컴퓨터 비전 기말발표

이미지 및 비디오에서 농작물(잎)병변 탐지를 위한 다양한 컴퓨터 비전 알고리즘의 적용

이 준 혁 (2023250419)

산 업 인 공 지 능 학 과

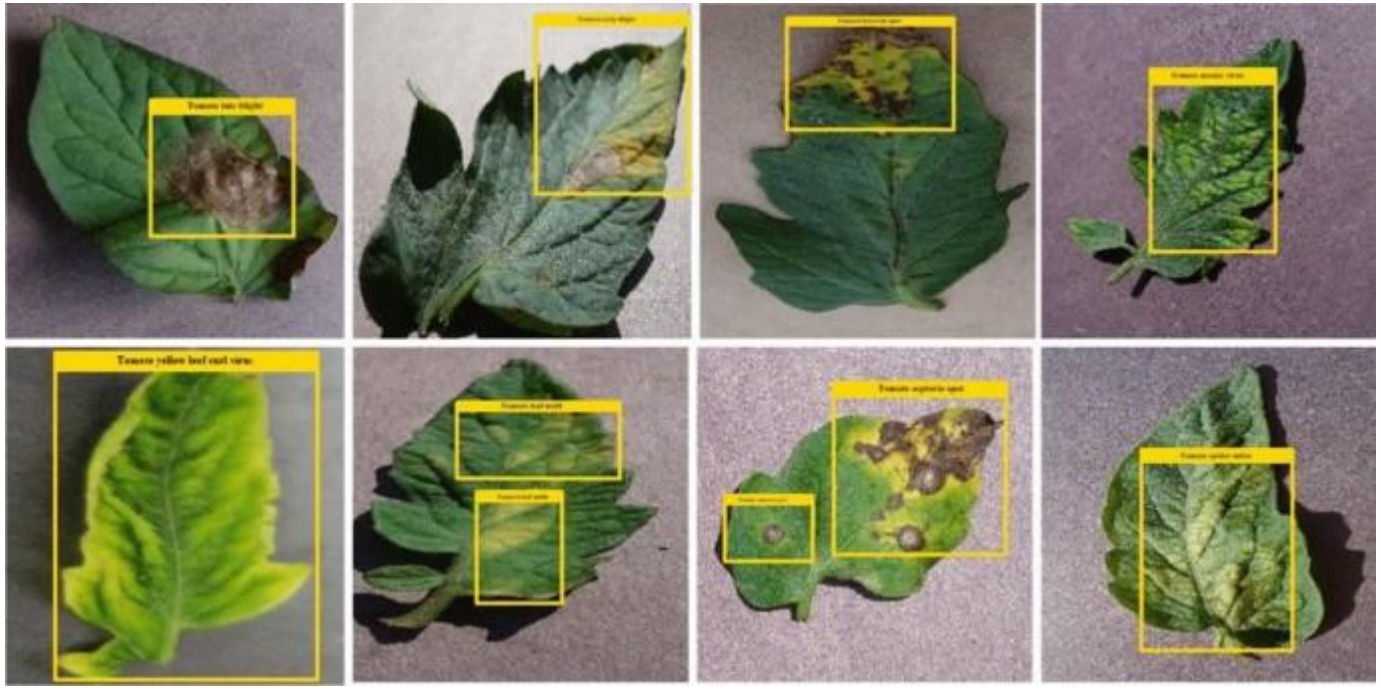
Dept. of Industrial AI

주제 선정

본인의 직/간접적 직무 분야 중 하나인 임베디드/IoT/스마트팜 분야로 선정
해당 주제 중 영상정보를 이용한 작물의 병변 확인을 주제로 선정

활용처(필요성)

본인이 수행하는 프로젝트 중 하나인 스마트팜에 설치된 카메라를 사용하여 영상정보를 입력하면 농작물의 잎에 갈변현상이 발생한 것을 확인할 수 있도록 하기 위함.



