

# Real World Problem Project Using OpenCV



**AVG**enius



**AVG**enius

## ❖ 프로젝트 개요

- 주제 : 악천후 환경에서 Camera Fail-Safe를 위한 Camera 영상의 필터링 방법
- 악천후 환경에서 자율주행차 주행 중 객체 인식(Object Detection)에 주로 활용하는 Camera 센서의 오염이나 급격한 환경 변화로 인해 객체 인식에 오류가 발생할 가능성이 있는 경우 해당 영상의 이미지에 필터링(filtering)을 적용하여 해결
- 악천후 환경 유형



야간(night)



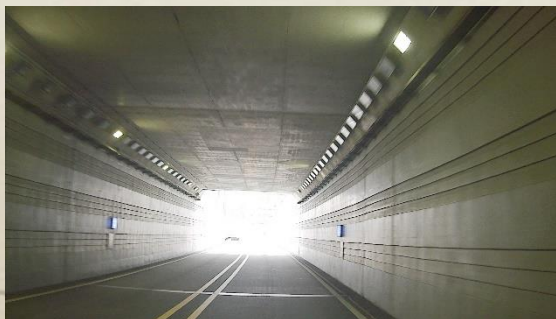
강우(rain)



안개(fog)



역광(backlight)



급격한 조도 변화(sudden change in illumination)



## ❖ 프로젝트 진행

### ➤ 악천후 환경 주행 데이터 취득

- AI-Hub(<https://www.aihub.or.kr/>) 공공 데이터셋의 딥러닝용 악천후 환경 자율주행 데이터셋 활용(2023년 상용 자율주행차 악천후 데이터)
- 도심로의 야간, 안개, 강우, 역광, 급격한 조도 변화 환경에 해당하는 데이터 선별

AI 데이터찾기

AI 허브소개

참여하기

커뮤니티

AI 개발지원

고객지원

마이페이지

로그아웃

사업공고

마이페이지

관심데이터 조회

신청 및 문의 조회

교육 영상 시청 내역

회원정보관리

데이터 신청내역

다운로드 목록

신규데이터 제안내역

AI 컴퓨팅 지원 이용 신청내역

1:1 문의내역

개인정보신고 내역

클라우드워커문의 내역

24년 초거대AI 확산 생태계 사업 문의 내역

데이터 신청 내역

데이터 다운로드 신청 절차

1.접수완료

2.승인완료

3.자동승인

※ 안심존 데이터의 경우 안심존 사용신청 상태가 제공됩니다

번호	분야	데이터명	구분	파일유형	상태	다운로드
1	교통물류	상용 자율주행차 악천후 데이터		3D,이미지	자동승인	다운로드

### AI 학습용 다운로드

- 를 클릭하시면 하위 폴더와 파일 목록을 확인할 수 있습니다.
- 전체 파일을 한번에 다운로드 받고자 할 경우는 [전체 다운로드] 를, 일부만 선택하여 다운로드 받고자 하실 경우는 다운로드 받을 파일을 선택하신 뒤, [선택 다운로드] 버튼을 눌러주세요.
- **주의 사항**  
파일 이어받기가 안되는 경우 제어판의 INNORIX EX Agent를 삭제 후 재설치 하시길 바랍니다.  
기존 설치된 Agent를 삭제하시면 install 페이지로 이동합니다.  
[수동설치](#)

