Tek Kamera Görüntülerinden YOLOv8 Tabanlı Basit IoU Takibi ve Köşegen Dinamiklerine Dayalı Sezgisel Hız Ayarlaması ile Araç Hız Tahmini Üzerine Bir Analiz

Özet
Bu rapor, tek kamera görüntülerinden araç hızlarını tahmin etmek için geliştirilmiş yeni bir bilgisayar görüsü tabanlı sistemi analiz etmektedir. Sistem, nesne tespiti için YOLOv8 modelini, özel olarak geliştirilmiş bir Kesişim üzeri Birleşim (IoU) tabanlı nesne takip mekanizmasını ve sınırlayıcı kutu köşegen dinamiklerine dayalı deneysel bir sezgisel hız ayarlama yöntemini birleştirmektedir. Bu çalışmanın temel amacı, sistemin mimarisini, algoritmik seçimlerini ve özellikle özel IoU izleyicisi ile sezgisel hız ayarlama mekanizmasının özgün yönlerini derinlemesine incelemektir. Karşılaştırmalı bir analiz, sistemin yaklaşımını yerleşik hız tespit ve nesne takip metodolojileriyle karşılaştırarak güçlü ve zayıf yönlerini ortaya koymaktadır. Sistemin ayırt edici özellikleri arasında, harici kütüphanelere olan bağımlılığı azaltan basitleştirilmiş takip mekanizması ve tam kamera kalibrasyonu gerektirmeden perspektif etkilerini dolaylı olarak ele almaya çalışan sezgisel hız düzeltmesi bulunmaktadır. Bununla birlikte, takip sisteminin karmaşık trafik senaryolarındaki sağlamlığı ve sezgisel hız ayarlamasının genelleştirilebilirliği, sistemin temel sınırlamaları olarak tanımlanmaktadır.

---

1. Giriş

1.1. Görüntü Tabanlı Araç Hız Tahmininin Bağlamı ve Önemi

Bilgisayar görüsü, Akıllı Ulaşım Sistemleri (AUS) alanında devrim yaratmakta olup, trafik izleme, hız denetimi ve otonom sürüş teknolojilerinde merkezi bir rol oynamaktadır.[1] Görüntü tabanlı sistemler, radar, LiDAR veya yola gömülü endüksiyon döngüleri gibi geleneksel sensör tabanlı yöntemlere kıyasla maliyet etkinliği, esneklik ve müdahalesiz doğası gibi önemli avantajlar sunmaktadır.[2] Bu sistemlerin yetenekleri, trafik uygulamalarının ötesine geçerek spor analitiği ve hayvan davranışlarının izlenmesi gibi çeşitli alanlara da uzanmaktadır.[2] Mevcut CCTV altyapılarının yaygınlaşması [4], özellikle kalibre edilmemiş veya otomatik kalibrasyon yapabilen hız tahmin yöntemlerine olan talebi artırmaktadır. İncelenen sistemin, minimum kalibrasyonla (varsayılan araç genişliği gibi) çalışmaya yönelik tasarımı, bu pratik ihtiyaca cevap verme potansiyeli taşımaktadır. Bu tür sistemler, genellikle karmaşık kalibrasyon prosedürlerinden kaçınarak, mevcut görsel altyapıdan nicel hız verileri elde etme zorunluluğunu ele alır.

1.2. Mevcut Yaklaşımlara ve Zorluklara Genel Bakış

Görüntü tabanlı hız tahmin sistemleri genellikle nesne tespiti, nesne takibi ve gerçek dünya

mesafe eşlemesi gibi temel bileşenlerden oluşan bir işlem hattını takip eder.[2] Ancak, bu sistemlerin geliştirilmesi ve uygulanması bir dizi zorlukla karşı karşıyadır:

- **Perspektif Bozulması:** Kameradan uzaklaşan nesnelerin daha küçük görünmesi ve aynı gerçek dünya hızında olsalar bile piksellerde daha yavaş hareket etmesi, doğru hız tahmini için önemli bir engeldir.[1]
- **Kamera Kalibrasyonu:** Doğru metrik ölçümler için kameranın içsel ve dışsal parametrelerinin bilinmesi gerekliliği ve bu kalibrasyon işleminin karmaşıklığı, yaygın uygulamayı kısıtlayabilmektedir.[1]
- **Örtüşme (Occlusion):** Nesnelerin kısmen veya tamamen diğer nesneler tarafından gizlenmesi, takip sürekliliğini ve doğruluğunu olumsuz etkileyebilir.[2]
- **Değişken Çevresel Koşullar:** Aydınlatma değişiklikleri, hava koşulları (yağmur, sis, kar) ve gölgeler, görüntü kalitesini ve dolayısıyla sistem performansını etkileyebilir.[2]
- **Hesaplama Maliyeti:** Özellikle karmaşık modeller veya yüksek çözünürlüklü videolar için gerçek zamanlı işleme gereksinimleri, önemli hesaplama kaynakları gerektirebilir.[2]
- **Nesne Çeşitliliği ve Görünüm Değişiklikleri:** Farklı araç türlerini ve hareket veya aydınlatma nedeniyle nesne görünümlerindeki değişiklikleri etkili bir şekilde ele almak zordur.[10]

Nesne tespit algoritmalarının, özellikle YOLO serisi gibi [15], verimlilik ve doğruluk açısından sürekli gelişimi, görüntü tabanlı hız tahmin sistemlerinin daha uygulanabilir ve sağlam hale gelmesini doğrudan sağlamaktadır. Tespit aşaması, tüm işlem hattının temelini oluşturduğundan, bu alandaki ilerlemeler genel sistem performansını önemli ölçüde etkilemektedir.

1.3. Analiz Edilen Sisteme ve Amaçlarına Giriş

Bu raporun konusu, kullanıcının sağladığı Python tabanlı bir araç hız tahmin sistemidir. Sistemin temel amacı, tek bir sabit kameradan alınan video görüntülerinden araç hızlarını tahmin etmektir. Bu amaca ulaşmak için sistem, nesne tespiti için YOLOv8 modelini, özel olarak geliştirilmiş bir Kesişim üzeri Birleşim (IoU) tabanlı takip algoritmasını ve sınırlayıcı kutu köşegenlerindeki değişim oranına dayanan yeni bir sezgisel hız ayarlama mekanizmasını kullanmaktadır.

1.4. Raporun Katkısı ve Yapısı

Bu rapor, söz konusu sistemin tasarımının, uygulamasının ve performans özelliklerinin derinlemesine bir analizini sunmaktadır. Rapor, sistemin benzersiz özelliklerine, özellikle basitleştirilmiş takip mekanizmasına ve sezgisel hız iyileştirme yaklaşımına odaklanmaktadır. Takip eden bölümlerde, öncelikle ilgili temel kavramlar ve literatürdeki çalışmalar incelenecek, ardından sistemin tasarımı ve uygulaması ayrıntılı olarak analiz edilecek, sonrasında diğer yöntemlerle karşılaştırmalı bir değerlendirme yapılacak ve son olarak genel sonuçlar ve gelecekteki araştırma yönelimleri tartışılacaktır.

---

2. Temel Kavramlar ve İlgili Çalışmalar

2.1. Nesne Tespiti: YOLOv8 Çerçevesi

2.1.1. YOLOv8 Mimarisi (Omurga, Boyun, Kafa)

"You Only Look Once" (YOLO) prensibi, nesne tespitini tek bir ağ geçişiyle gerçekleştirerek hız

ve verimlilik sağlamayı amaçlar.[16] YOLOv8 mimarisi, bu prensibi temel alarak üç ana bileşenden oluşur:

- **Omurga (Backbone):** Genellikle CSPDarknet53 gibi bir mimariye dayanan bu bölüm, giriş görüntüsünden özellik haritaları çıkarmakla görevlidir. Hiyerarşik özelliklerin yakalanmasında kritik bir rol oynar; alt katmanlar kenarlar ve dokular gibi düşük seviyeli özellikleri, üst katmanlar ise daha karmaşık nesne parçalarını ve örüntülerini öğrenir.[16]
- **Boyun (Neck):** Genellikle PANet (Path Aggregation Network) benzeri bir yapıya sahip olan boyun kısmı, omurganın farklı aşamalarından gelen özellik haritalarını birleştirir. Bu özellik birleştirme, farklı ölçeklerdeki nesnelerin (örneğin, hem yakın hem de uzaktaki araçların) etkili bir şekilde tespit edilebilmesi için hayati öneme sahiptir.[17]
- **Kafa (Head):** Bu bölüm, birleştirilmiş özellik haritalarını kullanarak nihai tespitleri üretir. Her bir tespit için sınırlayıcı kutu koordinatlarını, nesne sınıfını (örneğin, araba, otobüs) ve tespitin güven skorunu tahmin eder.[16]

