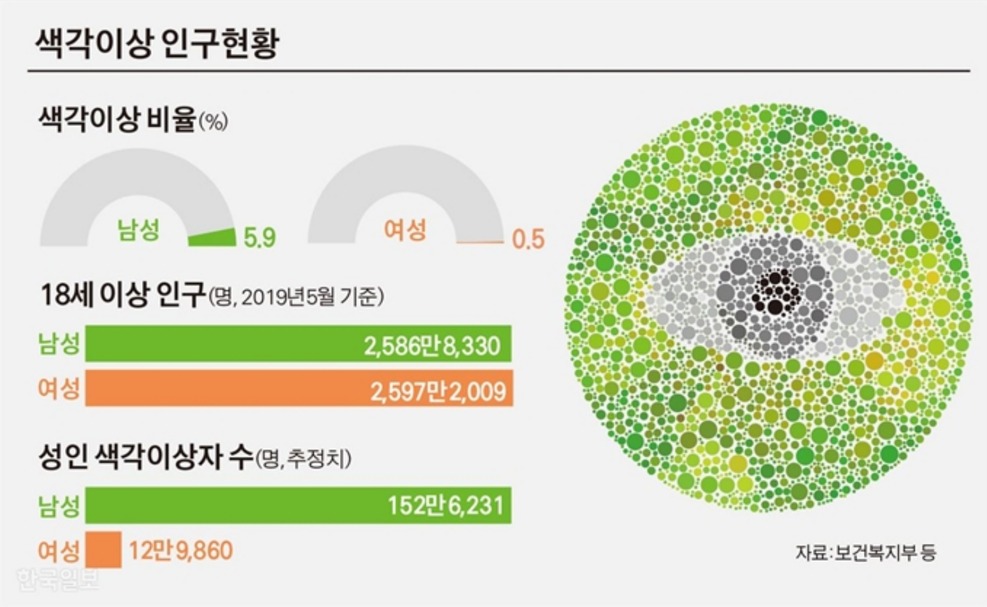
QD-OLED 활용
색각이상자를 위한 스마트 글라스

색조의 식별 능력이 없는 질환

제안 배경

색각이상자(색맹 + 색약)들의 인구현황과 대표적인 문제

특정한 색조를 인식하는 데 어려움을 느끼는 질환



색각이상자 인구현황

2021년 기준 대한민국 색각이상자는
국민 10명 중 3명 꼴

취업 제한

취업연령대인 색각이상자 20만 명이 취업에 제한을 받음

일상생활 속 어려움

대부분의 색각이상자가 일상생활 속 색조 인식에서 어려움을 겪음

-> 색각이상자들을 위한 아이템의 필요성을 깨달음

제안 배경

색각이상자들의 불편함을 돕기 위한 기존의 스마트 글라스

시간 지연

실시간으로 색상을 교정하는 과정에서 미세한 지연이 발생할 수 있어 빠른 반응이 필요한 상황에서 문제가 될 수 있음

효과 제한

주로 적록색맹에 효과적이고 청황색맹 등 다른 형태의 색맹에는 효과가 제한적임

성능 제한

조명 조건, 주변 환경의 색상 다양성 등에 따라 색상 교정 효과가 달라질 수 있어 일관된 성능을 보장하기 어려움

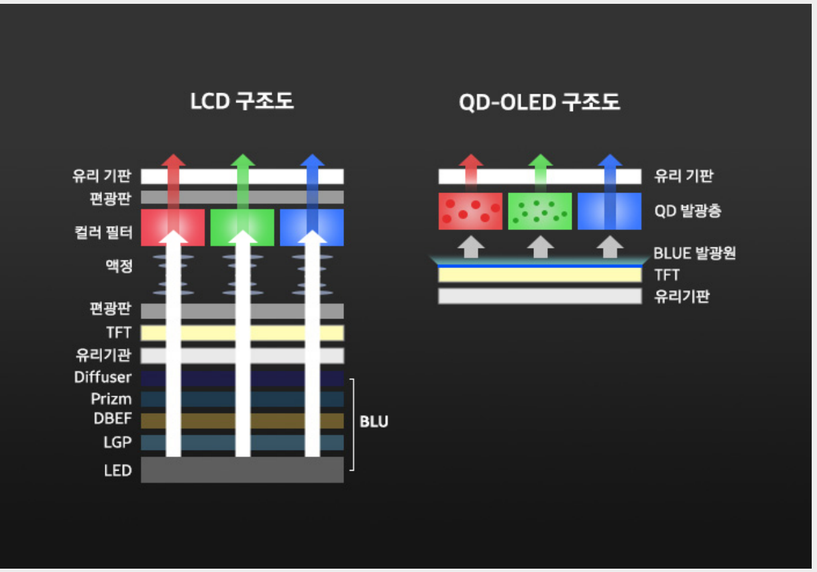
-> QD-OLED 기술을 활용해 이러한 한계를 극복할 수 있지 않을까?