☐ NAME ▲
 

SIZE ▲

☐ Validation
 

☐ 02.라벨링데이터
 ☐ 01.원천데이터

☐ Other
 

☐ Other.zip

☐ Training
 

☐ fileURI
 ☐ 라벨링데이터
 ☐ 01.원천데이터

171MB

선택 다운로드

전체 다운로드



## ❖ 프로젝트 진행

## ➤ 악천후 환경 별 필터링 적용

- 야간 환경의 저조도(low illumination) 및 노이즈(noise)에 대해 명암도와 노이즈 개선이 필요하며 이를 위해 Gaussian Blur와 Sharpening filtering을 통해 노이즈 감소 및 이미지 선명도를 높이는 작업 수행
- 안개 환경의 대비(contrast) 감소와 색상 변화로 인해 대비 향상 및 color balance를 조정하여 이미지 선명도를 높이는 작업 수행(다양한 Filtering 적용)
- 강우 환경의 빗물로 인한 Camera blur 처리는 second-order derivatives의 Laplacian filter를 적용하여 경계선(edge) 추출을 통해 경계선의 선명도를 분석하여 강우 상황을 판단할 수 있도록 진행
- 역광 환경의 대비(contrast) 감소와 음영(shadow) 발생으로 인해 대비 향상 및 음영 필터링을 위한 히스토그램 평활화(Histogram Equalization) 및 CLAHE Filtering을 통해 이미지 선명도를 높이는 작업 수행
- 급격한 조도 변화 환경은 히스토그램 평활화(Histogram Equalization) 및 CLAHE Filtering을 통해 이미지 선명도를 높이는 작업 수행

## ❖ 프로젝트 진행

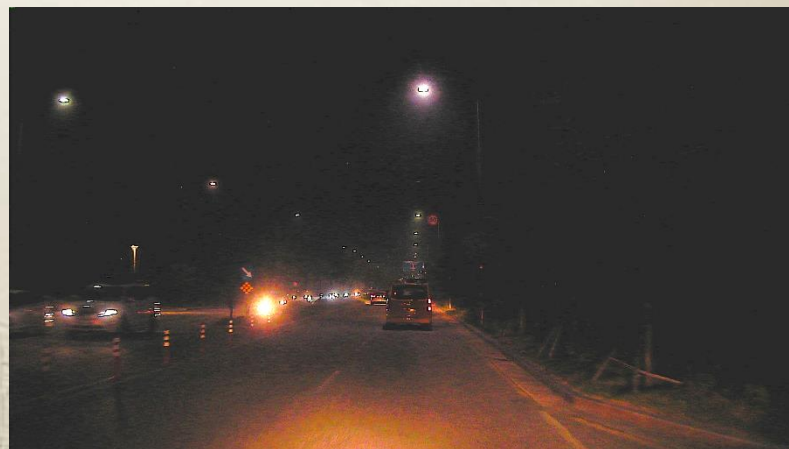
## ➤ 야간 환경 이미지 필터링

- Gaussian Blur 및 Sharpening Filtering을 통해 노이즈 감소 및 이미지 선명도 개선
- bad\_weather\_night.py

```
bad_weather_image_filtering.py  bad_weather_night.py ×  
1  import cv2  
2  import numpy as np  
3  
4  # image set-up(1920*1080)  
5  image = cv2.imread('/home/jinhakim/anaconda3/envs/opencv_lec/Midterm_Project/bad_weather_images/night/16_211922_220929_40.jpg')  
6  
7  # Applying Gaussian Blur to image  
8  gaussian_blur = cv2.GaussianBlur(image, ksize: (7,7), sigmaX: 0)  
9  
10 # Applying Sharpening filter  
11 sharpening = cv2.addWeighted(image, alpha: 2.8, gaussian_blur, -0.6, gamma: 0)  
12  
13 # Visualization of applying filtering  
14 cv2.imshow( winname: 'Filtering Applied Image', sharpening)  
15 cv2.waitKey()  
16 cv2.destroyAllWindows()
```