YOLOv8, eğitim sırasında kullanılan gelişmiş veri artırma teknikleri ve bazı versiyonlarında potansiyel olarak çapasız (anchor-free) tespit yaklaşımları gibi mimari yenilikler sunar.[16]

2.1.2. YOLOv8 Varyantlarının Performans Özellikleri

YOLOv8, farklı hesaplama kaynakları ve performans gereksinimlerine hitap etmek üzere çeşitli varyantlarda (örneğin, 'n' - nano, 's' - small, 'm' - medium, 'l' - large, 'x' - extra large) sunulmaktadır. Bu varyantlar, doğruluk (genellikle Ortalama Ortalama Kesinlik - mAP ile ölçülür), hız (gecikme süresi veya saniyedeki kare sayısı - FPS) ve model boyutu (parametre sayısı, FLOPs) arasında bir denge sunar.[15] Genel olarak, daha büyük modeller (örneğin, YOLOv8x) daha yüksek doğruluk sağlarken, daha fazla hesaplama kaynağı tüketir ve daha yavaştır. Daha küçük modeller (örneğin, YOLOv8n) ise daha hızlıdır ve daha az kaynak gerektirir ancak doğrulukları genellikle daha düşüktür.

Aşağıdaki tablo, farklı YOLOv8 varyantlarının COCO veri kümesi üzerindeki performans metriklerini özetlemektedir [22]:

Tablo 1: YOLOv8 Model Varyantlarının Performans Karşılaştırması

Model	Boyut (piksel)	mAP^{val 50-95}	Hız CPU ONNX (ms)	Hız T4 TensorRT10 (ms)	Parametre (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	1.47	3.2	8.7
YOLOv8s	640	44.9	128.4	2.66	11.2	28.6
YOLOv8m	640	50.2	234.7	5.86	25.9	78.9
YOLOv8l	640	52.9	375.2	9.06	43.7	165.2
YOLOv8x	640	53.9	479.1	14.37	68.2	257.8

İncelenen Python kodunda YOLO_MODEL = 'yolov8x.pt' olarak belirtilmiştir. Bu seçim, geliştiricinin tespit doğruluğuna öncelik verdiğini, potansiyel olarak daha yüksek hesaplama maliyeti ve daha düşük işlem hızı pahasına bu kararı aldığını göstermektedir. YOLOv8x, Tablo 1'de görüldüğü gibi en yüksek mAP değerine sahip olmakla birlikte en yavaş ve en büyük

modeldir. Bu, sistemin genel performansını, özellikle gerçek zamanlılık gereksinimlerini etkileyebilecek önemli bir faktördür.

2.2. Bilgisayar Görüsünde Nesne Takibi Prensipleri

Nesne takibi, video kareleri boyunca ilgi çekici nesneleri tanımlama ve kimliklerini koruyarak hareketlerini izleme sürecidir.[10] İncelenen kodun da benimsediği "tespit ile takip" (tracking-by-detection) paradigması, bu alandaki baskın yaklaşımdır. Bu paradigmada, her karede önce nesneler bir nesne tespit algoritması (örneğin YOLO) ile tespit edilir, ardından bu tespitler mevcut takip edilen nesnelerle ilişkilendirilir.[11]

2.2.1. IoU Tabanlı Takip

Kesişim üzeri Birleşim (IoU), iki sınırlayıcı kutu arasındaki mekansal örtüşmeyi ölçen temel bir metriktir.[27] IoU tabanlı takipte, mevcut karedeki tespitler, bir IoU eşik değerine göre önceki karelerdeki mevcut izlerle eşleştirilir.[11] Basit IoU takibinin temel sınırlamaları şunlardır:

- Nesnelerin birbirine yakın olduğu veya örtüştüğü durumlarda kimlik değişimlerine (ID switch) yatkındır.[11]
- Ek ipuçları olmadan uzun süreli örtüşmelerden sonra nesneleri yeniden tanımlayamaz.[11]
- Tespit kalitesine duyarlıdır; sınırlayıcı kutulardaki titreşimler izlerin kopmasına neden olabilir.

2.2.2. Kalman Filtresi ve Hareket Modeli Tabanlı Takip (örneğin, SORT)

Kalman filtresi, bir nesnenin mevcut durumuna ve bir hareket modeline dayanarak bir sonraki karedeki durumunu (konum, hız) tahmin etmek için kullanılan yinelemeli bir algoritmadır.[10] SORT (Simple Online and Realtime Tracking), YOLO gibi bir tespit ediciyi hareket tahmini için Kalman filtreleri ve IoU tabanlı atama için Macar algoritması ile birleştirir.[10] SORT verimlidir ancak yalnızca hareket ve IoU'ya dayandığı için kimlik değişimleri ve örtüşmeler konusunda sınırlamaları vardır.[11]

2.2.3. Takipte Görünüm Tabanlı Yeniden Tanımlama (örneğin, DeepSORT)

Görünüm özellikleri (derin öğrenme gömülmeleri), özellikle örtüşmelerden sonra nesneleri yeniden tanımlamak için kullanılır.[11] DeepSORT, eşleştirme sürecine derin bir ilişkilendirme metriği (görünüm özellikleri arasındaki kosinüs mesafesi) ekleyerek SORT'u geliştirir, bu da onu örtüşmelere karşı daha sağlam hale getirir ve kimlik değişimlerini azaltır.[11]

2.2.4. Sağlamlık için Gelişmiş Takip Stratejileri (örneğin, ByteTrack)

ByteTrack, takip için hem yüksek güvenli hem de düşük güvenli tespitleri kullanma yaklaşımını benimser.[25] İki aşamalı bir eşleştirme süreci kullanır: yüksek skorlu tespitler izlerle (genellikle IoU veya Kalman tabanlı), ardından örtüşme sırasında nesneleri kurtarmak için düşük skorlu tespitler kalan izlerle (IoU kullanarak) eşleştirilir.[25] Özellikle örtüşmeli senaryolarda MOTA (Multiple Object Tracking Accuracy) ve IDF1 (Identification F1 score) skorlarını iyileştirmede etkilidir.[25]

Takip algoritmalarının evrimi (IoU -> SORT -> DeepSORT -> ByteTrack), örtüşmelere ve kimlik değişimlerine karşı daha iyi sağlamlık elde etmek için artan karmaşıklık ve hesaplama maliyeti yönünde net bir eğilim göstermektedir. Bu ilerleme, takip probleminin doğasında var olan zorluğu vurgulamaktadır. Her bir adım, bir öncekinin bir sınırlamasını ele almaktadır. Bu nedenle, "herkese uyan tek bir" izleyici yoktur; seçim, uygulamanın hatalara toleransı ile

hesaplama bütçesine bağlıdır. İncelenen kodda özel, basit bir IoU izleyicisinin seçilmesi, geliştiricinin muhtemelen daha karmaşık izleyicilerin sunduğu gelişmiş sağlamlıktan bilinçli olarak feragat ederek basitliğe ve düşük bağımlılığa öncelik verdiğini göstermektedir.

2.3. Araç Hız Tahmin Metodolojileri

2.3.1. Homografi Kullanarak Perspektif Düzeltme

Perspektif bozulması, uzaktaki nesnelerin daha küçük görünmesine ve aynı gerçek dünya hızında olsalar bile piksellerde daha az hareket etmesine neden olur.[1] Homografi dönüşümü, görüntü noktalarını kuşbakışı bir görünüme eşleyerek sabit bir piksel başına metre ölçeği oluşturan bir yöntemdir.[1] Homografi matrisi, manuel nokta eşleştirmesi, kaybolma noktası tespiti veya derin öğrenme tabanlı yaklaşımlarla elde edilebilir.[1] Bu yöntem tipik olarak bir tür kalibrasyon veya bilinen gerçek dünya ölçümleri gerektirir.[7]

2.3.2. Bilinen Nesne Boyutları ve Piksel Ölçeklemesi ile Hız Tahmini

Bu yaklaşım, tespit edilen nesnenin bilinen bir gerçek dünya boyutunu (örneğin, ortalama araç genişliği/uzunluğu) varsaymaya dayanır.[5] Bu bilinen boyut, nesnenin görüntüdeki piksel genişliği/uzunluğu ile birlikte kullanılarak nesnenin konumunda bir piksel-metre ölçeklendirme faktörü hesaplanır. Zorluklar arasında gerçek nesne boyutlarındaki değişkenlik, perspektifin etkisi (ölçek faktörü mesafeyle değişir) ve kamera açısı yer alır.[2] İncelenen kod, ASSUMED_CAR_WIDTH_METERS parametresini bu amaçla kullanır.

