquisition Toolbox 제공모듈

I = imresize(im,inputSize(1:2));
imshow(I)

[bboxes,scores] = detect(detector,I);

if (scores > 0.5)      %프로그램 종료 방지루틴
I = insertObjectAnnotation(I,'rectangle',bboxes,scores);
imshow(I)
end
```

Object Detection 성능



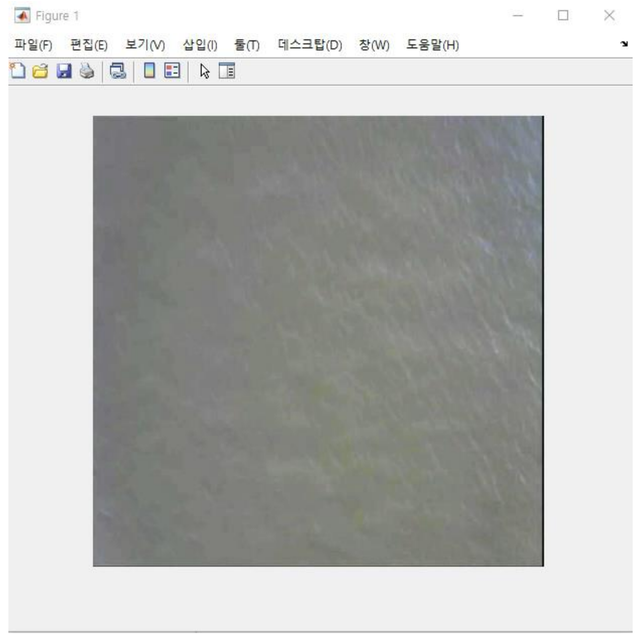


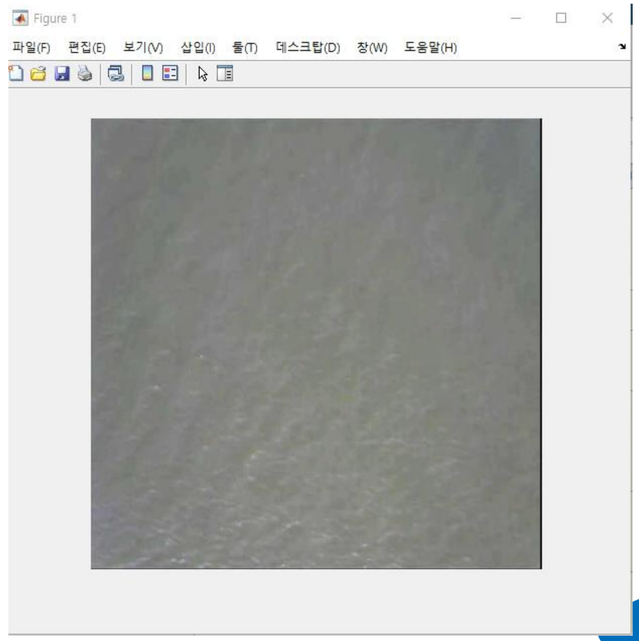
448x448



224x224

< 평택호 시험 영상 >





통신으로 다른 OS, 다른 언어, 다른 HW와 통합하기

- Communication Toolbox나 ROS Toolbox 이용
- UDP 통신 / 시리얼 통신

STR_UDP_UAV1 구동을 위한 통신, 프로토콜 포맷 세팅

```
% UDP 주소, 리모트포트, 로컬포트, 리모트 오픈, 터미네이터 CR

s1 = udp('192.168.0.6','RemotePort',4023,'LocalPort',12050, 'Terminator','CR');

% Input, Output 버퍼 사이즈 정의
set(s1, 'InputBufferSize', 30000);
set(s1, 'OutputBufferSize', 30000);

% 포트오픈
fopen(s1);
disp(s1.status);

% 전송 패킷 정의
formatSpec = 'Results, %3d, %7.6f, %6.6f, 1, %2.2f₩r₩0';
```

STR_UDP_UAV1 통신 모듈

