Ömer Faruk Dinçoğlu

Intel RealSense D435i

kullanılarak derinlik algılama hareket takibi yapabildiğimiz stero kamera.

Derinlik algılamada 10 m ye kadar menzil sunuyor. 0.2 ile 3 m arası optimum sonuçları alabiliyoruz.85 derece görüş açısı var 30fps görüntü sağlıyor

Renkli de Global shutter hareketli sahneler için ideal.

6 eksenli hareket sensörü mevcut.

Kızılötesi sensörlele düşük ışıkta yüksek performasn sağlanıyor .Intel'in özel derinlik algoritmaları derinlik algılamasında fark yaratıyor.

IR sensörü doku olmayan bölgelerde dahi derinlik ölçümü yapmamamızı sağlıyor

Kullanım alanlar:

Nesne tanıma ve manipülasyonu

Engelleri algılayıp kaçabilme

Otonom teknolojilerinde

Navigasyonlarda

Arttırılmış Gerçeklik uygulamalarında :

Realistic maping

Nesneyi istediğimiz lokasyona yerleştirme

3D modellemede:

Küçük büyük nesnelerin doğru ölçeklerde modellenmesi

Ortam haritalama

Hacim mesafe ve nesne boyutlama ve ölçme

Elde ettiğimiz görüntüleri çeşitli filtre ve algoritmalardan geçirip istediğim gibi kullanabiliriz. Cam ayna gibi yansıtıcı yüzeyler derinlik algılamada dezavantajlı duruma düşürüyor bizi.

Nesnelerin kalıbını alıp blender gibi uygulamalarda ufak düzeltmeler yardımıyla kolayca obj dosyalarının elde edilmesinde önemli rol oynar.

Verimlilik artışı için sanayi kuruluşlarında nesnelerin etrafı gezilerek maximum faydanın nasıl alınabileceğine karar mekanizması olarak kullanılabilir.

Mesela bir kargo şirketinde paketlerin depolanmasında hangi alanın hangi paketi koyduğumuzda daha verimli kullanabileceğimizi veya fiyat hesabı yaparken yardımcı olması.

Fabrikalardaki üretim bandından çıkan ürünlerde farklılık olup olmadığını kontrol etme. (Kalite kontrol)

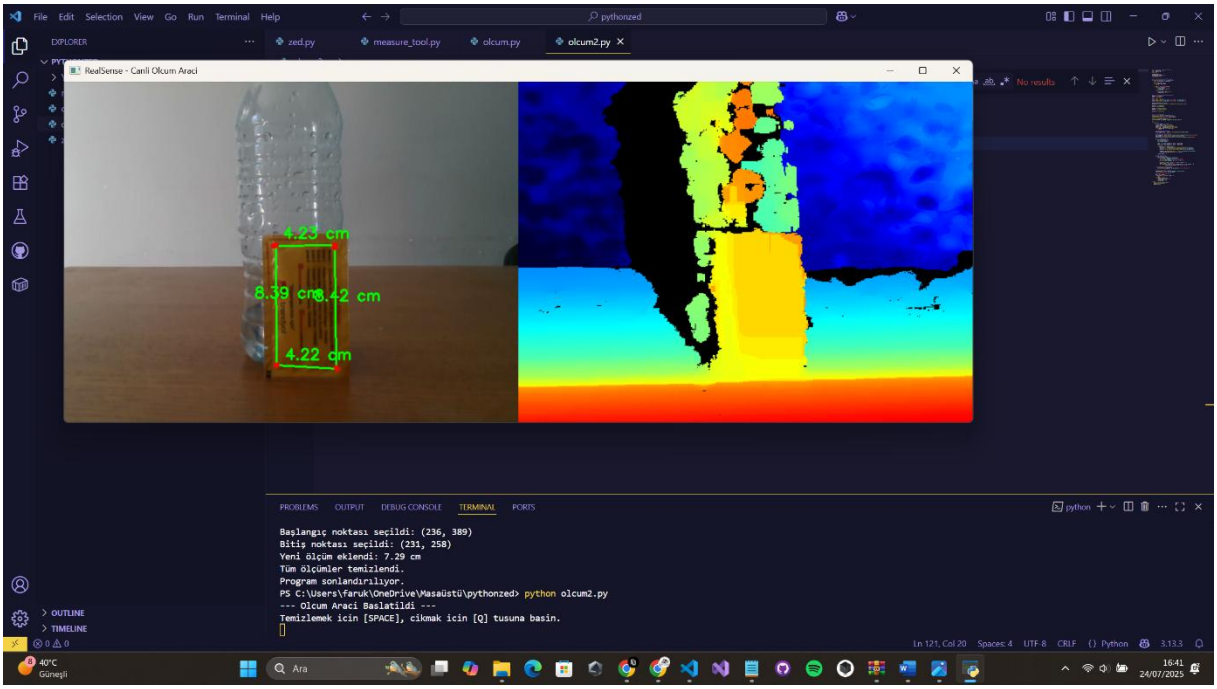
Tarım alanlarında meyvelerin/sebzelerin olgunlaşıp olgunlaşmadığını kontrol etme.