제안 배경

기술 독창성과 활용 적합성

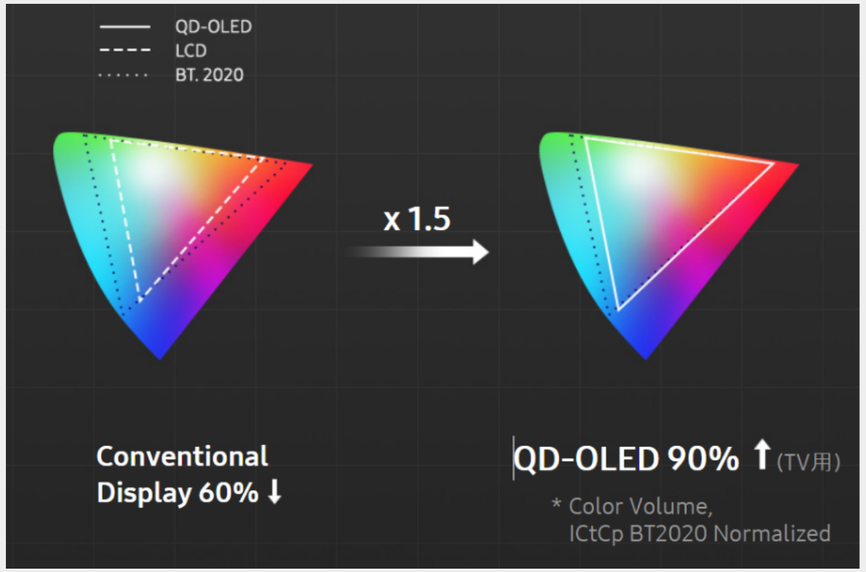
명암 단계별로 광범위하고 세밀하게 정확한 색 구현이 가능, 효율적인 빛의 활용과 간단한 구조
-> 차세대 디스플레이로 주목

QD-OLED



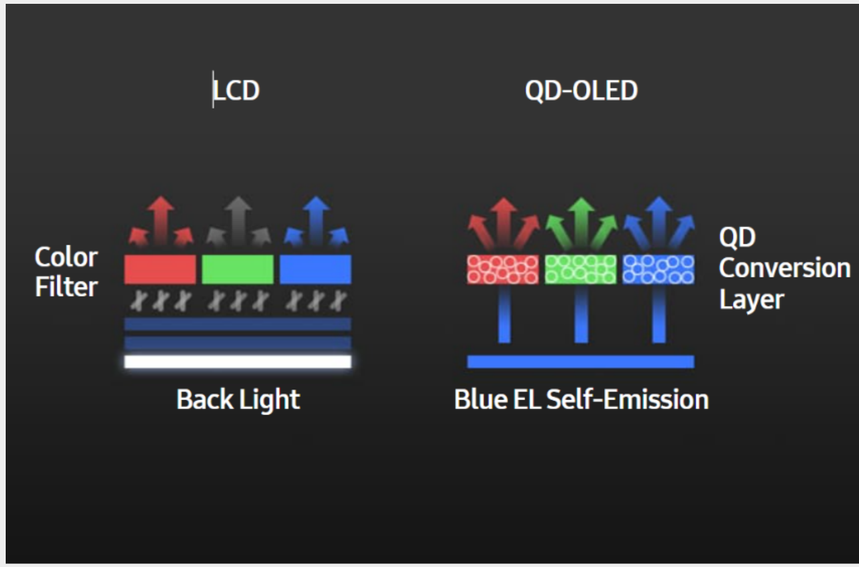
LCD에 비해 심플하고 효율적인 구조로 설계되어 얇고 가벼움

-> 안경 알에 활용하기 적합, 실시간 색상 교정 과정에서의 지연을 최소화



현존 디스플레이 중 가장 색표현 범위가 넓음

-> 적록색맹뿐만 아니라 청황색맹 등 다양한 유형의 색각이상자들의 이용에 적합, 색각이상자에게 더욱 높은 퀄리티의 색 경험 제공



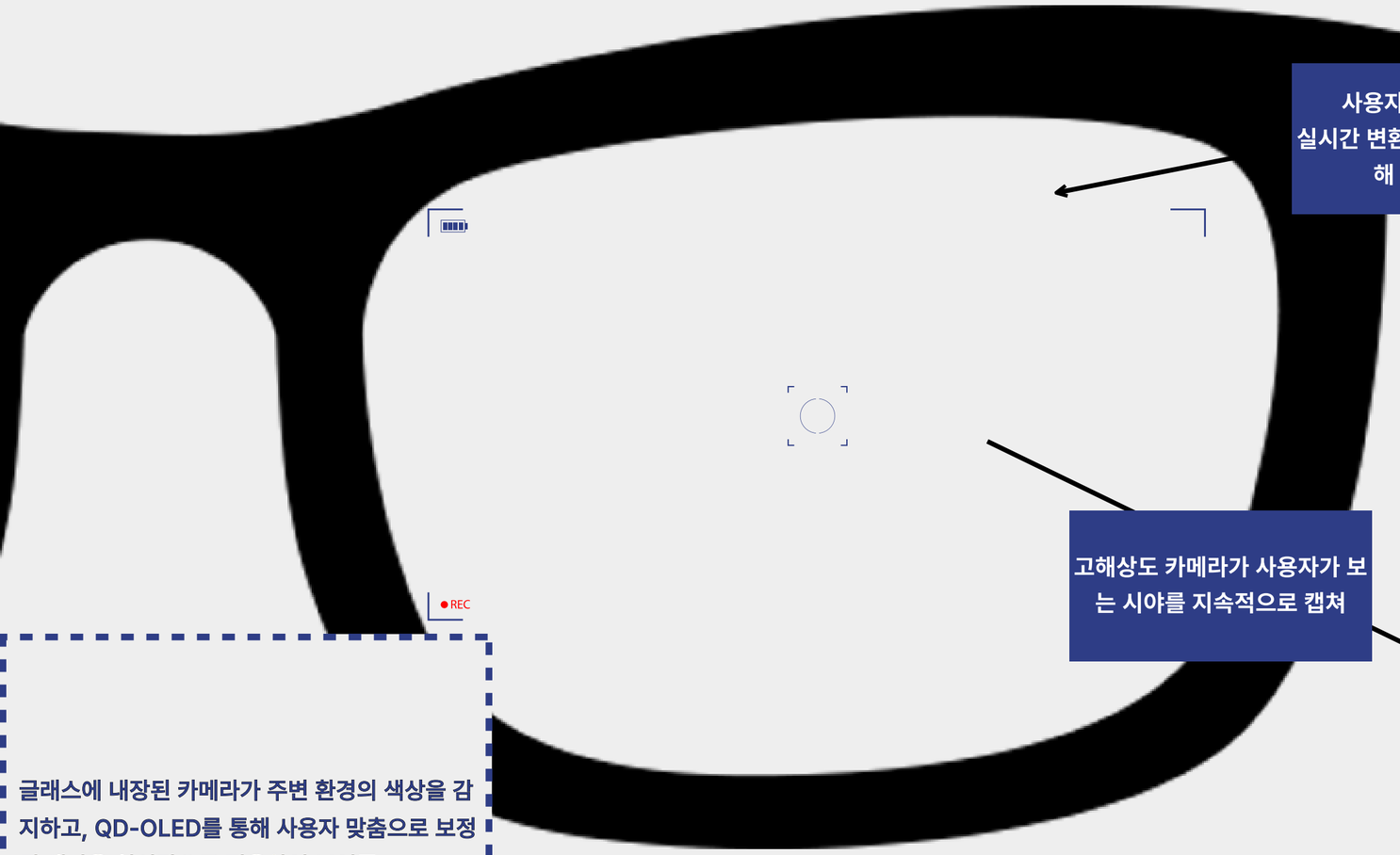
QD는 빛을 전방위로 균일하게 발광시키는 특성이 있어 균일한 휘도와 색감을 전달