사용한 알고리즘/필터링

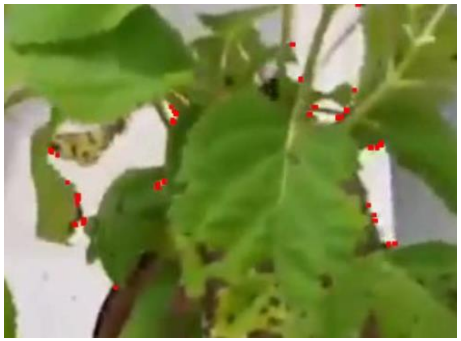
나열한 알고리즘이 시각화 하여 병변이 있는 잎사귀에 적용하기 좋을 것이라고 판단함.

HSV 기반 색상 탐지

설명: 이미지를 HSV 색상 공간으로 변환하여 특정 색상 범위를 탐지

적용: 초록색과 갈색 영역을 탐지하여 병변의 존재 여부를 파악

도움: 병변이 특정 색상으로 나타나는 경우, 이를 정확히 탐지하여 병변의 위치를 시각적으로 표시 가능



Harris 코너 검출

설명: 이미지에서 코너(모서리) 점을 탐지하는 알고리즘

적용: 병변의 경계나 특징적인 모서리 점을 탐지

도움: 병변의 경계가 명확하게 드러나는 경우, 모서리 점을 통해 병변의 형태를 더 정확히 파악 가능

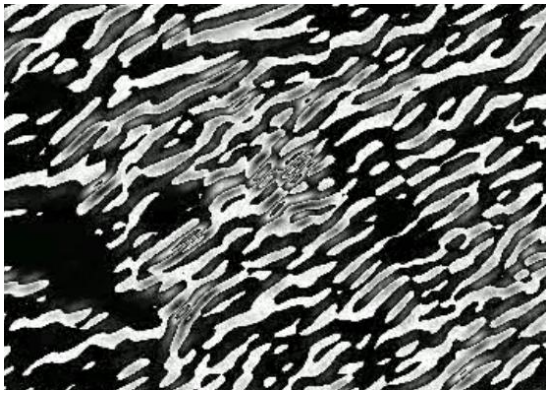
사용한 알고리즘/필터링

FAST 특징점 검출

설명: 이미지에서 특징점(interest points)을 빠르게 탐지하는 알고리즘

적용: 병변의 특징점들을 탐지

도움: 병변의 고유한 특징을 파악하여 이를 기반으로 병변을 구분하고 분석 가능



가보 필터링

설명: 특정 방향과 주파수를 갖는 텍스처를 강조하는 필터

적용: 병변의 텍스처와 패턴을 강조하여 탐지

도움: 병변의 표면 텍스처가 일반 영역과 다를 경우, 이를 강조하여 병변을 더 명확히 탐지 가능

HSV **기반 병변 탐지**

- BGR 색상공간 -> HSV색상공간
- 초록과 갈색 범위 지정
- 픽셀 마스크 생성
- 초록, 갈색 영역 결합
- 2가지 색상 이외 검정 지정
- 병변부 박스 그리기
- 질병 판단 기준 80%로 산정

```python
def process_frame(frame, display=False):
    # 이미지를 HSV로 변환
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    # 초록색 범위 정의 (HSV 색상 범위)
    lower_green = np.array([40, 40, 40])
    upper_green = np.array([70, 255, 255])

    # 갈색 범위 정의 (HSV 색상 범위)
    lower_brown = np.array([0, 50, 20])
    upper_brown = np.array([20, 230, 160])

    # 초록색 영역과 갈색 영역을 감지하여 마스크 생성
    green_mask = cv2.inRange(hsv, lower_green, upper_green)
    brown_mask = cv2.inRange(hsv, lower_brown, upper_brown)

    # 초록색 영역과 갈색 영역 결합
    result_mask = cv2.bitwise_or(green_mask, brown_mask)

    # 초록색과 갈색 이외의 부분을 검정색으로 설정
    result_image = cv2.bitwise_and(frame, frame, mask=result_mask)

    # 갈색 부분에 붉은색 상자 그리기
    contours, _ = cv2.findContours(brown_mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    disease_pixels = 0
    total_pixels = np.count_nonzero(result_mask)
    for contour in contours:
        x, y, w, h = cv2.boundingRect(contour)
        area = w * h
        disease_pixels += area
        cv2.rectangle(result_image, (x, y), (x + w, y + h), (0, 0, 255), 2)

    # 질병으로 판정하는 비율 계산
    disease_ratio = (disease_pixels / total_pixels) * 100

    # 질병 비율이 80% 이상일 때만 박싱 처리된 프레임을 반환
    if disease_ratio >= 80:
        if display:
            cv2_imshow(result_image)
        return result_image
    else:
```

