**OpenCV 2일차 보고서**

**로빛 20기 인턴 김민조**

**목차**

**1. Vision**

**2. OpenCV**

**3. Morphology**

**4. Filtering**

**5. Color space**

**1. Vision**

**(1) Vision이란?**

- 기계에 인간이 가지고 있는 시각과 판단 기능을 부여한 것

- 사람이 인지하고 판단하는 일을 기계의 하드웨어와 소프트웨어 시스템이 대신 처리하는 기술

**(2) Vision의 기본 목표**

- 디지털 이미지/비디오에서 유용한 정보 추출

- 의미 있는 설명이나 결정 도출

- 인간의 시각 시스템 능력 모방 및 확장

**(3) Vision의 기본 처리 단계**

**(a) 이미지 획득**

- 카메라나 센서를 통한 디지털 이미지 획득

**(b) 전처리**

- 노이즈 제거, 대비 향상 등 이미지 품질 개선

**(c) 특징 추출**

- 모서리, 윤곽선, 텍스처 등 중요 특징 검출

**(d) 세분화**

- 이미지를 의미 있는 영역으로 분할

**(e) 인식 및 해석**

- 객체 식별, 분류, 이해

**(4) Vision의 핵심 기술 및 알고리즘**

**(a) 이미지 처리 기법**

**- 필터링 및 엣지 검출**

- 가우시안 필터: 노이즈 제거에 효과적

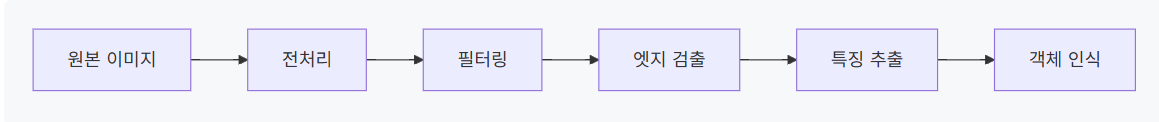
- 소벨 필터: 이미지의 수직 및 수평 엣지 검출

- 캐니 엣지 검출: 다단계 엣지 검출 알고리즘

**- 이미지 변환**

- 푸리에 변환: 이미지를 주파수 도메인으로 변환

- 허프 변환: 직선, 원 등 기본 도형 검출에 활용



**(b) 기계학습 기반 비전 기술**

**- 전통적 기계학습 접근법**

**-** SVM(Support Vector Machine): 이미지 분류에 활용

- SIFT(Scale-Invariant Feature Transform): 크기 불변 특징점 검출

- HOG(Histogram of Oriented Gradients): 객체 검출을 위한 특징 기술자

- **딥러닝 기반 접근법**

**-** CNN(Convolutional Neural Network): 이미지 분류, 객체 검출의 기반 구조

- R-CNN, Fast R-CNN, Faster R\_CNN: 객체 검출 딥러닝 모델

- YOLO(You Only Look Once): 실시간 객체 검출 알고리즘

- U-Net: 의료 이미지 세분화에 특화된 구조

- GAN(Generative Adversarial Network): 이미지 생성, 변환에 활용

**(4) Vision의 응용분야**

**(a) 산업 및 제조업**

- 제품 검사: 불량품 자동 검출, 품질 측정

- 바코드/QR 코드 인식: 물류 및 자동화 과정

- 로봇 비전: 부품 집기, 조립 등 로봇 작업 가이드

**(b) 의료 분야**

- 의료 영상 분석: X-RAY, CT, MRI 등의 분석 및 진단 보조

- 질병 진단: 암, 망막병증 등 질환의 조기 발견

- 수술 보조: 정밀 로봇 수술 시 시각 안내 시스템

**(c) 자율주행**

- 객체 인식: 차량, 보행자, 신호등, 도로 표지판 등 인식

- 경로 계획: 장애물 회피 및 최적 경로 설정

- 장면 이해: 도로 상황 인식 및 예측

도표, 라인, 스크린샷, 텍스트이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

**(d) 보안 및 감시**

- 얼굴 인식: 신원 확인, 접근 제어

- 이상 행동 감지: 공공장소에서 의심스러운 행동 식별

- 디지털 포렌식: 이미지/비디오 조작 검출

**(e) 증강현실 및 가상현실**

- 자세 추정: 사용자 움직임 추적

- 환경 인식: 실제 환경과 가상 객체 통합

- 마커리스 AR: 특정 마커 없이 환경 인식

**2. OpenCV**

**(1) OpenCV란?**

- Open Source Computer Vision Library의 약자로, 영상처리와 컴퓨터비전 관련 오픈 소스 라이브러리