-> 다양한 조명 조건과 환경 변화에도 항상 일관된 색상 보정과 밝기 경험 제공, 색각이상자에게 더욱 높은 퀄리티의 색 경험 제공

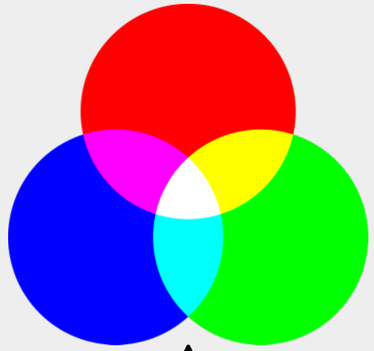
제품 개요

주요 기능

색상 감지 / 변환



사용자 맞춤 색상으로 실시간 변환 / QD-OLED를 통해 실시간 적용



카메라는 RGB 값으로 프로세서에 색상 정보를 기록

고해상도 카메라가 사용자가 보는 시야를 지속적으로 캡쳐

글래스에 내장된 카메라가 주변 환경의 색상을 감지하고, QD-OLED를 통해 사용자 맞춤으로 보정된 색상을 실시간으로 적용하여 보여줌

제품 개요

주요 기능

강조모드

사용자 인터페이스 버튼을 눌러
강조모드 활성화

탑재된 AI가 신호등 / 경고 표지
판 감지

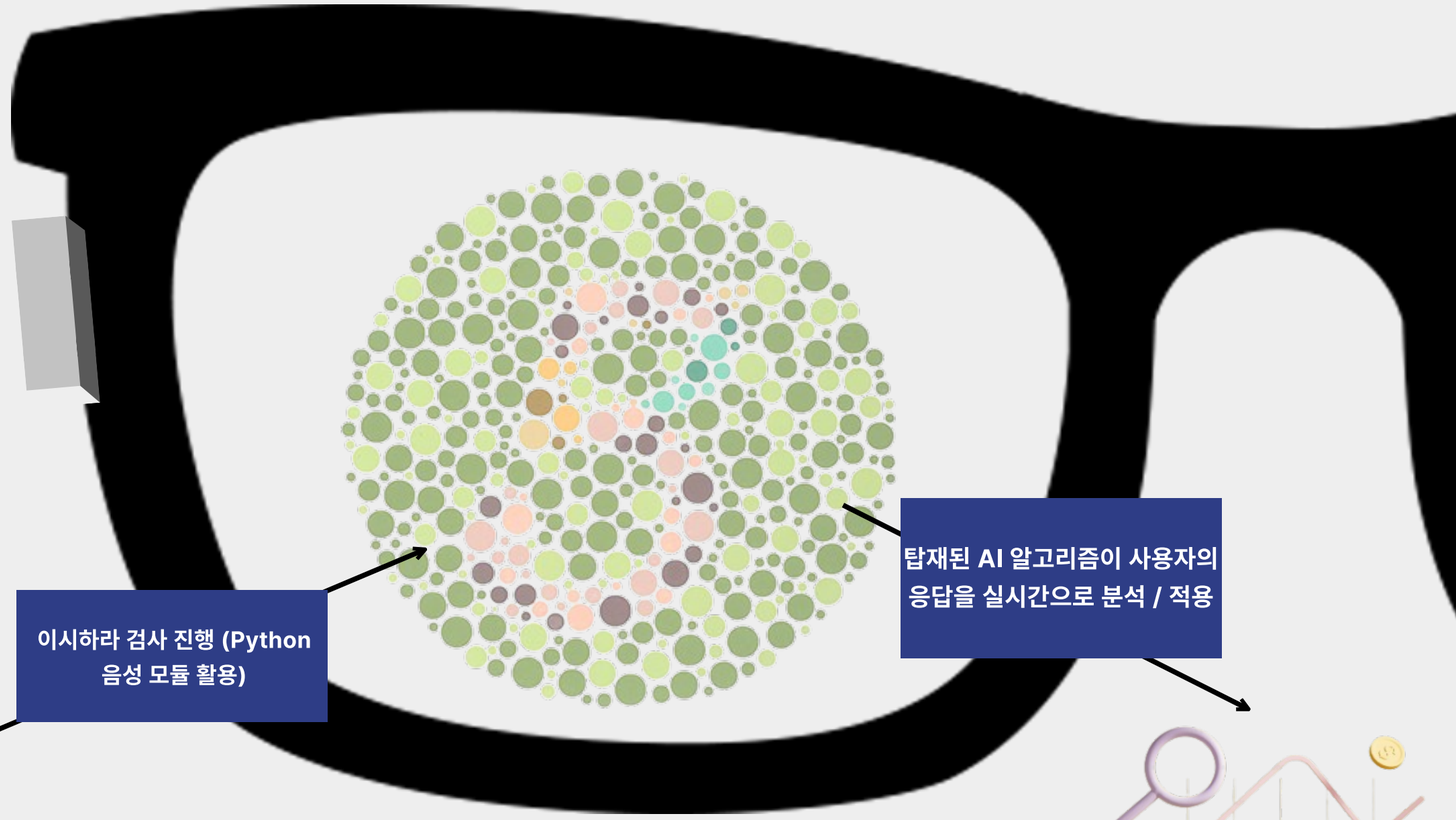
신호등 / 경고 표지판의
밝기를 높여 QD-OLED에 출력

교통 신호등, 경고 표지판 등 중요한 색을 강조해
표시하여 사용자가 즉시 인식할 수 있도록 도움

제품 개요

주요 기능

색각 테스트(초기사용시)



사용자 인터페이스 버튼을 길게 눌러 탑재된 디지털 이시하라 검사 애플리케이션 활성화

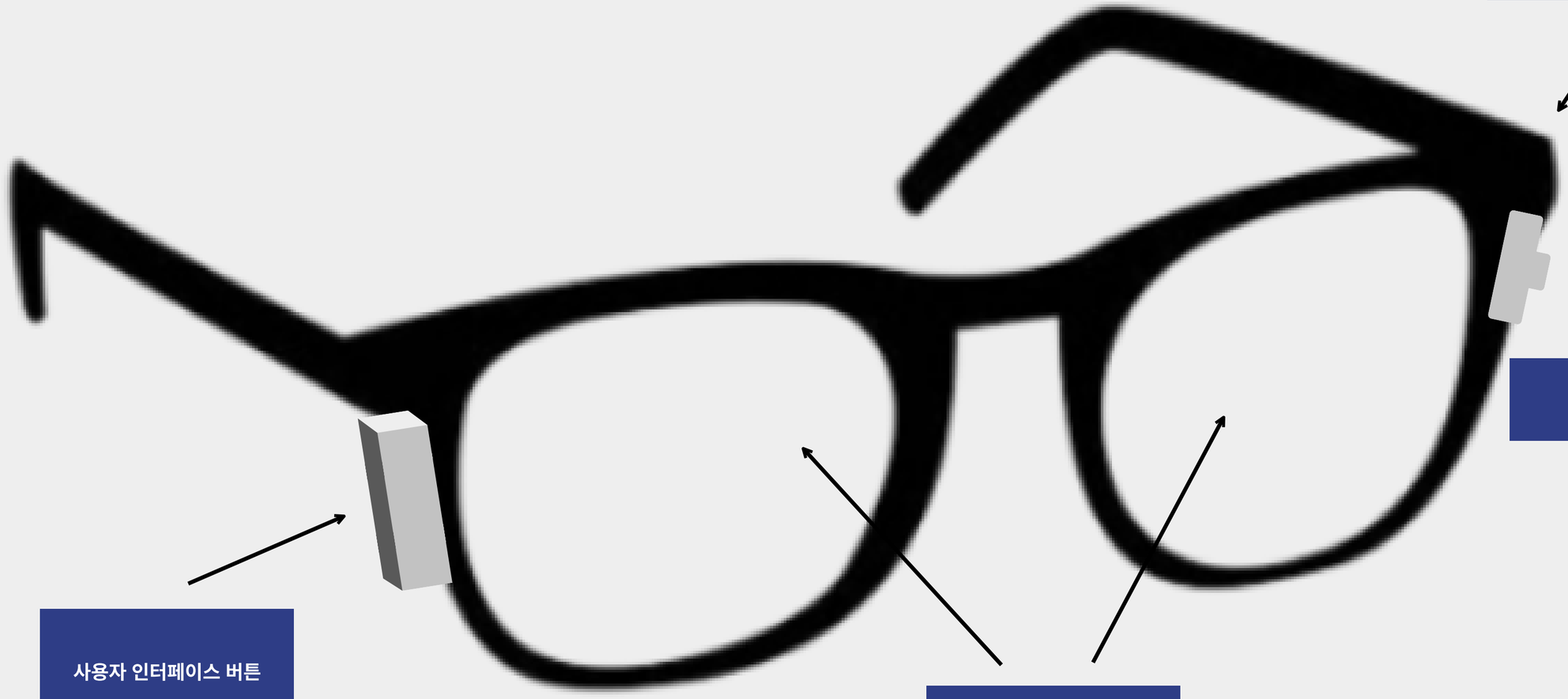
이시하라 검사 진행 (Python 음성 모듈 활용)