Harris 코너 : 모서리/코너 탐지 알고리즘 - 병변의 경계/모서리 탐지위함

```python
def apply_harris_corner(frame, display=False):
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    corners = cv2.cornerHarris(gray, 2, 3, 0.04)
    corners = cv2.dilate(corners, None)
    frame[corners > 0.1 * corners.max()] = [0, 0, 255]

    if display:
        cv2_imshow(frame)

    return frame
```

- BGR 색상공간 -> **그레이스케일 변환**

- **매개변수설정**
  -gray: 그레이스케일 이미지
  -2: 코너 검출에 사용할 블록의 크기 (블록 사이즈-노이즈 관련됨)
  -3: 소벨(Sobel) 커널의 크기 (그래디언트 계산)
  -0.04: Harris 검출기 방정식에서 사용하는 상수(0.04~0.06)

- 검출된 코너 점을 **팽창**시켜 강조

FAST **특징점 검출** : 이미지에서 특징점을 빠르게 탐지하는 방법

-병변의 독특한 특징점을 탐지하고 시각적으로 표시함으로써, 병변 탐지의 정확성 높임

```python
def apply_fast_feature(frame, display=False):
    fast = cv2.FastFeatureDetector_create(30, True, cv2.FAST_FEATURE_DETECTOR_TYPE_9_16)
    kp = fast.detect(frame)
    for point in kp:
        x, y = np.int0(point.pt)
        cv2.circle(frame, (x, y), 2, (0, 255, 0), cv2.FILLED)

    if display:
        cv2_imshow(frame)

    return frame
```

- **매개변수설정**

-**30: 임계값(threshold)**으로, 픽셀 강도가 주변 픽셀과 얼마나 달라야 특징점으로 간주되는지 결정

-**True: 비최대 억제(non-maximum suppression)**를 사용할지 여부. 비최대 억제는 주변 픽셀보다 강도가 높은 특징점만 남기고 제거

-**cv2.FAST_FEATURE_DETECTOR_TYPE_9_16: FAST** 알고리즘의 타입을 설정. 9-16은 16개의 원형 패턴 중 9개 이상의 픽셀이 중심 픽셀과 큰 차이를 가지는 경우 특징점으로 간주

- 특징점 검출
- 검출된 특징점 시각화

가보 필터링: 이미지의 특정 주파수와 방향을 강조, 특정 패턴을 강조

```python
def apply_gabor_filter(frame, display=False):
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY).astype(np.float32) / 255
    kernel = cv2.getGaborKernel((21, 21), 5, 1, 10, 1, 0, cv2.CV_32F)
    kernel /= np.sqrt((kernel * kernel).sum())
    filtered = cv2.filter2D(gray, -1, kernel)
    filtered = (filtered * 255).astype(np.uint8)
    result_image = cv2.cvtColor(filtered, cv2.COLOR_GRAY2BGR)

    if display:
        cv2_imshow(result_image)

    return result_image
```