2.3.3. Optik Akış Teknikleri

Optik akış, ardışık kareler arasındaki nesnelerin görünür hareket örüntüsüdür.[14] Lucas-Kanade (seyrek) ve Farneback (yoğun) gibi yaygın algoritmalar mevcuttur.[14] Avantajları (akış için açık tespit gerekmez) ve dezavantajları (yoğun akış için hesaplama açısından yoğun, aydınlatma değişikliklerine duyarlı, gerçek dünya hızı için kalibrasyon gerektirir) vardır.[14]

2.3.4. Kalibre Edilmemiş ve Sezgisel Yaklaşımlar

Açık kamera kalibrasyonu olmadan hızı tahmin etmeye çalışan, genellikle varsayımlara veya öğrenme tabanlı yaklaşımlara dayanan yöntemler de mevcuttur.[4] Bilinen gerçek dünya mesafelerine sahip "sanal çizgiler" veya ROI'lerin (Region of Interest - İlgi Alanı) kullanımı, daha basit bir kalibrasyon yöntemi olarak kabul edilebilir.[9] İncelenen kodun sınırlayıcı kutu köşegen değişikliklerine dayalı sezgisel yaklaşımı, tam kalibrasyon olmadan perspektifi dolaylı olarak ele alma girişimi olarak bu kategoriye girer.

Birçok gelişmiş hız tahmin yöntemi, metrik doğruluk için geometrik prensiplerle (homografi veya kaybolma noktalarından 3B sınırlayıcı kutu rekonstrüksiyonu gibi) tespit/özellik çıkarımı için derin öğrenmeyi birleştiren hibrit yaklaşımlara doğru kaymaktadır.[1] Derin öğrenme, özellik temsili ve örüntü tanımada üstünken, geometrik yöntemler gerçek dünya ölçümlerine bir köprü sağlar. İncelenen kod, güçlü bir derin öğrenme tespit edicisi (YOLOv8) kullanır, ancak nispeten basit bir geometrik varsayıma (sabit araç genişliği) ve bir sezgisel yönteme dayanır. Bu, onu bu hibrit SOTA (State-of-the-Art - En Son Teknoloji) yöntemlere kıyasla daha hafif bir yaklaşım olarak konumlandırır, ancak karmaşık senaryolarda potansiyel olarak daha az doğrudur. Köşegen değişikliğine ilişkin sezgisel yöntem, açık kalibrasyon olmaksızın veriye dayalı bir geometrik ipucu ekleme girişimidir.

Kamera perspektifini ele almak için tam 3B kalibrasyon ve homografiden bilinen nesne boyutlarına dayanan yöntemlere ve daha da ileri giderek tamamen kalibre edilmemiş veya

sezgisel yöntemlere kadar bir dizi yaklaşım bulunmaktadır. İncelenen kod, ASSUMED_CAR_WIDTH_METERS (bilinen nesne boyutu varsayımı) kullanarak ve ardından sınırlayıcı kutu köşegen değişikliklerine dayalı bir sezgisel yöntem uygulayarak ilginç bir orta noktada yer almaktadır. Bu köşegen değişikliği sezgiseli, açık homografi veya tam kalibrasyon yapmadan perspektif etkilerini dinamik olarak ayarlamaya veya hız tahminini iyileştirmeye çalışan, dolaylı bir telafi mekanizmasıdır. Bu, temel bir varsayımın üzerine veriye dayalı bir düzeltmedir ve sistemin en benzersiz yönlerinden biridir.

---

3. Sistem Tasarımı ve Uygulama Analizi (Kullanıcının Koduna Göre)

3.1. Uygulanan Hız Tahmin Edicisinin Mimari Genel Görünümü

İncelenen sistem, bir video akışından araç hızlarını tahmin etmek için sıralı bir işlem hattı izler. Bu hat şu adımlardan oluşur:

1. **Video Girişi:** Hedef video dosyası (VIDEO_PATH) okunur.
2. **Nesne Tespiti (YOLOv8):** Her karede, önceden eğitilmiş YOLOv8 modeli kullanılarak araçlar (veya hedeflenen diğer sınıflar) tespit edilir.
3. **Nesne Takibi (Özel IoU):** Tespit edilen nesneler, özel bir IoU tabanlı algoritma kullanılarak kareler arasında izlenir ve her bir izlenen araca benzersiz bir kimlik atanır.
4. **Hız Hesaplama (İlk + Sezgisel Ayarlama):** İzlenen her araç için, piksel cinsinden yer değiştirmesi ve varsayılan bir araç genişliği kullanılarak bir ilk hız tahmin edilir. Ardından, sınırlayıcı kutu köşegenindeki değişim oranına dayalı deneysel bir sezgisel yöntemle bu ilk hız ayarlanır.
5. **Görselleştirme:** Tespit edilen araçlar, sınırlayıcı kutuları ve tahmin edilen hızları ile birlikte orijinal kare üzerine çizilerek görüntülenir.

3.2. Nesne Tespit Alt Sistemi (YOLOv8 Entegrasyonu)

Sistem, nesne tespiti için ultralytics kütüphanesini ve önceden eğitilmiş yolov8x.pt modelini (YOLO_MODEL) kullanır. Bu, YOLOv8 ailesinin en büyük ve genellikle en doğru varyantıdır, bu da tespit doğruluğuna öncelik verildiğini gösterir. TARGET_CLASSES_IDX = parametresi, sistemin COCO veri kümesindeki 'araba' (indeks 2), 'otobüs' (indeks 5) ve 'kamyon' (indeks 7) sınıflarına odaklanmasını sağlar. Model çıkarımı sırasında verbose=False ayarı, konsola gereksiz çıktıların yazılmasını engeller.

3.3. Özel IoU Tabanlı Nesne Takip Mekanizması

Sistem, harici takip kütüphanelerine (örneğin, kodda yorum satırı haline getirilmiş supervision) dayanmak yerine kendi IoU tabanlı takip mantığını uygular.

3.3.1. IoU Hesaplama ve İlişkilendirme Mantığı

calculate_iou fonksiyonu, iki sınırlayıcı kutu arasındaki standart kesişim üzeri birleşim oranını hesaplar.[27] İlişkilendirme mantığı şu şekilde işler:

- Önceki karedeki izlenen nesneler (prev_tracks) ile mevcut karedeki ham tespitler (current_detections_raw) karşılaştırılır.
- Her bir önceki iz için, mevcut tespitler arasında en iyi IoU eşleşmesi aranır.
- Bir eşleşmenin geçerli sayılması için IoU değerinin IOU_THRESHOLD (kodda 0.5 olarak ayarlanmış) değerinden büyük veya eşit olması gerekir.
- Bir tespitin yalnızca bir izle eşleşmesini sağlamak için matched_current_indices kümesi kullanılır.

3.3.2. İz Yönetimi: Başlatma, Sürdürme ve Sona Erme

- **İz Başlatma:** Mevcut karede herhangi bir önceki izle eşleşmeyen yeni tespitler için yeni iz kimlikleri (new_track_id) oluşturulur ve next_track_id sayacı artırılır.
- **İz Sürdürme:** Eşleşen izlerin sınırlayıcı kutusu (bbox), zaman damgası (timestamp), köşegen uzunluğu (diag) ve sınıf kimliği (class_id) güncellenir. frames_since_last_seen (son görülmesinden bu yana geçen kare sayısı) sıfırlanır.
- **İz Sona Erme:** Bir iz, TRACK_EXPIRY_FRAMES (kodda 15 kare olarak ayarlanmış) boyunca eşleşmezse, bir sonraki kare için güncellenmiş iz listesi olan updated_prev_tracks'e dahil edilmez ve dolayısıyla örtük olarak sonlandırılır. Ayrıca, artık takip edilmeyen izlere ait geçmiş konum, zaman ve köşegen verileri (track_history_coords, track_history_times, track_history_diags) bellek yönetimi için aktif olarak silinir.

3.3.3. Takip için Veri Yapıları

- prev_tracks: Her aktif izin durumunu (sınırlayıcı kutu, zaman damgası, köşegen, son görülmesinden bu yana geçen kare sayısı, sınıf kimliği) tutan bir sözlüktür.
- track_history_coords, track_history_times, track_history_diags: Hız hesaplaması için yalnızca son iki durumu (önceki ve mevcut) depolamak üzere defaultdict(lambda: deque(maxlen=2)) yapısını kullanır. Bu, amaçlanan hesaplama için verimli bir bellek yönetimi sağlar.