탑재된 AI 알고리즘이 사용자의 응답을 실시간으로 분석 / 적용

사용자에게 적록색맹, 청황색맹 등 다양한 유형의 색각 이상을 진단할 수 있는 이시하라 색각 검사 제공, 테스트를 통해 사용자가 어느 색상 범위에서 어려움을 겪는지 식별

제품 구조

일상생활 속 부담 없이 사용할 수 있는 디자인



색상 데이터 처리, AI / 애플리케이션 활성화, 실시간 보정 프로세서 내장

충전 단자

사용자 인터페이스 버튼

QD-OLED, 색상 데이터 수집 내장 카메라 탑재 안경 알

구현 방법

색상 감지 / 변환

정 파장의 빛을 선택적으로 통과시키거나 차단시켜 카메라의 색상 인식 성능을 향상시킴, 색상 왜곡을 줄이고 정확한 색상 데이터를 수집하는 데 중요한 역할

(1) 고해상도 카메라(작은 크기 / 광학 필터 포함)가 주변 환경의 색상을 초당 30프레임의 속도로 실시간 캡처

1.
고해상도 카메라

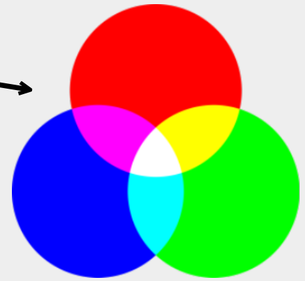
1080p 이상의 해상도를 제공하는 작은 크기의 CMOS 이미지 센서 이용

2.
광학 필터

RGB 필터, 편광 필터 활용

RGB 필터

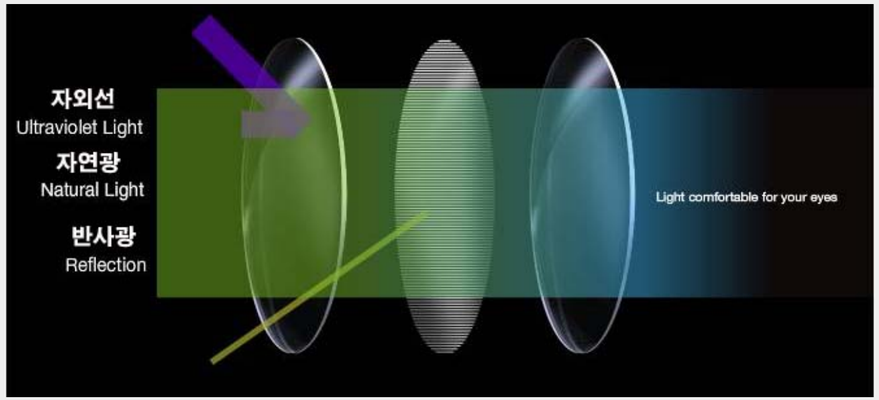
빨강, 초록, 파랑 빛을 분리하여 색상을 인지하는 데 사용



편광 필터

반사광을 줄이고 더 선명한 이미지를 제공해 상 인식 정확도를 높임

자연광은 그대로 두고 자외선과 반사광을 차단



CMOS 이미지 센서

1. 낮은 소비 전력
2. 빠른 처리속도로 인한 화질 우수
3. 스미어 현상 제거
4. 넓은 화각

구현 방법

색상 감지 / 변환

(2) 각 프레임의 픽셀 데이터를 분석해 RGB 값을 추출, 프로세서에 저장

1.

이미지 전처리

OpenCV를 사용해 노이즈 제거, 대비 조정 전처리 작업 수행

```python
import cv2
import numpy as np

# 이미지 불러오기
image = cv2.imread('예시.jpg')

# Gaussian 블러로 노이즈 제거
gaussian_blur = cv2.GaussianBlur(image, (5, 5), 0)

# Median 블러로 노이즈 제거
median_blur = cv2.medianBlur(image, 5)

#결과 저장
cv2.imshow('Original Image', image)
cv2.imshow('Gaussian Blur', gaussian_blur)
cv2.imshow('Median Blur', median_blur)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Gausssian, Median 블러를 사용한 노이즈 제거

히스토그램 균일화, CLAHE 적용한 대비 조정

```python
import cv2
import numpy as np

# 이미지 불러오기
image = cv2.imread('예시.jpg', cv2.IMREAD_GRAYSCALE)

# 히스토그램 균일화 적용해서 대비 조정
hist_equalized = cv2.equalizeHist(image)

# CLAHE 적용해서 대비 조정
clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
clahe_equalized = clahe.apply(image)

# 결과 저장
cv2.imshow('Original Image', image)
cv2.imshow('Histogram Equalized', hist_equalized)
cv2.imshow('CLAHE Equalized', clahe_equalized)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

구현 방법

색상 감지 / 변환

색상의 급격한 변화 탐지 / 경계 생성 알고리즘

(3) OpenCV에서 edge 검출 알고리즘을 이용해 프로세서에 저장된 색상의 경계 생성

1.

Canny edge 검출

비최대 억제, 이중 임계값 적용으로 더 정교하게 edge 검출

2.