## ❖ 프로젝트 진행

### ➤ 야간 환경 이미지 필터링(명암 및 노이즈 개선)



원본 이미지

필터링 적용 이미지

## ❖ 프로젝트 진행

## ➤ 강우 환경 이미지 필터링

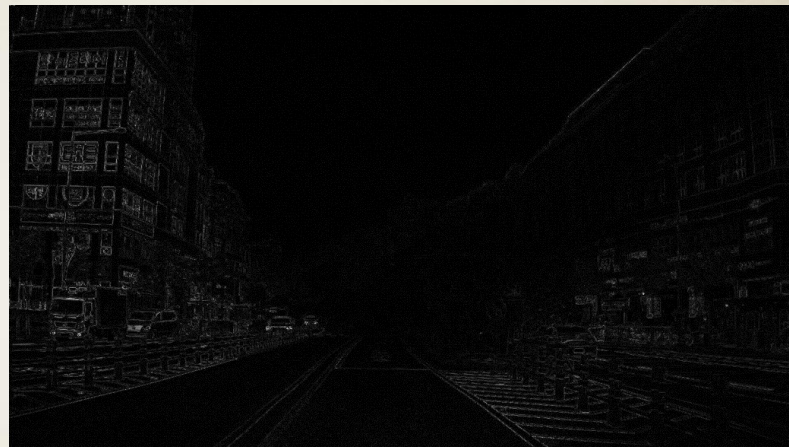
- Laplacian Filtering을 통해 빗물로 인한 Blur 제거
- bad\_weather\_rain.py

```
bad_weather_image_filtering.py bad_weather_night.py bad_weather_fog.py bad_weather_rain.py ×  
1 import cv2  
2 import numpy as np  
3  
4 # image set-up  
5 image = cv2.imread('/home/jinhakim/anaconda3/envs/opencv_lec/Midterm_Project/bad_weather_images/rain/24_111049_220829_01.jpg')  
6  
7 # Converting image color(BGR) to grayscale  
8 gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)  
9  
10 # Applying Laplacian filter to bad weather image(blurred)  
11 laplacian_filtering = cv2.Laplacian(gray_image, cv2.CV_64F)  
12  
13 # Applying absolute value and converting type  
14 laplacian_filtering = np.uint8(np.absolute(laplacian_filtering))  
15  
16 # Visualization of applying Laplacian filter  
17 cv2.imshow( winname: 'Laplacian Filter Applied Image', laplacian_filtering)  
18 cv2.waitKey()  
19 cv2.destroyAllWindows()
```



## ❖ 프로젝트 진행

## ➤ 강우 환경 이미지 필터링(빗방울로 인한 Blur 제거)



원본 이미지

Laplacian Filtering



## ❖ 프로젝트 진행

## ➤ 안개 환경 이미지 필터링-01

- Gaussian Blur, Median Blur, Bilateral Filter를 통해 이미지 선명도 개선
- bad\_weather\_fog.py

```
bad_weather_image_filtering.py  bad_weather_night.py  bad_weather_fog.py ×
1  import cv2
2  import matplotlib.pyplot as plt
3
4  # image set-up(1920*1080)
5  image = cv2.imread('/home/jinhakim/anaconda3/envs/opencv lec/Midterm_Project/bad_weather_images/fog/16_072312_221021_08.jpg')
6
7  # Gaussian Blur
8  gaussian_blur = cv2.GaussianBlur(image, ksize=(5, 5), sigmaX=0)
9
10 # Median Blur
11 median_blur = cv2.medianBlur(image, ksize=5)
12
13 # Bilateral Filter
14 bilateral_filter = cv2.bilateralFilter(image, d=9, sigmaColor=75, sigmaSpace=75)
15
16 # Visualization of Applying filtering
17 titles = ['original image', 'gaussian image', 'median image', 'bilateral image']
18 images = [image, gaussian_blur, median_blur, bilateral_filter]
19
20 for i in range(4):
21     plt.subplot(*args=(2, 2, i+1))
22     plt.imshow(images[i], cmap='gray')
23     plt.title(titles[i])
24     plt.xticks([], plt.yticks([]))
25
26 plt.tight_layout()
27 plt.show()
```

## ❖ 프로젝트 진행

- 안개 환경 이미지 필터링-01(각 필터링 별 육안상 뚜렷한 차이는 없음.)



원본 이미지



Gaussian Blur



Median Blur



Bilateral Filter

## ❖ 프로젝트 진행

## ➤ 안개 환경 이미지 필터링-02

- Laplacian Filtering을 통해 안개(Blur) 제거
- bad\_weather\_fog\_v2.py

```
bad_weather_fog_v2.py × bad_weather_rain.py bad_weather_backlight_v2.py bad_weather_tunnel_v2.py bad_weather_tunnel_v
1 import cv2
2 import numpy as np
3
4 # image set-up
5 image = cv2.imread('/home/jinhakim/anaconda3/envs/opencv_lec/Midterm_Project/bad_weather_images/fog/16_072312_221021_08.jpg')
6
7 # Converting image color(BGR) to grayscale
8 gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
9
10 # Applying Laplacian filter to bad weather image(blurred)
11 laplacian_filtering = cv2.Laplacian(gray_image, cv2.CV_64F)
12
13 # Applying absolute value and converting type
14 laplacian_filtering = np.uint8(np.absolute(laplacian_filtering))
15
16 # Visualization of applying Laplacian filter
17 cv2.imshow( winname: 'Laplacian Filter Applied Image', laplacian_filtering)
18 cv2.waitKey()
19 cv2.destroyAllWindows()
```



## ❖ 프로젝트 진행

## ➤ 안개 환경 이미지 필터링-02(Laplacian Filtering 적용 시)



원본 이미지



Laplacian Filtering

- 안개를 Blur로 간주하면 Laplacian Filtering 적용이 Blur 제거에 효과가 있는 것으로 판단됨.



