GÖRÜNTÜ IŞLEME PROJE ÖDEVI

HAZIRLAYANLAR:

Faruk MUSA
22010903132

Taha Melih ÖZTAŞ
b210109001

Umut Kuzey
GÖRGEÇ
22010903075

DR. ÖĞR. UYESİ MUHAMMED ALİ NUR ÖZ

Giriş:

Amacımız video kaydındaki nesneleri(araçları ve yayaları) tespit edip hızlarını(pixel/saniye) belirleyip onları takip etmektir. Geri planda, hareketli nesneleri izlemek için bir arka plan güncelleme yöntemi (background subtraction) ve nesne tespit algoritması kullanılır.

KULLANDIĞIMIZ YÖNTEMLER:

1. Gri Tonlamaya Çevirme

Neden Gri Tonlama Kullanıldı?

Bir video akışındaki nesneleri takip etmek için ilk adım, her pikselin ışık yoğunluğunun hesaplanmasıdır. Gri tonlama, renkli görüntülerde her pikselin kırmızı, yeşil ve mavi bileşenlerini tek bir değere indirger. Renkli Görüntüler vs. Gri Tonlama: Renkli görüntülerde her piksel için 3 farklı bileşen bulunur (RGB), bu da hesaplamayı daha karmaşık hale getirir. Gri tonlama ise yalnızca bir bileşen (ışık yoğunluğu) kullanarak, işlemeyi hızlandırır.

2. Arka Plan Çıkarma ve Güncelleme (Hareketli Ortalamalar)

Neden Bu Yöntem Kullanıldı?

Arka planın sürekli güncellenmesi, sadece hareketli nesnelerin tespit edilmesini sağlar. cv2.accumulateWeighted() fonksiyonu. Bu fonksiyon, her karede arka planın ortalama değerini yavaşça günceller. Sabit (hareketsiz) arka plan güncellendikçe, hareketli nesneler farklılık gösterir ve bu fark, nesnelerin tespit edilmesine olanak tanır.

3. Mesafe Hesaplama (scipy.spatial.distance.cdist())

Neden Bu Fonksiyon Kullanıldı?

Nesne takibinde, yeni tespit edilen nesneler ile önceki nesneler arasındaki mesafeyi hesaplamak gereklidir. Bu mesafe, nesnelerin doğru bir şekilde eşleştirilmesine yardımcı olur. Bu fonksiyon, mesafeleri hızlı bir şekilde hesaplamamıza olanak tanır. Nesnelerin merkez noktalarının arasındaki öklid mesafesini hesaplar. Bu mesafeler, nesnelerin birbirlerine yakın olup olmadığını belirler. Yakın olan nesneler eşleştirilir ve takip edilir.

4. Morfolojik İşlemler (cv2.morphologyEx())

Neden Kullanıldı?

Morfolojik işlemler, maskenin düzgün bir hale gelmesi, hareketli nesnelerin daha doğru bir şekilde tespit edilmesini sağlar. Biz de projede kapama(closing) işlemi kullanarak küçük boşlukları doldurduk. Kapama işlemi önce yayma(dilation) sonra aşındırma(erosion) yapılarak gerçekleştirilir.

5. Hız Hesaplama

Nesnelerin hareket hızını hesaplamak, onları doğru bir şekilde sınıflandırmamıza yardımcı olur. Taşıtlar genellikle hızlı hareket ederken, yayalar daha yavaş hareket eder. Bu, nesnelerin tipini belirlemek daha kolay olur. Hesaplanması ise nesnelerin merkezleri arasındaki mesafenin ölçülmesi ve bu mesafenin zamana bölünerek hızın elde edilmesi ile bulunur (px/saniye).

6.Yaya ve Taşıtların Kurallara Uygunluğunun Tespit Edilmesi

Tespit ettiğimiz nesnelerin yanında bir de yaya yollarını ve kaldırımların konumlarını, her video için girmemiz gerekiyor. Eğer tespit ettiğimiz nesne, bir taşıt ise ve yaya geçidinde de o sırada bir yaya varsa, taşıt kırmızı kare içine alınır ve ekranda gösterilir. Eğer yayalar kaldırımda veya yaya yolunda değilse de taşıt yolunda kabul edilirler ve onlar da kırmızı kare içine alınırlar.

AVANTAJLAR VE ALTERNATİF YOLLAR:

GRİ TONLAMAYA ÇEVİRME:

Avantajlar:

Tek bir değer üzerinden işlem yaptığımız için, daha az hesaplama gerektirir. Renkli görüntüler yerine gri tonlar olduğu için bellek kullanımını azalır.

Alternatif Yöntemler: Eğer renk önemli olsaydı, renkli (RGB) görüntüleri de işleyebilirdik. Ancak bu, işlem gücü gereksinimini artırır ve video akışlarında yavaşlamaya sebep olabilir.

ARKAPLAN ÇIKARMA VE GÜNCELLEME:

Avantajlar:

Daha doğru hareket tespiti. Hareketsiz arka plan sabit kalırken, hareketli nesneler kolayca fark edilir. Anlık ışık değişimleri ve küçük gürültüler arka planın güncellenmesi sayesinde göz ardı edilir.