```
%% UDP Data Receive
Data = []; % Flush 초기화

while (get(s1, 'BytesAvailable') > 0)
s1.BytesAvailable;
Data = fscanf(s1);
end

%% Data Parsing and Send
txt1 = string(Data);

while isempty(txt1)
    fprintf('No packet from ICSUV₩r')
    return
end

data_split = strsplit(txt1,',');
ID = str2double(data_split(1,2));
lon = str2double(data_split(1,3));
lat = str2double(data_split(1,4));

score1_n = double(score1(2));
A1 = [ID; lon; lat; score1_n];
fprintf(s1, formatSpec, A1);
fprintf(formatSpec, A1)
```

KRISO

딥러닝을 위한 하드웨어 추천 사양

- 서버 RTX2080 Ti 2ea, I9 Core : 800만원, Yolo 학습 8분(GPU 1개만 사용)
- 데스크탑 RTX2080 Ti, I7 Core : 400만원(추천), Yolo 학습 8분
- RTX2070 Super 탑재 데스크탑 : 200만원
- 썬더볼트(40Gbps) 랩탑 + eGPU(RTX2080 Ti) : 70% 정도 성능
- LG RTX2060 노트북 : 220만원, 휴대성
- 삼성 플렉스 노트북 MX250 : 250만원, Yolo 학습 20분, 휴대성
- Nvidia Jetson Nano 보드 : 20만원 (MATLAB Coder, GPU Coder 필요)

eGPU

LG울트라북

삼성플렉스

Edge-AI
나노보드

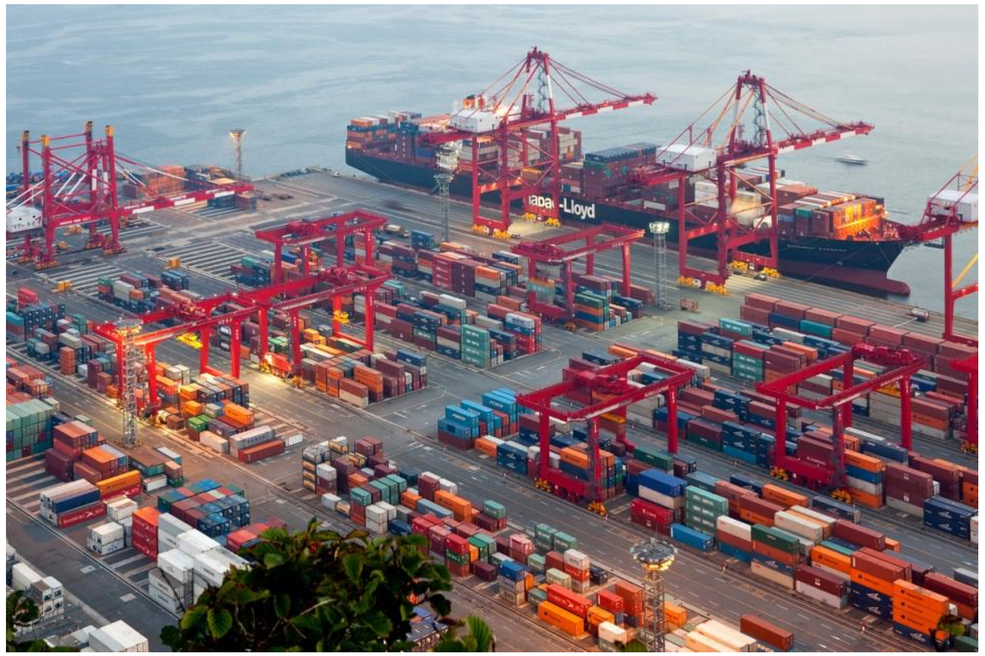
KRISO

현재 수행 중 연구 소개
- 스마트 항만IOT – 항만 자동화 연구

터미널 게이트

- 외부트럭 차량번호인식
- 컨테이너 번호인식
- 컨테이너 도어방향 인식
- 컨테이너 손상도 자동 분석

- 침입
- 배회
- 유기
- 쓰러짐
- 화재



갠트리 크레인 하부