Sobel 필터

수직 및 수평 방향의 그라디언트를 계산하여 edge 검출

```python
import cv2
import numpy as np

# 이미지 불러오기
image = cv2.imread('예시.jpg', cv2.IMREAD_GRAYSCALE)

# Canny edge 검출
low_threshold = 50
high_threshold = 150
edges = cv2.Canny(blurred_image, low_threshold, high_threshold)

# 결과 저장
cv2.imshow('Original Image', image)
cv2.imshow('Edges', edges)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Canny edge 검출

low_threshold와 high_threshold 값 조정으로 edge 검출의 민감도를 변경 후 Canny edge 검출

```python
import cv2
import numpy as np

# 이미지 불러오기
image = cv2.imread('예시.jpg', cv2.IMREAD_GRAYSCALE)

# Sobel 필터로 그라디언트 계산
sobel_x = cv2.Sobel(blurred_image, cv2.CV_64F, 1, 0, ksize=3)  # 수평 그라디언트
sobel_y = cv2.Sobel(blurred_image, cv2.CV_64F, 0, 1, ksize=3)  # 수직 그라디언트

# 두 그라디언트 절대값 계산 -> 합치기
sobel_combined = cv2.magnitude(sobel_x, sobel_y)

# 8비트 이미지로 결과 변환
sobel_combined = cv2.convertScaleAbs(sobel_combined)

# 결과 저장
cv2.imshow('Original Image', image)
cv2.imshow('Sobel X', cv2.convertScaleAbs(sobel_x))
cv2.imshow('Sobel Y', cv2.convertScaleAbs(sobel_y))
cv2.imshow('Sobel Combined', sobel_combined)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Sobel 필터

Sobel 필터를 이용한 수직 / 수평 그라디언트 계산, 이를 바탕으로 한 edge 검출

구현 방법

색상 감지 / 변환

(4) AI 기반 색상 보정

1.
CNN 모델 정의

OpenCV에서 CNN 모델 정의

딥러닝에서 주로 이미지나 영상 데이터를 처리할 때 쓰이는 Neural Network 모델

2.
모델 학습

이미지 데이터를 CNN에 맞게 조정하기 위해 데이터 로더 수정->
수정된 데이터 로더를 사용해 모델을 학습

3.
실시간 색상 보정

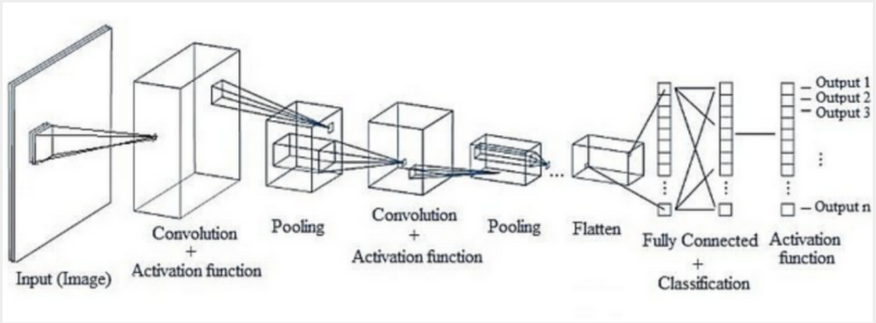
CNN 모델을 사용해 색상 보정을 하고 결과 저장

```python
import torch
import torch.nn as nn
import torch.nn.functional as F

class ColorCorrectionCNN(nn.Module):
    def __init__(self):
        super(ColorCorrectionCNN, self).__init__()
        self.conv1 = nn.Conv2d(3, 16, kernel_size=3, padding=1)
        self.conv2 = nn.Conv2d(16, 32, kernel_size=3, padding=1)
        self.conv3 = nn.Conv2d(32, 64, kernel_size=3, padding=1)
        self.fc1 = nn.Linear(64 * 32 * 32, 512)
        self.fc2 = nn.Linear(512, 3)

    #컨볼루션/ 활성화 함수 적용
    def forward(self, x):
        x = F.relu(self.conv1(x))
        x = F.max_pool2d(x, 2)
        x = F.relu(self.conv2(x))
        x = F.max_pool2d(x, 2)
        x = F.relu(self.conv3(x))
        x = F.max_pool2d(x, 2)
        x = x.view(-1, 64 * 32 * 32)
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return x
```

CNN 원리



CNN 모델 정의

이미지 데이터의 색상을 보정하기 위한 신경망 ColorCorrectionCNN 클래스 정의

구현 방법

색상 감지 / 변환

(4) AI 기반 색상 보정

1.
CNN 모델 정의

OpenCV에서 CNN 모델 정의

2.
모델 학습

이미지 데이터를 CNN에 맞게 조정하기 위해 데이터 로더 수정->
수정된 데이터 로더를 사용해 모델을 학습

3.
실시간 색상 보정

CNN 모델을 사용해 색상 보정을 하고 결과 저장

```python
import torch.optim as optim

criterion = nn.MSELoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

num_epochs = 100

for epoch in range(num_epochs):
    running_loss = 0.0
    for i, data in enumerate(dataloader, 0):
        inputs, targets = data
        inputs = inputs.permute(0, 3, 1, 2)  # (N, H, W, C) to (N, C, H, W)

        optimizer.zero_grad()

        outputs = model(inputs)

        loss = criterion(outputs, targets)
        loss.backward()
        optimizer.step()

        running_loss += loss.item()

    print(f"Epoch {epoch+1}, Loss: {running_loss/len(dataloader)}")
```

```python
model.load_state_dict(torch.load('color_correction_model.pth'))  # 학습된 모델 가중치 로드
model.eval()

# 색상 보정 함수
def correct_color(image_path, model):
    image= cv2.imread(image_path)
    image= cv2.resize(image, (128, 128))
    image =image/255.0
    image = torch.tensor(image, dtype=torch.float32).permute(2,0, 1).unsqueeze(0)

    with torch.no_grad():
        corrected_color= model(image).squeeze().permute(1,2,0).numpy()

    corrected_color= (corrected_color * 255).astype('uint8')
    corrected_color = cv2.resize(corrected_color, (640, 480))
    return corrected_color

# 색상 보정/ 결과 저장
input_image_path= '입력.jpg'
output_image_path = 'corrected_image.jpg'

corrected_image =correct_color(input_image_path, model)
cv2.imwrite(output_image_path, corrected_image)
```