- (21, 21): 커널의 크기 (폭, 높이)
- 5: 가우시안 커널의 표준 편차 (sigma)
- 1: 가보 함수의 방향 (theta)
- 10: 가보 함수의 주파수 (lambda)
- 1: 가보 함수의 종횡비 (gamma)
- 0: 가보 함수의 위상 (psi)
- cv2.CV_32F: 커널의 데이터 타입

- **그레이스케일 변환 및 정규화**
- **가보 커널 생성**
- **가보 커널 정규화 – > 합이 1 이 되도록 (출력 이미지 밝기 수준 관여)**
- 필터링 결과 정규화 및 타입 변환: uint8(일반적인 이미지 데이터타입)

비디오 처리 및 저장

```
cap = cv2.VideoCapture(file_name)
fourcc = cv2.VideoWriter_fourcc(*'mp4v')
out = cv2.VideoWriter('output_video.mp4', fourcc, 20.0, (int(cap.get(3)), int(cap.get(4))))

while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    processed_frame = process_frame(frame, display=True)
    if processed_frame is not None:
        out.write(processed_frame)

cap.release()
out.release()
cv2.destroyAllWindows()

# 동영상 다운로드 링크 제공
files.download('output_video.mp4')
```

업로드된 비디오 파일을 토대로 결과를 새로운 비디오 파일로 저장

결과물


output_fast.mp4


output_gabor.mp4

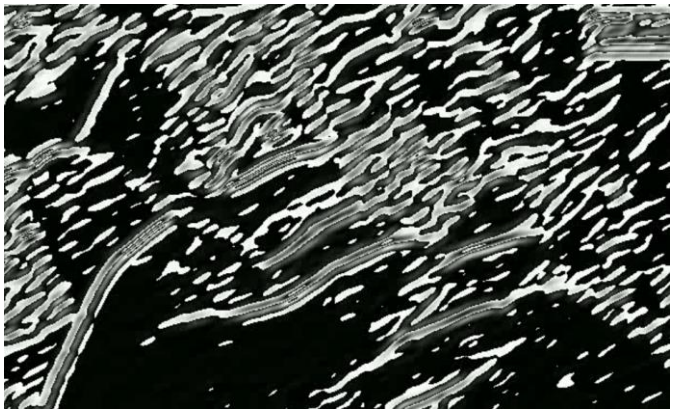

output_harris.mp4


output_hsv.mp4


output_video.mp4

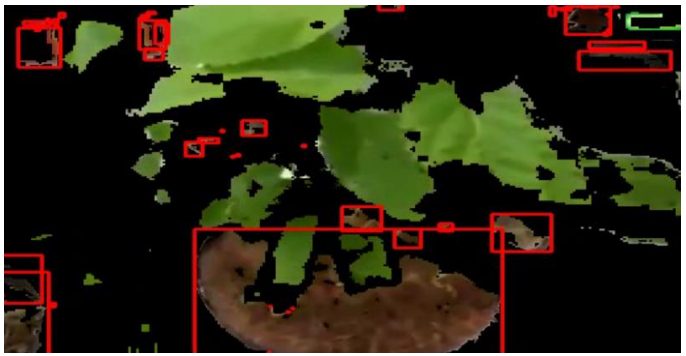
FAST 특징점

가보필터링



Harris 코너

HSV



최종 평가

- 주로 HSV 기반 색상 탐지가 병변을 명확하게 탐지하고 시각화하는 데 가장 효과적임.
- Harris 코너 검출과 FAST 특징점 검출, 가보 필터링은 추가적인 정보를 제공하지만 단독으로 병변을 효과적으로 탐지하는 데는 제한적이었음.

결론

- 본 프로젝트에서 다양한 컴퓨터 비전 알고리즘을 활용하여 이미지와 비디오에서 병변을 탐지하는 방법을 연구함.
- HSV 기반 색상 탐지가 병변 탐지에 가장 유효한 것으로 판단됨.
- 향후 연구에서는 알고리즘의 매개변수 변경 및 다양한 테스트를 통하여 실제 필드에 적용이 가능한 지 검토.

감사합니다