Alternatif Yöntemler: cv2.backgroundSubtractorMOG2() gibi başka arka plan çıkarım yöntemleri de vardır, ancak bu yöntem daha karmaşık ve işlemci gücü gerektirir. cv2.accumulateWeighted() daha hızlı ve yapılandırılması daha basittir.

MESAFE HESAPLAMA:

Avantajlar:

Mesafelerin hızlıca hesaplanması, takip sürecini hızlandırıyor. En yakın nesneler doğru bir şekilde eşleştirilir.

Alternatif Yöntemler: Manuel mesafe hesaplaması da yapılabilirdi, ancak bu daha karmaşık olurdu. cdist() fonksiyonu, bu işlemi daha verimli hale getirir.

MORFOLOJİK İŞLEMLER:

Avantajlar:

Maske üzerindeki gürültüleri ortadan kaldırarak, nesnelerin daha doğru şekilde tespit edilmesini sağlar. Hareketli nesnelerin sınırlarını daha düzgün hale getirir.

Alternatif Yöntemler: Diğer görüntü işleme yöntemleri (örneğin Gaussian Blur) de kullanılabilir, ancak morfolojik işlemler daha hızlıdır ve özellikle video akışları gibi gerçek zamanlı işlemler için iyidir.

HIZ HESAPLAMA:

Avantajlar:

Nesnelerin hızına göre, taşıt ve yaya ayrımını yapabiliriz. Hız, nesnelerin gerçek hareketlerini yansıtarak daha doğru sonuçlar sağlar.

Alternatif Yöntemler: Yalnızca nesnenin alanına ve en-boy oranına bakarak da sınıflandırma yapılabilirdi, ancak hız, nesnelerin türünü daha doğru şekilde belirler.

HATALAR VE ÇÖZÜM FİKİRLERİMİZ:

1. **VİDEONUN BELİRLİ BİR KISMINDA BOZUK OLMASI:**
   ÇÖZÜM: Bu sorundan hareketli ortalamalar yöntemini kullanarak kurtulduk. Bu yöntem daha önce açıklandı.

2. **ARAÇ VE YAYA TESBİTİNİN TAM DOĞRU OLMAMASI VE TOPLAM SAYILARIN YANLIŞ OLMASI:**
   ÇÖZÜM: Daha gelişmiş bir makine öğrenmesi modeli kullanarak nesneleri sınıflandırmak, doğruluğu artırabilir. Örneğin, YOLO (You Only Look Once) veya Haar Cascade Classifier gibi derin öğrenme tabanlı nesne tespit algoritmalarını kullanmak, özellikle karmaşık ve dinamik sahnelerde sınıflandırma hatalarını azaltabilir.

3. **SABİT BİR HIZ TESBİT EDİLEMEMESİ:**
   ÇÖZÜM: Bu tür problemleri önlemek için nesnelerin hızlarını sürekli takip edebilmek ve yönlerini de dikkate alarak hız tahminlerini daha doğru yapabilmek gerekir. Nesne takip algoritmalarının hız tahminine yön ve hareket bilgisini dahil etmek faydalı olabilir. Tek bir videoda bu sorunda kurtulabiliriz ama genel videolar için optik akış(optical flow) kullanılabilir.

BİZE NELER KATTI:

Temel görüntü işleme tekniklerine hakim olduk. Bu, gri tonlama, arka plan çıkarma, hareket algılama, kontur bulma gibi temel adımları içeriyor. Nesnelerin sınıflandırılması ve takibi konularında deneyim kazandık. Nesne sınıflandırma (yayalar ve taşıtlar) gibi pratik zorluklarla karşılaştık ve bunları nasıl aşacağımızı öğrendik. **Bozuk video kesitleri**, **yanlış hız tespiti** ve **gürültü gibi sorunları** çözme konusunda önemli deneyimler kazandık. Bu tür sorunların nasıl yönetileceğini öğrenmek, gerçek dünya projelerinde karşılaşılan hataların çözülmesinde yardımcı olur. Hataları **görselleştirme** ve **debugging** yaparak, yazılım geliştirme sürecinin daha etkili ve verimli olmasını sağladık.

Kaynakça

1. **OpenCV. (n.d.). *OpenCV documentation*. Retrieved December 3, 2024, from https://docs.opencv.org/**
2. **PyImageSearch. (2019, November 6). *A beginner's guide to background subtraction in OpenCV*. Retrieved December 3, 2024, from https://pyimagesearch.com**
3. **TensorFlow. (n.d.). *TensorFlow object detection*. Retrieved December 3, 2024, from https://www.tensorflow.org/official_models**
4. **GeeksforGeeks. (2021, January 5). *Python OpenCV tutorial: Object tracking*. Retrieved December 3, 2024, from https://www.geeksforgeeks.org/python-opencv-tutorial-object-tracking/**
5. **Stack Overflow. (2020, May 13). *How to calculate speed of moving objects in a video with OpenCV?*. Retrieved December 3, 2024, from https://stackoverflow.com/questions/61124993/how-to-calculate-speed-of-moving-objects-in-a-video-with-opencv**