모델 학습

수정된 데이터 로더를 사용해 순전파, 손실 계산, 역전파, 옵티마이저 스텝 수행

색상 보정

correct_color 함수가 전처리된 이미지 텐서를 모델에 입력해 보정된 색상을 얻음

구현 방법

색상 감지 / 변환

(5) QD-OLED를 활용해 디스플레이에 출력

1.
디스플레이 해상도

4K 고해상도 QD-OLED 디스플레이에 출력

```python
import cv2
import numpy as np

def display_image_fullscreen(image_path, display_resolution=(3840, 2160)):
    """
    QD-OLED 디스플레이에 전체 화면으로 출력
    """
    image = cv2.imread(image_path)

    if image is None:
        print(f"Error: Unable to load image at {image_path}")
        return

    #이미지를 디스플레이 해상도로 리사이즈
    image = cv2.resize(image, display_resolution)

    cv2.namedWindow('Corrected Image', cv2.WND_PROP_FULLSCREEN)
    cv2.setWindowProperty('Corrected Image', cv2.WND_PROP_FULLSCREEN, cv2.WINDOW_FULLSCREEN)
    cv2.imshow('Corrected Image', image)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

# 보정된 이미지 디스플레이
input_image_path = 'corrected_image.jpg'
display_image_fullscreen(input_image_path)
```

QD-OLED 디스플레이에 출력

구현 방법

색각 이상 유형 분석

1.

데이터 로드 / 전처리

모델이 데이터를 효율적으로 학습할 수 있도록 전처리

데이터 전처리 기술

데이터 정제
(Cleansing)

데이터 변환
(Transformation)

데이터 로드 / 전처리

데이터 로드, 크기 조정, 텐서로 변환

```python
import torch.optim as optim
from torch.utils.data import DataLoader

def train_model(train_loader, model, criterion, optimizer, num_epochs=10):
    for epoch in range(num_epochs):
        for images, labels in train_loader:
            optimizer.zero_grad()
            outputs= model(images)
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()
        print(f'Epoch {epoch + 1}/{num_epochs},Loss: {loss.item()}')

#예시 데이터
image_paths = ['path_to_image1', 'path_to_image2']
labels = [0, 1]  #0: 적록색맹 1: 청황색맹
dataset = ColorBlindnessDataset(image_paths, labels)
train_loader = DataLoader(dataset, batch_size=2, shuffle=True)

#모델 학습
model =ColorBlindnessCNN()
criterion= nn.CrossEntropyLoss()
optimizer= optim.Adam(model.parameters(), lr=0.001)
train_model(train_loader, model, criterion, optimizer, num_epochs=5)
```

구현 방법

색각 이상 유형 분석

2.
유형 분석

사용자의 시각 검사 결과를 바탕으로 색각 이상 유형을 분석

3.
결과 저장

사용자 맞춤 색상 보정 제공을 위해 결과를 프로세서에 저장

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

#사용자별로 평균 정확도 계산
user_data = data.groupby('user_id').mean().reset_index()

#K-평균 군집화로 사용자 분류
X = user_data[['accuracy', 'response_time']]
kmeans = KMeans(n_clusters=3, random_state=42).fit(X)
user_data['cluster'] = kmeans.labels_

#군집화 품질평가
score = silhouette_score(X, kmeans.labels_)
print(f'Silhouette Score: {score:.2f}')
```

```python
output_path = 'color_vision_analysis_results.csv'
user_data.to_csv(output_path, index=False)
print(f'Results saved to {output_path}')
```

결과 저장

이후에도 사용할 수 있도록 분석 결과를 CSV 파일로 저장

유형 분석

K-평균 군집화를 사용해 사용자를 색각 이상 유형으로 분류

K-평균 군집화

구현 방법

강조모드

```python
#YOLOv3 모델이 준비되었다는 가정에서 짠 코드
def detect_objects(img):
    height, width,channels = img.shape
    blob= cv2.dnn.blobFromImage(img, 0.00392, (416, 416), (0, 0, 0), True, crop=False)
    net.setInput(blob)
    outs = net.forward(output_layers)