Özel IoU izleyicisi hafif ve anlaşılması kolaydır. Ancak, SORT'taki Kalman filtreleri gibi hareket modellerinden [11] ve DeepSORT'taki görünüm özelliklerinden [12] vazgeçerek, örtüşmelerin, hızlı hareket eden nesnelerin (kareler arasında düşük IoU'ya neden olan) veya çok benzer görünümlü nesnelerin yakın geçtiği senaryolarda doğal olarak daha fazla hataya açık olacaktır. TRACK_EXPIRY_FRAMES parametresi, kısa süreli örtüşmeler için basit bir mekanizma sunar ancak daha uzun kaybolmaları ele alamaz. Geliştiricinin muhtemelen basitliği, uygulama hızını veya minimum bağımlılığı önceliklendirdiği anlaşılmaktadır. Bu tasarım tercihi, sistemi daha gelişmiş izleyiciler kullanan sistemlere kıyasla karmaşık trafik senaryolarında potansiyel olarak daha az sağlam kılmaktadır.

3.4. Hız Hesaplama ve Sezgisel İyileştirme Modülü

3.4.1. Piksel Yer Değiştirmesi ve Varsayılan Araç Genişliğinden İlk Hız Tahmini

Hız tahmini için referans noktası, sınırlayıcı kutunun alt-orta noktasıdır (cx_pixel = (x1 + x2) // 2, cy_pixel = y2). Ardışık iki karedeki bu referans noktaları arasındaki piksel cinsinden Öklid mesafesi (math.dist) hesaplanır. Geçen süre (elapsed_time), karelerin zaman damgaları arasındaki farktan elde edilir. Piksel hızı (speed_pixels_per_sec), piksel mesafesinin geçen süreye bölünmesiyle bulunur.

Gerçek dünya hızına geçiş için kritik bir adım, piksel başına metre ölçeklendirme faktörünün (scale_factor_m_per_pix) hesaplanmasıdır. Bu, ASSUMED_CAR_WIDTH_METERS (kodda 1.8 metre olarak varsayılmış) değerinin mevcut sınırlayıcı kutunun piksel cinsinden genişliğine (current_bbox_width_pixels) bölünmesiyle elde edilir.[31] Bu varsayım, sistemin temelini oluşturur. İlk hız (m/s cinsinden) (speed_mps_initial), piksel hızının bu ölçeklendirme faktörüyle çarpılmasıyla bulunur.

Tüm metrik hız tahmininin bu tek ASSUMED_CAR_WIDTH_METERS değerine dayanması önemlidir. Gerçek araç genişliğinin bu varsayılan değerden herhangi bir sapması, hız hatasını

doğrudan ölçekleyecektir. Sezgisel yöntem bunu modüle etmeye çalışsa da, temel ölçek sabittir. Özellikle farklı araç sınıflarını (arabalar, otobüsler, kamyonlar) hedefleyen bir sistem için bu, önemli bir potansiyel hata kaynağıdır, çünkü bu sınıfların tipik genişlikleri çok farklıdır ve kod tüm sınıflar için tek bir ASSUMED_CAR_WIDTH_METERS kullanır.

3.4.2. Sınırlayıcı Kutu Köşegen Dinamiklerine Dayalı Yeni Sezgisel Ayarlama

Sistemin en özgün kısmı, bu ilk hız tahminini iyileştirmek için kullanılan sezgisel ayarlama mekanizmasıdır. Bu mekanizma, sınırlayıcı kutunun köşegen uzunluğundaki (bbox_diag = math.sqrt(bbox_width**2 + bbox_height**2)) değişim oranına (diag_change_rate = (curr_diag - prev_diag) / elapsed_time) dayanır. Sezgisel mantık şöyledir:

- Eğer diag_change_rate > DIAG_RATE_THRESHOLD_NEAR (kodda 40 piksel/saniye) ise (nesne hızla büyüyor, daha yakın olduğu varsayılıyor), hızı azaltmak için ADJUST_FACTOR_NEAR (kodda 0.8) uygulanır.
- Eğer diag_change_rate < DIAG_RATE_THRESHOLD_FAR (kodda -15 piksel/saniye) ise (nesne küçülüyor veya yavaş büyüyor, daha uzak olduğu varsayılıyor), hızı artırmak için ADJUST_FACTOR_FAR (kodda 1.7) uygulanır.
- Diğer durumlarda (orta hızda büyüme), distance_adjustment_factor 1.0 olarak kalır.

Nihai hız (speed_mps_final), ilk hızın bu ayarlama faktörüyle çarpılmasıyla elde edilir ve ardından km/s'ye dönüştürülür. Kodda bu eşiklerin ve faktörlerin (DIAG_RATE_THRESHOLD_NEAR, DIAG_RATE_THRESHOLD_FAR, ADJUST_FACTOR_NEAR, ADJUST_FACTOR_FAR) "ÇOK DENEYSEL!" olarak etiketlenmesi ve sahneye/kameraya göre ayarlanması gerektiği vurgulanmaktadır.

Bu köşegen tabanlı hız ayarlaması "ÇOK DENEYSEL!" olarak etiketlenmiştir. Etkinliği, sınırlayıcı kutu köşegen boyutundaki değişikliklerin mesafedeki değişiklikler için güvenilir bir vekil olduğu VE bu değişikliklerin bir hız düzeltme faktörüne eşlenebileceği varsayımına dayanmaktadır. Bu güçlü bir varsayımdır. Bu sezgisel yöntem, perspektifin dolaylı, kaba bir modelini oluşturmaya çalışır. Ancak, köşegen değişikliği ile gerçek mesafe değişikliği arasındaki ilişki doğrusal değildir ve kamera parametrelerine, nesne boyutlarına ve 3B yönelimine bağlıdır. Sabit eşikler ve faktörlerin farklı sahnelerde veya hatta aynı sahnenin farklı kamera bakış açılarında iyi genelleme yapması olası değildir. Örneğin, dönen bir araç, kameraya göre ileri hızı değişmeden köşegenini önemli ölçüde değiştirebilir. Bu sezgisel yöntem, sistemin en yenilikçi ama aynı zamanda potansiyel olarak en kırılgan kısmıdır.

3.5. Ayırt Edici Özellikler ve Uygulama Seçimleri

- **İzleyicinin Basitliği:** Harici kütüphanelerden veya DeepSORT/ByteTrack gibi daha karmaşık algoritmalardan kaçınarak özel bir IoU izleyicisi uygulama kararı, bağımlılıkları ve karmaşıklığı azaltır.
- **Sezgisel Perspektif Telafisi:** Köşegen tabanlı ayarlama, açık kalibrasyon veya homografi olmadan perspektif etkilerini telafi etmeye yönelik benzersiz bir girişimdir ve sistemin temel yeniliğini oluşturur.
- **Doğrudan Parametrelendirme:** IOU_THRESHOLD, TRACK_EXPIRY_FRAMES, ASSUMED_CAR_WIDTH_METERS ve sezgisel eşik/faktörler gibi birçok anahtar parametrenin sabit kodlanmış olması, mevcut deneysel/videoya özgü doğasını vurgular.
- **Belirli Araç Sınıflarına Odaklanma:** Arabalar, otobüsler ve kamyonlar hedeflenmektedir.

- **deque(maxlen=2) Kullanımı:** Hız hesaplaması için yalnızca gerekli önceki ve mevcut durumları verimli bir şekilde depolayarak bu özel görev için belleği iyi yönetir.

---

4. Karşılaştırmalı Analiz ve Eleştirel Değerlendirme

4.1. Takip Performansı: Özel IoU İzleyici vs. Yerleşik Algoritmalar (SORT, DeepSORT, ByteTrack)

İncelenen sistemdeki özel IoU izleyicisi, basitliği ve düşük hesaplama yükü ile öne çıksa da, daha gelişmiş takip algoritmalarıyla karşılaştırıldığında bazı önemli sınırlamalara sahiptir.