## ❖ 프로젝트 진행

## ➤ 안개 환경 이미지 필터링-03

- Canny Edge Filtering을 통해 안개(Blur) 제거
- bad\_weather\_fog\_v3.py

```
bad_weather_rain.py  bad_weather_backlight_v2.py  bad_weather_tunnel_v2.py  bad_weather_tunnel_v1.py  bad_weather_fog_v3.py
1  import cv2
2
3  # image set-up(1920*1080)
4  image = cv2.imread('/home/jinhakim/anaconda3/envs/opencv lec/Midterm_Project/bad_weather_images/fog/16_072312_221021_08.jpg')
5
6  # Converting image color(BGR) to grayscale
7  gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
8
9  # Applying Canny Edge Filtering
10 edges = cv2.Canny(gray_image, threshold1=30, threshold2=60)
11
12 # Visualization of applying filtering
13 cv2.imshow( winname: 'Canny Edge Filtering', edges)
14 cv2.waitKey()
15 cv2.destroyAllWindows()
```

## ❖ 프로젝트 진행

## ➤ 안개 환경 이미지 필터링-03(Canny Edge Filtering 적용 시)



원본 이미지



Canny Edge Filtering

- 안개를 Blur로 간주하면 Canny Edge Filtering 적용이 Blur 제거에 효과가 있는 것으로 판단됨.
- 다만, 결과 이미지의 노이즈를 제거하기 위한 방법도 고민할 필요가 있음.





## ❖ 프로젝트 진행

## ➤ 역광 환경 이미지 필터링-01

- 히스토그램 평활화(Histogram Equalization)를 통해 이미지 전체 밝기를 조절하여 명암 차이를 감소시켜 주변 배경 및 객체 선명도 향상
- bad\_weather\_backlight\_v1.py

```
bad_weather_image_filtering.py  bad_weather_backlight_v1.py  bad_weather_night.py  bad_weather_fog.py  bad_weather_rain.py
1  import cv2
2
3  # image set-up(1920*1080)
4  image = cv2.imread('/home/jinhakim/anaconda3/envs/opencv_lec/Midterm_Project/bad_weather_images/reverse_light/16_080557_221019_01.jpg')
5
6  # Converting image color(BGR) to grayscale
7  gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
8
9  # Applying histogram equalization
10 equalized_image = cv2.equalizeHist(gray_image)
11
12 # Converting equalized_image to BGR type
13 equalized_image_bgr = cv2.cvtColor(equalized_image, cv2.COLOR_GRAY2BGR)
14
15 # Visualization of applying filtering
16 cv2.imshow( winname: 'Backlight Corrected Image', equalized_image_bgr)
17
18 cv2.waitKey(0)
19 cv2.destroyAllWindows()
```