- OpenCV를 사용하면 쉽게 영상 처리를 할 수 있고, 다양한 영상 처리 알고리즘을 지원

- 카메라와 같은 장치를 통한 영상 입력 기능 제공

- 영상 처리, 패턴 인식, 컴퓨터 비전 알고리즘을 구현한 풍부한 함수 제공

- 영상의 인식과 처리, 색상 공간의 변환, 필터링, DTF 및 FET, 주파수 변환 등의 다양한 기능들을 제공

- 다른 영상 처리 라이브러리에 비해 체계적으로 개발되고 있고, 오픈 소스이므로 자유롭게 사용 가능

**(2) OpenCV 버전 별 특징**

**(a) 0.1ver**

- C언어 기반 API 사용

- visual studio에서 라이브러리 컴파일 후 사용

- 구조체 기반 데이터 구조 사용

- highGUI 모듈에서 8비트 PNG, JPEG2000 입출력 지원

**(b) 2.0ver**

- C++언어 기반 API 사용

- 클래스 기반 데이터 구조 도입

- CMake를 이용하여 라이브러리 컴파일 후 사용 가능

- highGUI에서 스테레오 카메라 지원

**(c) 2.1ver**

- OpenMP에서 인텔 TBB로 병렬처리 루프를 변경

- Window & Mac OS X의 64비트 모드에서 OpenCV 빌드 가능

- 에러 체킹 코드 대신에 C++ try-catch문 사용

- Mac OS에서 Cocoa와 QTKit 지원

**(d) 2.2ver**

- 안드로이드 지원

- GPU 처리 지원

- 템플릿 자료구조 추가

- 5개의 기존 라이브러리를 12개의 작은 모듈로 재구성

**(e) 2.3ver**

- GPU 모듈에서 CUDA 4.0 지원

- Stitching 모듈에서 파노라마 지원

- 새롭게 제공되는 바이너리 패키지가 다양한 프리컴파일 라이브러리를 포함

**(f) 2.4ver**

- SIFT와 SUTF를 유료 모듈로 변경

- SIFT 성능을 대폭 개선

- 새 기본 클래스인 cv::Algorithm 도입

- 캐니 에지 알고리즘을 컬러 영상에서 수행 가능

**(g) 3.0ver**

- 모바일 CUDE 지원

- 기존 C++ API 대폭 개선

- IPP, FastCV 같은 저수준 API 지원

- cv::Algorithm을 적극 사용

**(3) OpenCV 모듈**

**(a) calib3d**

- 카메라 캘리브레이션과 3차원 재구성

**(b) core**

- 행렬, 벡터 등 OpenCV 핵심 클래스와 연산 함수

**(c) dnn**

- 심층 신경망 기능

**(d) features2d**

- 2차원 특징 추출과 특징 벡터 기술, 매칭 방법

**(e) flann**

- 다차원 공간에서 빠른 최근방 이웃 검색

**(f) highgui**

- 영상의 화면 출력, 마우스 이벤트 처리 등 사용자 인터페이스

**(g) imgcodecs**

- 영상 파일 입출력

**(h) imgproc**

- 필터링, 기하학적 변환, 색 공간 변환 등 영상 처리 기능

**(i) ml**

- 통계적 분류, 회기 등 머신 러닝 알고리즘

**(j) objdetect**

- 얼굴, 보행자 검출 등 객체 검출

**(k) photo**

- HDR, 잡음 제거 등 사진 처리 기능

**(l) stitching**

- 영상 이어 붙이기

**(m) video**

- 옵티컬 플로우, 배경 차분 등 동영상 처리 기술

**(n) videoio**

- 동영상 파일 입출력

**(o) world**

- 여러 OpenCV 모듈을 포함하는 하나의 통합 모듈

**3. Morphology**

**(1) Morphology란?**

- 영상 분야에서 노이즈 제거, 구멍 채우기, 끊어진 선 이어붙이기 등에 쓰이는 형태학적 연산

- 모폴로지 연산은 검은색과 흰색으로 구성되어있는 binary 이미지에 적용 가능

- 모폴로지 연산으로 침식, 팽창, 열림, 닫힘이 존재

**(2) Structuring Element**

- 원본 이미지에 적용되는 Kernel

- 수학적 형태학에서 Structuring Element란 이 형태가 이미지에 맞는지, 맞지 않는지를 결론내기 위해 주어진 이미지를 탐색하거나 작용하는데 사용되는 형태