- 야드트럭 번호인식
- 컨테이너 번호인식
- 대기 차량 계수
- 차량 / 사람 추적 사고위험 경고

현재 수행 중 연구 소개
- 소나 영상 딥러닝 연구

- **Configuration**

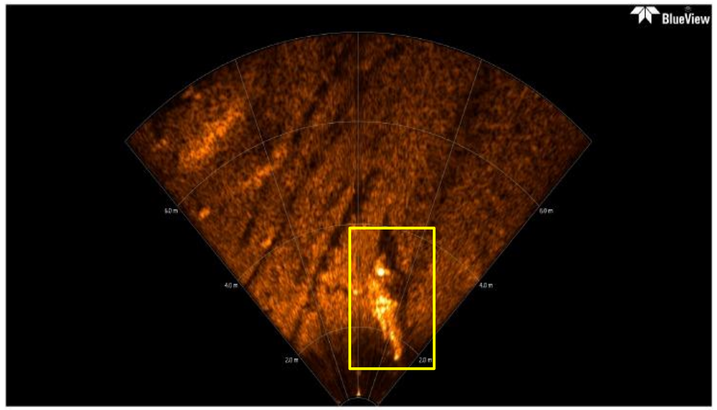
Imaging sonar
exclusive driver

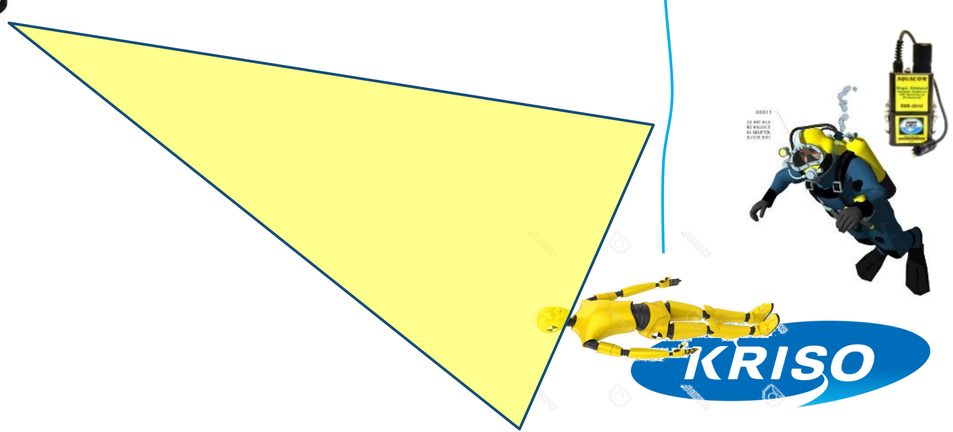
TCP

POE box

Tcp/poe

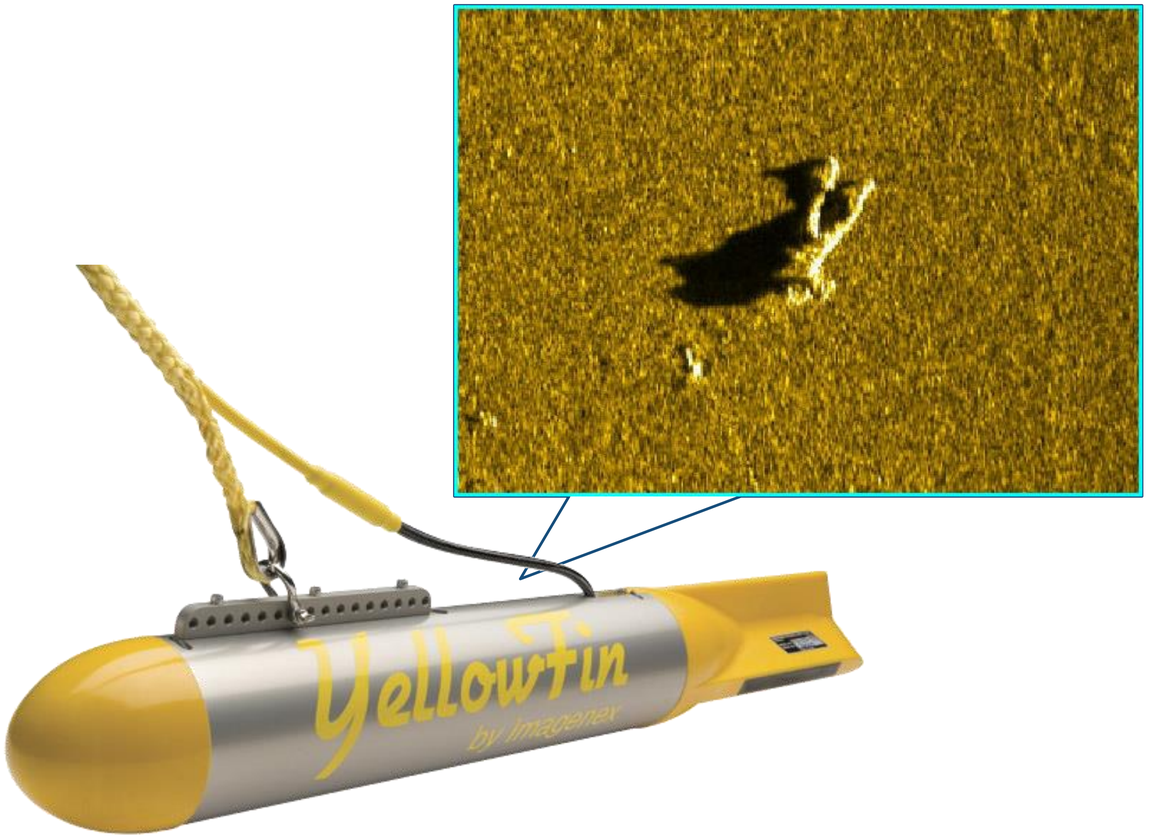
STX-101 surface station



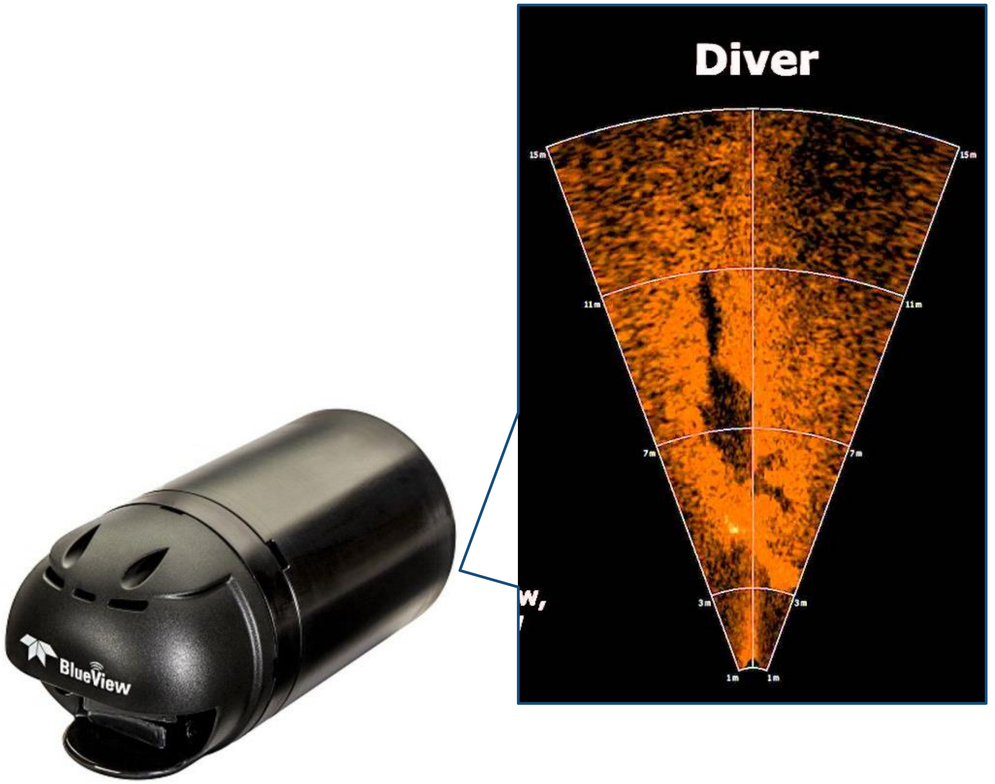


현재 수행 중 연구 소개
- 소나 영상 딥러닝 연구

Side scan sonar



Imaging sonar



경청 해주셔서 감사합니다

편안하게 질문 또는 코멘트 주시기 바랍니다

궁금한 점이 있으신 분이나
협업을 원하시는 분은
아래 연락처로 연락바랍니다

선박해양플랜트연구소
김기훈 책임연구원
042-866-3814
khkim@kriso.re.kr