## ❖ 프로젝트 진행

### ➤ 역광 환경 이미지 필터링-01(대비 향상 및 음영 감소)



원본 이미지

Histogram Equalization

## ❖ 프로젝트 진행

## ➤ 역광 환경 이미지 필터링-02

- CLAHE(Contrast Limited Adaptive Histogram Equalization)를 통해 이미지의 contrast 및 주변 객체의 선명도 향상(수업에서 다루지 않은 기법으로 관련 code는 google에서 찾아 적용)
- bad\_weather\_backlight\_v2.py

```
bad_weather_night.py bad_weather_fog.py bad_weather_rain.py bad_weather_backlight_v1.py bad_weather_backlight_v2.py x v ...
1 import cv2
2
3 # image set-up(1920*1080)
4 image = cv2.imread('/home/jinhakim/anaconda3/envs/opencv_lec/Midterm_Project/bad_weather_images/reverse_light/16_082602_221022_35.jpg')
5
6 # Reducing shadow
7 clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
8 clahe_image = cv2.cvtColor(image, cv2.COLOR_BGR2LAB)
9 clahe_image[:, :, 0] = clahe.apply(clahe_image[:, :, 0])
10 shadow_reduced = cv2.cvtColor(clahe_image, cv2.COLOR_LAB2BGR)
11
12 # Enhancing contrast
13 lab = cv2.cvtColor(shadow_reduced, cv2.COLOR_BGR2LAB)
14 l, a, b = cv2.split(lab)
15 clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
16 l = clahe.apply(l)
17 lab = cv2.merge((l, a, b))
18 contrast_enhanced = cv2.cvtColor(lab, cv2.COLOR_LAB2BGR)
19
20 # Visualization of applying filtering
21 cv2.imshow( winname: 'CLAHE Filter Applied Image', contrast_enhanced)
22 cv2.waitKey(0)
23 cv2.destroyAllWindows()
```



## ❖ 프로젝트 진행

### ➤ 역광 환경 이미지 필터링-02(대비 향상 및 음영 감소)



원본 이미지

CLAHE Filtering

## ❖ 프로젝트 진행

## ➤ 급격한 조도 변화 환경 이미지 필터링-01

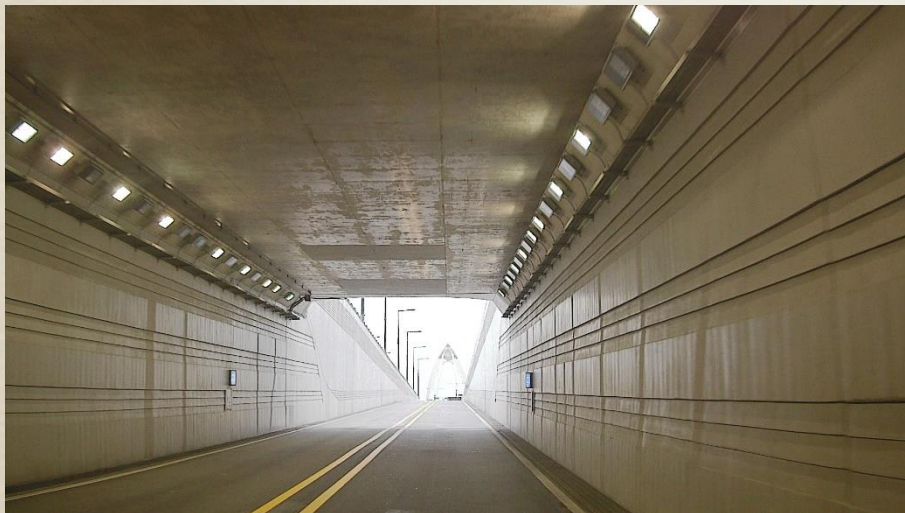
- 히스토그램 평활화(Histogram Equalization)를 통해 이미지 전체 밝기를 조절하여 명암 차이를 감소시켜 주변 배경 및 객체 선명도 향상
- bad\_weather\_tunnel\_v1.py

```
bad_weather_fog.py bad_weather_rain.py bad_weather_backlight_v2.py bad_weather_tunnel_v2.py bad_weather_tunnel_v1.py
1 import cv2
2
3 # image set-up(1920*1080)
4 image = cv2.imread('/home/jinhakim/anaconda3/envs/opencv_lec/Midterm_Project/bad_weather_images/tunnel/24_111849_220829_38.jpg')
5
6 # Converting image color(BGR) to grayscale
7 gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
8
9 # Applying histogram equalization
10 equalized_image = cv2.equalizeHist(gray_image)
11
12 # Converting equalized_image to BGR type
13 equalized_image_bgr = cv2.cvtColor(equalized_image, cv2.COLOR_GRAY2BGR)
14
15 # Visualization of applying filtering
16 cv2.imshow( winname: 'Corrected Image', equalized_image_bgr)
17
18 cv2.waitKey(0)
19 cv2.destroyAllWindows()
```



## ❖ 프로젝트 진행

### ➤ 급격한 조도 변화 환경 이미지 필터링-01



원본 이미지



Histogram Equalization

- Histogram Equalization을 적용했을 때 오히려 원본 이미지보다 터널 안/밖 밝기 조절이 떨어지는 결과가 나왔음.