- 팽창, 침식, 열림, 닫힘과 같은 형태학적 연산에 사용

- 가장 많이 사용되는 4가지 형태

직사각형, 도표, 사각형, 라인이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

**(3) 침식(Erosion)**

- 각 픽셀에 구조적 요소를 적용하여 하나라도 0이 있으면 대상 픽셀을 제거하는 방법

- ex) 십자형 구조적 요소를 원본 이미지에 적용하게 되고, 원본의 각 픽셀에 적용하여 겹치는 부분이 하나라도 있으면 그 중심 픽셀을 제거

도표, 라인, 평면도, 직사각형이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

- 작은 객체를 제거하는데 효과적

**(4) 팽창(Dilation)**

- 침식 연산과 반대로 객체를 확대한 후 작은 구멍을 채우는 방법

- 침식 연산과 마찬가지로 각 픽셀에 구조적 요소 적용

- 대상 픽셀에 OR연산을 수행해 겹치는 부분이 하나라도 있으면 이미지 확장

도표, 텍스트, 스크린샷, 디자인이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

- 경계가 부드러워지고 구멍이 메꿔지는 효과

**(5) 열림(opening)**

- 열림 연산은 침식 -> 팽창하는 연산으로 침식을 하면서 작은 범위의 픽셀은 사라지고 다시 팽창

- 큰 객체들은 사이즈 변화 X => 노이즈 제거에 효과적

**(6) 닫힘(closing)**

- 팽창 -> 침식을 하는 연산으로 제거가 아닌 이어주는 것이 목적

**4. Filtering**

**(1) 필터링이란?**

- 픽셀 주변의 값을 평균 내거나 강조해서 이미지를 부드럽게 만들거나 노이즈를 줄이는 처리 방식

- 영상처리에서는 이미지 품질 향상이나 전처리 단계로 주로 사용

**(2) Blur**

**(a) 평균 블러**

- 간단하고 빠른 흐림 효과

- cv::blur() or cv::boxFilter()가 기본적인 블러 필터

- 지정한 커널 크기 내의 평균 값으로 픽셀을 대체해 이미지를 부드럽게 만들어줌

- 가장자리에 아티팩트가 생기기 쉽고 고급 노이즈 제거엔 적합 X

**(b) 가우시안 블러**

- 부드럽고 자연스러운 흐림

- cv::GaussianBlur()는 픽셀 주변을 가우시안 분포 가중치로 평균하는 방식

- 중앙 픽셀에 가까울수록 더 많은 비중을 주기 때문에 자연스럽게 흐려짐

- 카메라 노이즈 제거, 엣지 전처리 등에서 자주 사용

**(c) 미디언 필터**

- cv::medianBlur()는 주변 픽셀의 중앙값을 선택해서 픽셀을 대체하는 방식

- 특히 salt&pepper 같은 점 형태의 노이즈에 매우 효과적

- 엣지를 유지하면서 노이즈 제거 가능

- But, 계산 비용이 높고 커널이 너무 크면 이미지가 뭉개짐 -> 적절한 크기 선택 중요

- 얼굴 인식 전처리, 디지털 문서 정리에 자주 사용

**(d) 필터 선택**

- 필터링 전후의 이미지를 비교하여 최적의 커널 크기를 찾으려면 수치 기반의 평가 지표를 사용

- 대표적으로 PSNR(Peak Signal-to-Noise Ratio), SSIM(Structural Similarity Index)을 이용해 필터링 전 이미지와 필터링 후 이미지를 비교한 뒤, 가장 높은 유사도를 보이는 커널 크기 선택 가능

- PSNR이 높을수록 원본에 가까운 결과, 흐림 효과를 유지하면서도 이미지 품질을 최대한 보존하는 커널 크기 결정 가능

**5. Color space**

**(1) Color space란?**

- 픽셀의 색을 숫자로 표현하는 방식

- 대표적인 색 공간: Binary, Grayscale, RGB, HSV 방식

**(2) Binary 이미지**

- 픽셀이 두 가지 색상, 주로 흰색과 검은색으로만 구성

- 각 픽셀은 단일 비트(0 or 1)로 표현

- 영상 분할, 객체 탐지 등 다양한 영상처리 기법에 활용

**(3) Grayscale 이미지**

- 색상의 차이를 나타내지 않고 밝기만으로 이미지를 표현하는 형태

- 각 픽셀의 값은 일반적으로 0~255사이로 표현(0은 검은색 255는 흰색)

- 픽셀 당 8비트, 즉 256단계의 명암 표현 가능

**(4) RGB 이미지**

- 빨강, 녹색, 파랑의 세 가지 원색 채널로 구성

- 각 채널은 0~255 범위의 값으로 해당 원색의 농도를 나타냄

- 각 색상은 8비트를 사용하므로 픽셀당 24비트 차지

- 이렇게 표현되는 이미지는 총 16,777,216가지의 다양한 색을 나타낼 수 있음

**(5) HSV 이미지**

- 색과 채도, 명도 모두 알 수 있는 이미지

- H(Hue): 일반적인 색을 의미하고 원추모형에서 각도로 표현

- S(Saturation): 색의 진함의 정도를 나타내고 중심에서 바깥쪽으로 이동하면 채도가 높음을 의미

- V(Value): 색의 밝고 어두운 정도를 나타내고 수직축의 깊이로 표현