Tablo 2: Nesne Takip Algoritmalarının Niteliksel Karşılaştırması

Algoritma	Birincil İlişkilendirme İpucu	Örtüşme Ele Alma Yeteneği	Kimlik Değişimi Sağlamlığı	Hesaplama Karmaşıklığı	Harici Özelliklere Bağımlılık
Kullanıcının IoU'su	Yalnızca IoU	Düşük	Düşük	Düşük	Yok
SORT	IoU + Kalman Hareketi	Orta	Orta	Orta	Yok
DeepSORT	IoU + Hareket + Görünüm	Yüksek	Yüksek	Yüksek	Derin Özellikler
ByteTrack	Yüksek/Düşük Güven IoU + (isteğe bağlı) Görünüm	Çok Yüksek	Çok Yüksek	Yüksek	Yok/Derin Özellikler

11

Özel izleyicinin avantajları arasında basitlik, OpenCV, NumPy ve YOLO dışında harici bağımlılıkların olmaması ve nesne sayısı az olduğunda potansiyel olarak daha hızlı olması sayılabilir. Ancak dezavantajları daha belirgindir:

- Kalabalık sahnelerde veya örtüşmelerle karşılaşıldığında SORT/DeepSORT/ByteTrack'e kıyasla muhtemelen daha yüksek kimlik değişimi oranı.[11]
- Uzun süreli örtüşmelerin zayıf bir şekilde ele alınması (nesneler kaybolacak ve yeni olarak yeniden tespit edilecektir).
- Tespit edici gürültüsüne/titremesine karşı hassasiyet.

4.2. Hız Tahmin Doğruluğu: Önerilen Sezgisel Yöntem vs. Geleneksel Teknikler (Homografi, Kalibre Edilmiş Ölçekleme)

Sistemin sezgisel hız ayarlama mekanizması, kalibrasyonsuz senaryolar için yenilikçi bir yaklaşım sunsa da, yerleşik yöntemlerle karşılaştırıldığında doğruluk ve genelleştirilebilirlik açısından değerlendirilmelidir.

Tablo 3: Araç Hız Tahmin Metodolojilerine Genel Bakış

Yöntem	Kalibrasyon Gereksinimi	Perspektif Ele Alma	Tipik Avantajlar	Tipik Dezavantajlar
Kullanıcının	Minimal/Varsayılan	Sezgisel	Basitlik,	Sahneye bağımlı,

Sezgisel Yaklaşımı			kalibrasyonsuz adaptasyon denemesi	deneysel, varsayımlara dayalı
Homografi Tabanlı	Kısmi/Tam	Geometrik Dönüşüm	Yüksek doğruluk (iyi kalibrasyonla)	Kalibrasyon çabası, bilinen noktalar gerektirir
Bilinen Nesne Boyutu (Sabit Ölçek)	Varsayılan Nesne Boyutu	Temel Ölçekleme	Basit uygulama	Nesne boyutu değişkenliği, perspektife duyarlı
Optik Akış Tabanlı	İsteğe Bağlı/Tam	Akış Vektörü Analizi	Tespit gerektirmeyebilir	Hesaplama yoğunluğu, aydınlatmaya duyarlı, kalibrasyon

1

Sezgisel yöntemin analizi:

- **Potansiyel Güçlü Yönleri:** Açık kalibrasyon olmadan perspektife uyum sağlamaya çalışır; hesaplama açısından ucuzdur.
- **Potansiyel Zayıf Yönleri:**
    - Son derece deneyseldir: DIAG_RATE_THRESHOLD_NEAR/FAR ve ADJUST_FACTOR_NEAR/FAR videoya özgüdür ve dikkatli bir şekilde ayarlanması gerekir. Yeni videolara/sahnelere genelleştirilmesi sorgulanabilir.
    - Korelasyon varsayımı: Köşegen değişim oranının doğrudan ve basit bir şekilde mesafe ile ilişkili olduğunu ve doğrusal bir faktörle düzeltilebileceğini varsayar. Bu, karmaşık hareketler (örneğin, dönen araçlar, kameraya çapraz olarak şerit değiştiren araçlar) için geçerli olmayabilir.
    - Sınırlayıcı kutu kararlılığına duyarlılık: Sınırlayıcı kutu yüksekliği/genişliğindeki küçük titreşimler, gürültülü bir diag_change_rate değerine ve düzensiz hız ayarlamalarına yol açabilir.
    - Gerçek 3B geometriyi ve kamera içsel/dışsal parametrelerini göz ardı eder.

Homografi tabanlı yöntemler, daha sağlam perspektif düzeltmesi sunar ancak kalibrasyon noktaları gerektirir.[1] Basit bilinen nesne boyutu ölçeklemesi (sezgisel olmadan), kullanıcının sezgisel yönteminin olmaya çalıştığından daha az derinlik değişimlerine uyarlanabilir.[8] Kullanıcının yöntemi bununla başlar ve onu iyileştirmeye çalışır.

4.3. Uygulanan Sistemin Güçlü ve Avantajlı Yönleri

- **Bağımsız ve Hafif:** Takip ve hız mantığı için minimum harici bağımlılık.
- **Güçlü Tespit Ediciden Yararlanma:** Doğru nesne tespitleri için güçlü bir temel sağlayan YOLOv8 (özellikle yolov8x.pt) kullanır.[16]
- **Yeni Sezgisel Yöntem:** Köşegen tabanlı ayarlama, deneysel olsa da, kalibre edilmemiş senaryolar için yenilikçi bir fikirdir.

- **Anlaşılır Uygulama:** Kodun temel mantığı nispeten kolay takip edilebilir.

4.4. Sınırlamalar, Potansiyel Yanlışlıklar ve Gelecekteki İyileştirme Alanları

- **Takip Sağlamlığı:** Daha önce tartışıldığı gibi, IoU izleyicisi karmaşık sahnelerde önemli bir sınırlamadır.
- **Hız Doğruluğu:**
    - ASSUMED_CAR_WIDTH_METERS değerine ve tüm araç türleri için tek bir değere bağımlılık.
    - Sezgisel parametrelerin deneysel doğası ve sahneye özgülüğü.
    - Gerçek kamera kalibrasyonunun olmaması.
    - Kod, video zamanı için current_time_sec = frame_count / fps kullanıldığını belirtir, ancak current_time_sec = time.time() satırını da yorum satırı haline getirmiştir. Video FPS'sini kullanmak, sabit FPS ve zamanlamayı etkileyen işlem gecikmesi olmadığını varsayar. time.time() kullanmak işlem gecikmesini hesaba katacaktır ancak FPS çılgınca dalgalanırsa gürültülü olabilir. Video FPS'si güvenilirse mevcut seçim genellikle tutarlı hız tahmini için daha iyidir.
- **Genelleştirilebilirlik:** Sistem, özellikle sezgisel yöntem, farklı kamera açılarına, yüksekliklerine veya yol geometrilerine yeniden ayarlanmadan iyi bir şekilde genelleştirilemeyebilir.

Kullanıcının sistemi, karmaşık kalibrasyon, daha fazla bağımlılık veya daha ağır modeller gerektirebilecek SOTA doğruluğuna ulaşmak yerine, basit dağıtımla "yeterince iyi" bir tahminin tercih edildiği senaryolar için tasarlanmış gibi görünmektedir. Bu geçerli bir mühendislik ödünleşimidir.

Tüm sistem (takip ve hız tahmini, özellikle sezgisel yöntem), YOLOv8'den gelen sınırlayıcı kutuların kalitesine ve kararlılığına büyük ölçüde bağlıdır. Kutu boyutlarındaki, özellikle köşegeni etkileyen genişlik ve yükseklikteki küçük titreşimler veya tutarsızlıklar, sezgisel yöntemle büyütülebilir.[8] Köşegenin *değişim oranına* bakan sezgisel yöntem, köşegen ölçümündeki gürültüye özellikle duyarlı olacaktır. Genişlik veya yükseklikteki küçük bir titreme, büyük, hatalı bir diag_change_rate değerine yol açabilir. Bu, yolov8x.pt doğruluğu için seçilmiş olsa da, çıktı sınırlayıcı kutularındaki herhangi bir kalıntı kararsızlığın yeni sezgisel yöntemi doğrudan etkileyeceğini göstermektedir. Bu, sezgisel yöntemin daha kararlı olması için köşegen oranı hesaplanmadan önce sınırlayıcı kutu boyutlarının zamansal olarak düzeltilmesinin veya diag_change_rate değerinin kendisinin düzeltilmesinin gerekli bir ön işleme adımı olabileceğini düşündürmektedir.