## ❖ 프로젝트 진행

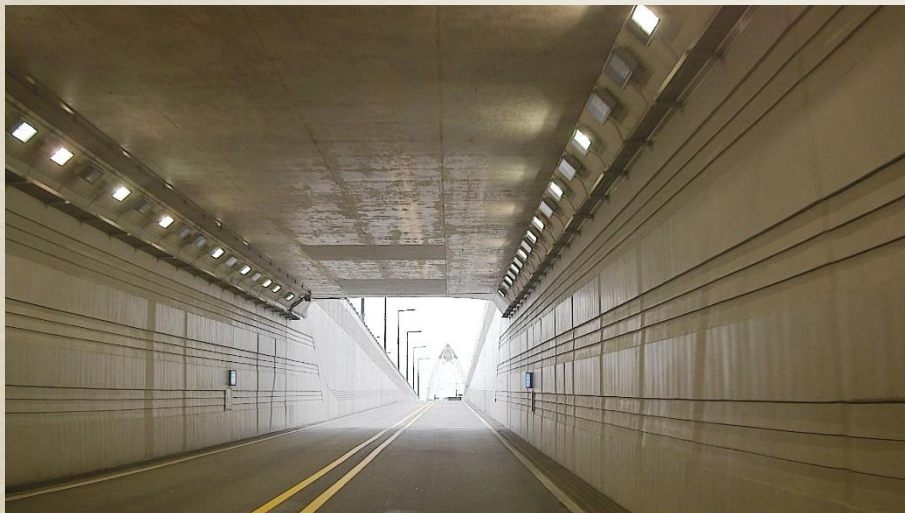
## ➤ 급격한 조도 변화 환경 이미지 필터링-02

- CLAHE(Contrast Limited Adaptive Histogram Equalization)를 통해 이미지의 contrast 및 주변 객체의 선명도 향상(수업에서 다루지 않은 기법으로 관련 code는 google에서 찾아 적용)
- bad\_weather\_tunnel\_v2.py

```
bad_weather_fog.py bad_weather_rain.py bad_weather_backlight_v2.py bad_weather_tunnel_v2.py x bad_weather_tunnel_v1.py
1 import cv2
2
3 # image set-up(1920*1080)
4 image = cv2.imread('/home/jinhakim/anaconda3/envs/opencv_lec/Midterm_Project/bad_weather_images/tunnel/24_111849_220829_38.jpg')
5
6 # Reducing shadow
7 clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
8 clahe_image = cv2.cvtColor(image, cv2.COLOR_BGR2LAB)
9 clahe_image[:, :, 0] = clahe.apply(clahe_image[:, :, 0])
10 shadow_reduced = cv2.cvtColor(clahe_image, cv2.COLOR_LAB2BGR)
11
12 # Enhancing contrast
13 lab = cv2.cvtColor(shadow_reduced, cv2.COLOR_BGR2LAB)
14 l, a, b = cv2.split(lab)
15 clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
16 l = clahe.apply(l)
17 lab = cv2.merge((l, a, b))
18 contrast_enhanced = cv2.cvtColor(lab, cv2.COLOR_LAB2BGR)
19
20 # Visualization of applying filtering
21 cv2.imshow( winname: 'CLAHE Filter Applied Image', contrast_enhanced)
22 cv2.waitKey(0)
23 cv2.destroyAllWindows()
```

## ❖ 프로젝트 진행

### ➤ 급격한 조도 변화 환경 이미지 필터링-01



원본 이미지



CLAHE Filtering

- CLAHE Filtering을 적용했을 때 원본 이미지보다 터널 안/밖 밝기 조절이 향상되는 결과가 나왔음.



### ❖ 프로젝트 결과 고찰

- 야간, 강우, 역광, 급격한 조도 변화 환경에서의 이미지는 대체로 적합한 filtering을 통해 이미지 개선이 되는 것을 확인할 수 있었음.
- 안개의 경우 여러 가지 filtering 기법을 적용하였고 Gaussian, Median, Bilateral Filtering은 큰 차이는 없었고, 안개를 Blur라고 가정했을 때 Blur를 제거한 Laplacian 이나 Canny Edge Filtering이 적절할 것으로 판단되나 더 적합한 Filtering 기법에 대한 추후 연구가 필요함.





# THANK YOU



**AVG**enius



**AVG**enius