Bir arsaya oraya yapılması gereken binayı 3d model olarak yerleştirmek istediğimizde .

Örneğin Google Haritalar uygulamasında yapılan Street viev lerin işlenmesinde doğru ölçek renk vs sağlanıp modellenmesinde kullanılabilir.

Örnek proje:

El ile bir nesnenin etrafında dolaşarak ölçüm yapılması



Kod kısmı:

```
### 1. Kütüphaneler ve Global Değişkenler
import pyrealsense2 as rs
import numpy as np
import cv2

current_points = []
completed_measurements = []
is_measuring = False

### 2. Fare Tıklamalarını Yöneten Fonksiyon
def mouse_callback(event, x, y, flags, param):
    global current_points, is_measuring
```

```
    color_image_width = 640
    if x >= color_image_width:
        return

    if event == cv2.EVENT_LBUTTONDOWN:
        if not is_measuring:
            current_points = [(x, y)]
            is_measuring = True
        else:
            current_points.append((x, y))
            is_measuring = False

### 3. Intel RealSense Kamera Yapılandırması
pipeline = rs.pipeline()
config = rs.config()

WIDTH, HEIGHT = 640, 480
config.enable_stream(rs.stream.depth, WIDTH, HEIGHT, rs.format.z16, 30)
config.enable_stream(rs.stream.color, WIDTH, HEIGHT, rs.format.rgb8, 30)

profile = pipeline.start(config)
depth_scale = profile.get_device().first_depth_sensor().get_depth_scale()

align_to = rs.stream.color
align = rs.align(align_to)

spatial = rs.spatial_filter()
temporal = rs.temporal_filter()

colorizer = rs.colorizer()

### 4. Ana Program Döngüsü
WINDOW_TITLE = 'RealSense - Canli Olcum Araci'
cv2.namedWindow(WINDOW_TITLE, cv2.WINDOW_AUTOSIZE)
cv2.setMouseCallback(WINDOW_TITLE, mouse_callback)

print("--- Olcum Araci Baslatildi ---")
print("Temizlemek icin [SPACE], cikmak icin [Q] tusuna basin.")

try:
    while True:
        # Veri akışını al ve hizala
        frames = pipeline.wait_for_frames()
        aligned_frames = align.process(frames)
        depth_frame = aligned_frames.get_depth_frame()
        color_frame = aligned_frames.get_color_frame()
        if not depth_frame or not color_frame:
            continue

        # Derinlik verisini filtrele
        filtered_depth_frame = temporal.process(spatial.process(depth_frame))

        # Görüntüleri ve verileri hazırla
        color_image_bgr = cv2.cvtColor(np.asanyarray(color_frame.get_data()), cv2.COLOR_RGB2BGR)
        depth_image = np.asanyarray(filtered_depth_frame.get_data())
        depth_colormap = np.asanyarray(colorizer.colorize(filtered_depth_frame).get_data())
```

```
        # Ölçüm hesaplamasını yap
        if len(current_points) == 2:
            p1 = current_points[0]
            p2 = current_points[1]

            depth_m_p1 = depth_image[p1[1], p1[0]] * depth_scale
            depth_m_p2 = depth_image[p2[1], p2[0]] * depth_scale

            if depth_m_p1 > 0 and depth_m_p2 > 0:
                depth_intrin = depth_frame.profile.as_video_stream_profile().intrinsics
                point1_3d = rs.rs2_deproject_pixel_to_point(depth_intrin, p1, depth_m_p1)
                point2_3d = rs.rs2_deproject_pixel_to_point(depth_intrin, p2, depth_m_p2)

                distance = np.sqrt(sum([(a - b) ** 2 for a, b in zip(point1_3d, point2_3d)]))
                completed_measurements.append([p1, p2, distance])

            current_points = []

        # Görselleştirmeyi yap
        if completed_measurements:
            for p1, p2, dist in completed_measurements:
                cv2.line(color_image_bgr, p1, p2, (0, 255, 0), 2)
                cv2.circle(color_image_bgr, p1, 4, (0, 0, 255), -1)
                cv2.circle(color_image_bgr, p2, 4, (0, 0, 255), -1)

                mid_point = ( (p1[0] + p2[0]) // 2, (p1[1] + p2[1]) // 2 )
                text = f"{dist * 100:.2f} cm"
                cv2.putText(color_image_bgr, text, (mid_point[0] - 30, mid_point[1] - 10),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0), 2)

        if is_measuring and current_points:
            cv2.circle(color_image_bgr, current_points[0], 4, (0, 255, 255), -1)

        # Görüntüleri birleştir ve göster
        combined_image = np.hstack((color_image_bgr, depth_colormap))
        cv2.imshow(WINDOW_TITLE, combined_image)

        # Kullanıcı girdisini kontrol et
        key = cv2.waitKey(1)
        if key & 0xFF == ord('q') or key == 27:
            break
        elif key == ord(' '):
            completed_measurements = []
            current_points = []
            is_measuring = False
finally:
    print("Program sonlandiriliyor.")
    cv2.destroyAllWindows()
    pipeline.stop()
```