DIAG_RATE_THRESHOLD_NEAR ve DIAG_RATE_THRESHOLD_FAR eşikleri, sınırlayıcı kutu köşegeninin ne kadar hızlı değiştiğine bağlı olarak alanı "yakın", "uzak" ve "orta" bölgelere ayırmaya çalışır. Bu, sürekli bir derinlik etkisinin basitleştirilmesidir. Perspektif etkisi süreklidir; bir nesnenin görünür boyutu mesafeyle doğrusal olmayan bir şekilde değişir. Ayrık eşikler, ayarlama faktöründe ani değişiklikler yaratır. Bir eşiğe yakın gezinen bir nesne, köşegen değişim oranındaki küçük dalgalanmalar nedeniyle hız tahmininde önemli bir sıçrama görebilir. Ayrıca, DIAG_RATE_THRESHOLD_FAR ve DIAG_RATE_THRESHOLD_NEAR arasındaki "nötr" bölge, ASSUMED_CAR_WIDTH_METERS'e dayalı ilk hız tahmininin bu ara mesafe için doğru olduğunu varsayar ki bu da kendi içinde bir yaklaşımdır. Sezgisel yöntem zekice olsa da, hız tahmininde kendi doğrusalsızlık veya ani değişiklik biçimini getirebilir.

İyileştirme Önerileri:

- Daha sağlam bir izleyici entegre etmek (örneğin, SORT veya bağımlılıklar kabul edilebilirse ByteTrack'e geçme seçeneği).
- Sezgisel parametreleri öğrenmek veya yarı otomatik kalibrasyon yöntemlerini araştırmak.
- Sınıf kimliği güvenilir bir şekilde belirlenirse, sınıf başına ASSUMED_WIDTH_METERS uygulamak.
- BrnoCompSpeed [15] gibi referans veri kümelerine veya bir hız tabancası [1] kullanılarak elde edilen gerçek dünya verilerine karşı titiz nicel değerlendirme yapmak.
- Sezgisel yöntemin farklı araç türlerine ve bunların 3B şekillerine duyarlılığını araştırmak (köşegen değişimi bir sedan ile uzun bir kamyon için farklı davranabilir).

---

5. Sonuç ve Gelecek Yönelimler

5.1. Temel Bulguların Özeti

Bu rapor, YOLOv8 nesne tespiti, özel bir IoU tabanlı takip mekanizması ve sınırlayıcı kutu köşegen dinamiklerine dayalı deneysel bir sezgisel hız ayarlama modülünü içeren bir araç hız tahmin sistemini analiz etmiştir. Sistemin temel güçlü yönleri arasında, takip ve hız mantığı için minimum harici bağımlılığa sahip, kendi kendine yeten ve hafif yapısı, güçlü bir tespit edici olan YOLOv8'den yararlanması ve kalibre edilmemiş senaryolar için yenilikçi bir fikir olan köşegen tabanlı sezgisel ayarlama yer almaktadır. Bununla birlikte, sistemin önemli zayıf yönleri arasında, özellikle karmaşık trafik koşullarında özel IoU izleyicisinin sağlamlık eksikliği, hız tahmininin doğruluğunun büyük ölçüde tek bir ASSUMED_CAR_WIDTH_METERS değerine ve sahneye özgü, deneysel sezgisel parametrelere dayanması ve genel olarak farklı kamera kurulumlarına veya yol geometrilerine genelleştirilebilirliğinin düşük olması bulunmaktadır.

5.2. Sistemin Pratikliği ve Yenilikçiliğinin Değerlendirilmesi

Sistem, yapılan ödünleşmeler göz önüne alındığında pratik uygulama potansiyeline sahiptir. Özellikle, kamera görüntülerinin nispeten tutarlı olduğu ve yüksek hassasiyetin birincil öncelik olmadığı, ancak hızlı dağıtımın ve minimum kurulumun önemli olduğu senaryolar için uygun olabilir. Köşegen tabanlı sezgisel ayarlama, hafif perspektif telafisi için ilginç bir araştırma yönü olarak yenilikçi bir değer taşımaktadır, ancak mevcut haliyle önemli ölçüde deneyseldir. Sistemin genel yaklaşımı, "yeterince iyi" bir çözümün, karmaşık kalibrasyon veya ağır modeller gerektirebilecek en son teknoloji doğruluğundan daha çok tercih edildiği durumlar için bir mühendislik tercihi olarak görülebilir.

5.3. Daha Fazla Araştırma ve Geliştirme için Öneriler

Sistemin yeteneklerini ve güvenilirliğini artırmak için çeşitli araştırma ve geliştirme yolları önerilmektedir:

- **Nicel Değerlendirme:** Sistemin doğruluğunu objektif olarak ölçmek ve sezgisel yöntemin etkinliğini farklı koşullar altında karşılaştırmak için standartlaştırılmış referans veri kümeleri (örneğin, BrnoCompSpeed [15]) üzerinde veya hız tabancası gibi yerinde doğrulama yöntemleriyle [1] kapsamlı testler yapılması kritik öneme sahiptir.
- **İzleyici Geliştirme:** Takip sağlamlığını artırmak için, basit bir Kalman filtresi gibi hafif hareket modellerinin entegrasyonu veya mevcut IoU eşiklerinin uyarlanabilir hale getirilmesi araştırılabilir. Daha karmaşık ancak daha sağlam izleyicilere (örneğin,

ByteTrack) geçiş, bağımlılık ve hesaplama maliyeti dengesi gözetilerek değerlendirilebilir.

- **Sezgisel Yöntemin İyileştirilmesi:**
  - Sezgisel parametrelerin (DIAG_RATE_THRESHOLD_NEAR/FAR, ADJUST_FACTOR_NEAR/FAR) veriden öğrenilmesi veya sahne özelliklerine göre otomatik olarak ayarlanması için makine öğrenmesi teknikleri araştırılabilir.
  - Sınırlayıcı kutu alanı değişim oranı gibi alternatif veya tamamlayıcı ipuçlarının sezgisel yönteme dahil edilmesi değerlendirilebilir.
  - Farklı araç sınıfları (araba, otobüs, kamyon) için ayrı ASSUMED_WIDTH_METERS değerleri veya sınıf başına özel sezgisel parametreler kullanılması, doğruluğu artırabilir.
- **Kalibrasyon Keşfi:** Tam kalibrasyon karmaşık olsa da, bir karede yoldaki birkaç bilinen uzunluğun işaretlenmesi gibi basit yarı otomatik kalibrasyon yöntemleri bile temel scale_factor_m_per_pix değerini önemli ölçüde iyileştirebilir ve genel hız tahmin doğruluğuna katkıda bulunabilir.
- **Sınırlayıcı Kutu Gürültüsüne Karşı Sağlamlık:** Sezgisel yöntemin kararlılığını artırmak için, sınırlayıcı kutu boyutlarına veya hesaplanan köşegen değişim oranına, sezgisel mantık uygulanmadan önce zamansal bir düzeltme (smoothing) filtresi uygulanması önerilir. Bu, tespit ediciden kaynaklanan anlık titreşimlerin hız tahminleri üzerindeki olumsuz etkisini azaltabilir.

Bu öneriler, incelenen sistemin mevcut sınırlamalarını ele almayı ve onu daha doğru, sağlam ve pratik olarak uygulanabilir bir araç hız tahmin çözümüne dönüştürmeyi amaçlamaktadır.

---

6. Kaynakça

Bu raporun hazırlanmasında aşağıdaki kaynaklardan yararlanılmıştır:

- [16] Folio3. (t.y.). *What is YOLOV8? A Step-By-Step Guide*. Folio3 AI Blog. https://www.folio3.ai/blog/what-is-yolov8-architecture/
- [17] YOLOv8.org. (t.y.). *YOLOv8 Architecture Explained*. https://yolov8.org/yolov8-architecture-explained/
- [34] YOLOv8.org. (t.y.). *YOLOv8 Documentation*. https://yolov8.org/yolov8-documentation/
- [19] YOLOv8.org. (t.y.). *YOLOv8 Official Website*. https://yolov8.org/
- [27] Ultralytics. (t.y.). *Intersection over Union (IoU)*. Ultralytics Glossary. https://www.ultralytics.com/glossary/intersection-over-union-iou
- [28] Encord. (t.y.). *IoU Definition*. Encord Glossary. https://encord.com/glossary/iou-definition/
- [10] Encord. (t.y.). *Object Tracking Guide*. Encord Blog. https://encord.com/blog/object-tracking-guide/
- [11] Ikomia. (t.y.). *Deep SORT Object Tracking Guide*. Ikomia Blog. https://www.ikomia.ai/blog/deep-sort-object-tracking-guide
- [24] NextbrainTech. (t.y.). *AI Object Tracking with DeepSORT*. NextbrainTech Blog. https://www.nextbraintech.com/blog/ai-object-tracking-with-deepsort
- [12] Ahmad, S. (2023). *Object Tracking with DeepSORT*. ResearchGate. https://www.researchgate.net/publication/368646461_Object_Tracking_with_DeepSORT