    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:
                center_x = int(detection[0]* width)
                center_y = int(detection[1]*height)
                w = int(detection[2]*width)
                h = int(detection[3]*height)
                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)
    return boxes, confidences,class_ids,indexes
```

1.

신호등 / 경고 표지판 인식

YOLO 모델을 사용해 신호등 / 경고 표지판을 실시간으로 탐지

높은 정확도와 빠른 속도를 자랑하는 딥러닝 Object Detection 모델

신호등 / 경고 표지판 인식

각 객체를 인식 / 객체의 위치와 클래스 ID를 반환

객체 강조

객체의 위치 ID를 기반으로 해당 영역을 더 밝게 처리

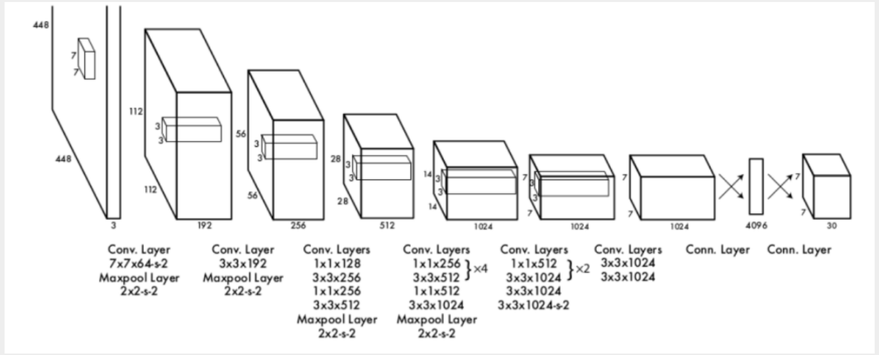
```python
def highlight_objects(img,boxes,indexes):
    for i in range(len(boxes)):
        if i in indexes:
            x,y,w,h=boxes[i]
            # 객체 영역을 더 밝게/강조
            sub_img=img[y:y+h, x:x+w]
            white_rect=np.ones(sub_img.shape,dtype=np.uint8)*255
            res=cv2.addWeighted(sub_img,0.5,white_rect,0.5,1.0)
            img[y:y+h,x:x+w]=res
    return img
```

2.

객체 강조

탐지된 객체의 밝기를 올려 강조

YOLO 모델 원리



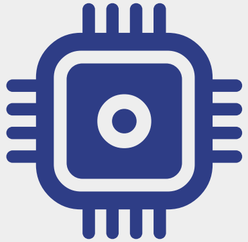
YOLO 모델 객체 인식 예시

기대 효과

QD-OLED 활용 색각이상자를 위한 스마트 글라스

삶의 질 향상

색각이상자가 스마트 글라스를 착용함으로써 색상을 더 정확하게 구분할 수 있고 이를 통해 옷차림, 요리, 예술 활동 등 일상 생활에서의 색상 선택 / 구별이 쉬워짐



신기술 발전

QD-OLED / 웨어러블 디바이스 / AI 기술 융합을 통해 각 시장이 확대되고 융합되어 신기술 발전을 기대할 수 있음

안전성 향상

스마트 글라스의 강조모드를 통해 교통 상황 / 산업 현장에서의 사고를 예방할 수 있어 색각이상자의 생활 안전성이 향상됨

Q&A

기술적 코멘트와 답변

Q1

사용자가 인식 불가능한 색상을 인식할 수 있도록 해주는 스마트 글라스인가요?

Q2

사용자가 인식하는 물체에 대한 색상을 덧입히는 것인가요?

Q3

데이터를 처리하는 부분 등에서 발생하는 전력 소비에 대한 방안은 고안하지 않았나요?

Q4

색각이상자가 필요로 하는 색상 부분에 대하여 글라스에 색상을 변환하는 값을 미리 조정하여 반영하는 방법이 더 효율적이지 않을까요?

A1

아닙니다. 사용자가 식별에 어려움을 겪는 색상을 인식 가능한 범위로 보정해 일상생활 속 불편을 줄이고 더 풍부한 시각 경험을 제공하는 스마트 글라스입니다.

A2

아닙니다. 초당 30프레임 속도로 캡쳐된 이미지를 보정해 출력하는 것입니다. 쉽게 말해 혼합현실의 개념입니다.

A2

전력 효율성 극대화를 위해서 GPU를 가동하고, 고용량 배터리를 탑재하는 방법을 생각했습니다.

A4

더 높은 퀄리티의 색 경험을 제공하기 위해 색상 뿐만 아니라 휘도와 밝기 등 구체적인 조정이 필요합니다. 그래서 상황에 따라 실시간으로 보정을 하는 방법을 고안하였습니다.

출처

https://www.kmib.co.kr/article/view.asp?arcid=1713254315
https://www.wikitree.co.kr/articles/824687
https://ipsi.scau.ac.kr/Application/08_Etc/popup_color.html
https://www.samsungdisplay.com/kor/tech/quantum-dot.jsp
[https://ssgjn.tistory.com/entry](https://ssgjn.tistory.com/entry/%EC%83%89%EB%A7%B9-%EC%83%89%EC%95%BD-%EA%B5%90%EC%A0%95-%EC%95%88%EA%B2%BD-%EC%97%94%ED%81%AC%EB%A1%9C%EB%A7%88-%EC%9B%90%EB%A6%AC-%EC%83%89%EB%A7%B9-%EC%83%89%EC%95%BD-%EA%B2%80%EC%82%AC#google_vignette)
http://blueedu.dothome.co.kr/xe/astro/29322
http://unithink.co.kr/product/ir-cut-filter/343/
[http://bglasses.co.kr/home/contents/mobile/board.php?mode=view&id=polarizing&no=81&page=2&search[word]=](http://bglasses.co.kr/home/contents/mobile/board.php?mode=view&id=polarizing&no=81&page=2&search%5Bword%5D=)
https://news.skhynix.co.kr/post/skhynix-next-generation-cmos-image-sensor-a4c
[https://velog.io/@nayeon_p00/딥러닝-모델-CNNConvolutional-Neural-Network](https://velog.io/@nayeon_p00/%EB%94%A5%EB%9F%AC%EB%8B%9D-%EB%AA%A8%EB%8D%B8-CNNConvolutional-Neural-Network)
https://tech.hyundai-rotem.com/digital/technology-that-adds-value-to-big-data-data-visualization/
https://brunch.co.kr/@aischool/11
[https://velog.io/@openjr/TIL-13W25D-객체-인식-프로젝트](https://velog.io/@openjr/TIL-13W25D-%EA%B0%9D%EC%B2%B4-%EC%9D%B8%EC%8B%9D-%ED%94%84%EB%A1%9C%EC%A0%9D%ED%8A%B8)
[https://tyami.github.io/machine learning/k-means-clustering/](https://tyami.github.io/machine%20learning/k-means-clustering/)

감사합니다