- [25] Roboflow. (2024, 21 Ağustos). *What is ByteTrack in Computer Vision?*. Roboflow Blog. https://blog.roboflow.com/what-is-bytetrack-computer-vision/
- [30] Roboflow. (t.y.). *ByteTrack Model*. Roboflow. https://roboflow.com/model/bytetrack
- [13] Encord. (t.y.). *Video Object Tracking Algorithms*. Encord Blog. https://encord.com/blog/video-object-tracking-algorithms/
- [35] BroutonLab. (t.y.). *A Complete Review of the OpenCV Object Tracking Algorithms*. BroutonLab Blog. https://broutonlab.com/blog/opencv-object-tracking/
- [15] Kocur, V., et al. (2025). *Efficient Vision-based Vehicle Speed Estimation*. arXiv:2505.01203v1 [cs.CV]. https://arxiv.org/html/2505.01203v1
- [36] Reddit. (t.y.). *Speed Estimation of ANY Object in Video using Computer Vision (Vehicle Speed Detection with YOLO 11)*. r/computervision. https://www.reddit.com/r/computervision/comments/1iutg00/speed_estimation_of_any_object_in_video_using/
- [1] Fadhil, M. A., et al. (2024). *Vehicle Speed Estimation Using Consecutive Frame Approaches and Deep Image Homography for Image Rectification on Monocular Videos*. ResearchGate. https://www.researchgate.net/publication/386195390_Vehicle_Speed_Estimation_Using_Consecutive_Frame_Approaches_and_Deep_Image_Homography_for_Image_Rectification_on_Monocular_Videos
- [3] Abdulkader, O., et al. (t.y.). *Apparatus for Passively Measuring Vehicle Speed*. UCF STARS. https://stars.library.ucf.edu/cgi/viewcontent.cgi?article=1244&context=patents
- [31] Ultralytics. (t.y.). *Speed Estimation Guide*. Ultralytics Docs. https://docs.ultralytics.com/guides/speed-estimation/
- [7] Mateusz K. (t.y.). *Vehicle Speed Estimation with YOLO11 & OpenCV*. Kaggle. https://www.kaggle.com/code/mateuszk013/vehicle-speed-estimation-with-yolo11-opencv
- [14] Nurhadiyatna, A., et al. (2021). *Vehicle Speed Estimation using Optical Flow on Orthographic Projection of Traffic Video*. SIMPLE Conference. https://www.simple.ascee.org/index.php/simple/article/download/30/pdf [14]
- [14] Nurhadiyatna, A., et al. (2021). *Vehicle Speed Estimation using Optical Flow on Orthographic Projection of Traffic Video*. SIMPLE Conference. https://www.simple.ascee.org/index.php/simple/article/download/30/pdf
- [8] Roboflow. (t.y.). *How to Estimate Speed with Computer Vision*. Roboflow Blog. https://blog.roboflow.com/estimate-speed-computer-vision/
- [2] Ultralytics. (t.y.). *Ultralytics YOLOv8 for Speed Estimation in Computer Vision Projects*. Ultralytics Blog. https://www.ultralytics.com/blog/ultralytics-yolov8-for-speed-estimation-in-computer-vision-projects
- [21] ITM Web of Conferences. (2024). *Analysis of YOLO Algorithm Evolution and Performance in Object Detection: A Study of YOLOv1, YOLOv5, and YOLOv8*. ITM Web of Conferences, 60, 03008.

https://www.itm-conferences.org/articles/itmconf/abs/2025/01/itmconf_dai2024_03008/itmconf_dai2024_03008.html

- [18] Solawetz, J., & Francesco. (2024, 23 Ekim). *YOLOv8: The Premier Real-Time Object Detection Model - A Complete Guide*. Roboflow Blog. https://blog.roboflow.com/what-is-yolov8/
- [17] YOLOv8.org. (t.y.). *YOLOv8 Architecture Explained: Exploring the YOLOv8 Architecture*. https://yolov8.org/yolov8-architecture-explained/ [17]
- [20] Labellerr. (t.y.). *Understanding YOLOv8: Architecture, Applications, and Key Features*. Labellerr Blog. https://www.labellerr.com/blog/understanding-yolov8-architecture-applications-features/
- [33] Gu, Y., et al. (2025). *ClipGrader: Grading Object Detection Bounding Box Annotations with Vision-Language Models*. arXiv:2503.02897v1 [cs.CV]. https://arxiv.org/html/2503.02897v1
- [37] Zhou, Y., & Suri, S. (t.y.). *Analysis of a Bounding Box Heuristic for Object Intersection*. ResearchGate. https://www.researchgate.net/publication/2500978_Analysis_of_a_Bounding_Box_Heuristic_for_Object_Intersection
- [15] Kocur, V., et al. (2025). *Efficient Vision-based Vehicle Speed Estimation*. arXiv:2505.01203v1 [cs.CV]. https://arxiv.org/html/2505.01203v1 [15]
- [5] Kocur, V., et al. (2025). *Efficient Vision-based Vehicle Speed Estimation*. arXiv:2505.01203 [cs.CV]. https://www.arxiv.org/pdf/2505.01203
- [22] Ultralytics. (t.y.). *Model Comparison: YOLOv10 vs YOLOv8 for Object Detection*. Ultralytics Docs. https://docs.ultralytics.com/compare/yolov10-vs-yolov8/
- [38] Applied Sciences. (2024). *A Comparative Study of YOLO Models for Stamp Detection in Scanned Documents*. MDPI. https://www.mdpi.com/2076-3417/15/6/3154
- [9] Scientific Reports. (2024). *Monocular camera-based vehicle speed estimation method with variable focal length calibration*. PMC. https://pmc.ncbi.nlm.nih.gov/articles/PMC11754901/
- [15] Kocur, V., et al. (2025). *Efficient Vision-based Vehicle Speed Estimation*. arXiv:2505.01203v1 [cs.CV]. https://arxiv.org/html/2505.01203v1 [15]
- [5] Kocur, V., et al. (2025). *Efficient Vision-based Vehicle Speed Estimation*. arXiv:2505.01203 [cs.CV]. https://www.arxiv.org/pdf/2505.01203 [5]
- [32] Sensors. (2024). *Monocular Ranging Method for Forward Vehicles Based on Camera Attitude Estimation and Multi-Reference Information Fusion*. MDPI. https://www.mdpi.com/2032-6653/15/8/339
- [26] Research Square. (2024). *Uncalibrated camera approach for vehicle speed estimation using detection and tracking*. ResearchGate. https://www.researchgate.net/publication/380410353_Uncalibrated_camera_approach_for_vehicle_speed_estimation_using_detection_and_tracking
- [4] WSDOT. (t.y.). *Using Roadway Features to Augment Un-calibrated Camera Speed*

Measurements. Washington State Department of Transportation. https://www.wsdot.wa.gov/research/reports/fullreports/575.1.pdf

- [39] Kocur, V., et al. (2025). *Efficient Vision-based Vehicle Speed Estimation*. arXiv:2505.01203 [cs.CV]. https://arxiv.org/abs/2505.01203
- [6] Kocur, V., et al. (2025). *Efficient Vision-based Vehicle Speed Estimation*. arXiv:2505.01203 [cs.CV]. https://arxiv.org/pdf/2505.01203? [5]
- [13] Encord. (t.y.). *Video Object Tracking Algorithms*. Encord Blog. https://encord.com/blog/video-object-tracking-algorithms/ [13]
- [29] Zhang, Y., et al. (2024). *Hierarchical IoU Tracking*. arXiv:2406.13271v1 [cs.CV]. https://arxiv.org/html/2406.13271v1
- [11] Ikomia. (t.y.). *Deep SORT Object Tracking Guide*. Ikomia Blog. https://www.ikomia.ai/blog/deep-sort-object-tracking-guide [11]
- [12] Ahmad, S. (2023). *Object Tracking with DeepSORT*. ResearchGate. https://www.researchgate.net/publication/368646461_Object_Tracking_with_DeepSORT [12]
- [25] Roboflow. (2024, 21 Ağustos). *What is ByteTrack in Computer Vision?*. Roboflow Blog. https://blog.roboflow.com/what-is-bytetrack-computer-vision/ [25]
- [1] Fadhil, M. A., et al. (2024). *Vehicle Speed Estimation Using Consecutive Frame Approaches and Deep Image Homography for Image Rectification on Monocular Videos*. ResearchGate. https://www.researchgate.net/publication/386195390_Vehicle_Speed_Estimation_Using_Consecutive_Frame_Approaches_and_Deep_Image_Homography_for_Image_Rectification_on_Monocular_Videos [1]
- [8] Roboflow. (t.y.). *How to Estimate Speed with Computer Vision*. Roboflow Blog. https://blog.roboflow.com/estimate-speed-computer-vision/ [8]
- [14] Nurhadiyatna, A., et al. (2021). *Vehicle Speed Estimation using Optical Flow on Orthographic Projection of Traffic Video*. SIMPLE Conference. https://www.simple.ascee.org/index.php/simple/article/download/30/pdf [14]
- [23] Ultralytics. (t.y.). *Ultralytics GitHub Repository*. GitHub. https://github.com/ultralytics/ultralytics [22]

Alıntılanan çalışmalar

1. (PDF) Vehicle Speed Estimation Using Consecutive Frame ..., erişim tarihi Mayıs 8, 2025, https://www.researchgate.net/publication/386195390_Vehicle_Speed_Estimation_Using_Consecutive_Frame_Approaches_and_Deep_Image_Homography_for_Image_Rectification_on_Monocular_Videos
2. YOLOv8 Speed Estimation: Computer Vision Guide - Ultralytics, erişim tarihi Mayıs 8, 2025, https://www.ultralytics.com/blog/ultralytics-yolov8-for-speed-estimation-in-computer-vision-projects

3. Homography-Based Passive Vehicle Speed Measuring - ucf stars, erişim tarihi Mayıs 8, 2025, https://stars.library.ucf.edu/cgi/viewcontent.cgi?article=1244&context=patents
4. Algorithms for Estimating Mean Vehicle Speed Using Uncalibrated Traffic Management Cameras - wsdot, erişim tarihi Mayıs 8, 2025, https://www.wsdot.wa.gov/research/reports/fullreports/575.1.pdf
5. Efficient Vision-based Vehicle Speed Estimation - arXiv, erişim tarihi Mayıs 8, 2025, https://www.arxiv.org/pdf/2505.01203
6. Efficient Vision-based Vehicle Speed Estimation - arXiv, erişim tarihi Mayıs 8, 2025, https://arxiv.org/pdf/2505.01203?
7. Vehicle Speed Estimation with YOLO11 & OpenCV - Kaggle, erişim tarihi Mayıs 8, 2025, https://www.kaggle.com/code/mateuszk013/vehicle-speed-estimation-with-yolo11-opencv
8. How to Estimate Speed with Computer Vision - Roboflow Blog, erişim tarihi Mayıs 8, 2025, https://blog.roboflow.com/estimate-speed-computer-vision/
9. Vehicle speed measurement method using monocular cameras - PMC, erişim tarihi Mayıs 8, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC11754901/
10. The Complete Guide to Object Tracking [Tutorial] - Encord, erişim tarihi Mayıs 8, 2025, https://encord.com/blog/object-tracking-guide/
11. Deep SORT: Realtime Object Tracking Guide - Ikomia, erişim tarihi Mayıs 8, 2025, https://www.ikomia.ai/blog/deep-sort-object-tracking-guide
12. (PDF) Object Tracking with DeepSORT | - ResearchGate, erişim tarihi Mayıs 8, 2025, https://www.researchgate.net/publication/368646461_Object_Tracking_with_DeepSORT
13. Top 10 Video Object Tracking Algorithms in 2025 - Encord, erişim tarihi Mayıs 8, 2025, https://encord.com/blog/video-object-tracking-algorithms/
14. www.simple.ascee.org, erişim tarihi Mayıs 8, 2025, https://www.simple.ascee.org/index.php/simple/article/download/30/pdf
15. Efficient Vision-based Vehicle Speed Estimation - arXiv, erişim tarihi Mayıs 8, 2025, https://arxiv.org/html/2505.01203v1
16. What is YOLOv8 Architecture? A Step-By-Step Guide - Folio3 AI, erişim tarihi Mayıs 8, 2025, https://www.folio3.ai/blog/what-is-yolov8-architecture/
17. YOLOv8 Architecture Explained: Key Features, erişim tarihi Mayıs 8, 2025, https://yolov8.org/yolov8-architecture-explained/
18. What is YOLOv8? A Complete Guide - Roboflow Blog, erişim tarihi Mayıs 8, 2025, https://blog.roboflow.com/what-is-yolov8/
19. YOLOv8: Object Detection Algorithm for Accurate Recognition, erişim tarihi Mayıs 8, 2025, https://yolov8.org/
20. Understanding YOLOv8 Architecture, Applications & Features - Labellerr, erişim tarihi Mayıs 8, 2025, https://www.labellerr.com/blog/understanding-yolov8-architecture-applications-features/
21. Comparative Analysis of YOLO Variants Based on Performance Evaluation for

Object Detection | ITM Web of Conferences, erişim tarihi Mayıs 8, 2025, https://www.itm-conferences.org/articles/itmconf/abs/2025/01/itmconf_dai2024_03008/itmconf_dai2024_03008.html
22. Model Comparison: YOLOv10 vs YOLOv8 for Object Detection - Ultralytics YOLO, erişim tarihi Mayıs 8, 2025, https://docs.ultralytics.com/compare/yolov10-vs-yolov8/
23. ultralytics/ultralytics: Ultralytics YOLO11 - GitHub, erişim tarihi Mayıs 8, 2025, https://github.com/ultralytics/ultralytics
24. A Complete Guide To AI Object Tracking with DeepSORT - Nextbrain, erişim tarihi Mayıs 8, 2025, https://www.nextbraintech.com/blog/ai-object-tracking-with-deepsort
25. What is ByteTrack? A Deep Dive. - Roboflow Blog, erişim tarihi Mayıs 8, 2025, https://blog.roboflow.com/what-is-bytetrack-computer-vision/
26. (PDF) Uncalibrated camera approach for vehicle speed estimation using detection and tracking - ResearchGate, erişim tarihi Mayıs 8, 2025, https://www.researchgate.net/publication/380410353_Uncalibrated_camera_approach_for_vehicle_speed_estimation_using_detection_and_tracking
27. Intersection over Union (IoU) Explained - Ultralytics, erişim tarihi Mayıs 8, 2025, https://www.ultralytics.com/glossary/intersection-over-union-iou
28. What is Intersection over Union (IoU)? | Definition - Encord, erişim tarihi Mayıs 8, 2025, https://encord.com/glossary/iou-definition/
29. Hierarchical IoU Tracking based on Interval - arXiv, erişim tarihi Mayıs 8, 2025, https://arxiv.org/html/2406.13271v1
30. ByteTrack Object Detection Model: What is, How to Use - Roboflow, erişim tarihi Mayıs 8, 2025, https://roboflow.com/model/bytetrack
31. Speed Estimation using Ultralytics YOLO11, erişim tarihi Mayıs 8, 2025, https://docs.ultralytics.com/guides/speed-estimation/
32. A Vehicle Monocular Ranging Method Based on Camera Attitude Estimation and Distance Estimation Networks - MDPI, erişim tarihi Mayıs 8, 2025, https://www.mdpi.com/2032-6653/15/8/339
33. ClipGrader: Leveraging Vision-Language Models for Robust Label Quality Assessment in Object Detection - arXiv, erişim tarihi Mayıs 8, 2025, https://arxiv.org/html/2503.02897v1
34. YOLOv8 Documentation: A Deep Dive into the Documentation, erişim tarihi Mayıs 8, 2025, https://yolov8.org/yolov8-documentation/
35. A Complete Review of the OpenCV Object Tracking Algorithms - BroutonLab, erişim tarihi Mayıs 8, 2025, https://broutonlab.com/blog/opencv-object-tracking/
36. Speed Estimation of ANY Object in Video using Computer Vision (Vehicle Speed Detection with YOLO 11) : r/computervision - Reddit, erişim tarihi Mayıs 8, 2025, https://www.reddit.com/r/computervision/comments/1iutg00/speed_estimation_of_any_object_in_video_using/
37. Analysis of a Bounding Box Heuristic for Object Intersection - ResearchGate, erişim tarihi Mayıs 8, 2025, https://www.researchgate.net/publication/2500978_Analysis_of_a_Bounding_Box_Heuristic_for_Object_Intersection

38. Performance Evaluation of YOLOv8, YOLOv9, YOLOv10, and YOLOv11 for Stamp Detection in Scanned Documents - MDPI, erişim tarihi Mayıs 8, 2025, https://www.mdpi.com/2076-3417/15/6/3154
39. [2505.01203] Efficient Vision-based Vehicle Speed Estimation - arXiv, erişim tarihi Mayıs 8, 2025, https://arxiv.org/abs/2505.01203