Görüntü İşleme Temelli Park Alanı Sayacı ve Yönlendirme Sistemi Uygulaması

**Fırat Üniversitesi Teknoloji Fakültesi Yazılım Mühendisliği Bölümü**

210542007@firat.edu.tr

**Öz**

Bu çalışma, bir video akışı üzerinden park alanlarının doluluk durumunu gerçek zamanlı olarak takip eden ve boş yer bulduğunda araçları yönlendiren bir akıllı park sistemi uygulamasını sunmaktadır. Sistem, temel görüntü işleme tekniklerini kullanarak hareketli nesneleri (araçları) algılar, tanımlanmış park alanlarını izler ve boş yerleri tespit eder. Giriş noktasından bir araç algılandığında, sistem videoyu duraklatır, en yakın boş park yerini bulur ve sürücüye görsel ve metinsel yönlendirmeler sağlar. Uygulama, OpenCV kütüphanesi kullanılarak Python dilinde geliştirilmiştir. Park yerlerinin manuel olarak işaretlenmesine ve giriş noktalarının belirlenmesine olanak tanıyan bir araç da mevcuttur.

**1. Giriş**

Modern şehirlerde artan araç sayısı, otoparklarda boş yer bulmayı giderek zorlaştırmakta ve bu durum trafik yoğunluğuna, zaman kaybına ve çevresel kirliliğe yol açabilmektedir. Geleneksel otopark sistemleri bu zorluklara yeterli çözümler sunamamaktadır. Bilgisayarla görme ve görüntü işleme teknolojileri, bu alanda akıllı ve verimli sistemlerin geliştirilmesi için büyük bir potansiyel sunmaktadır. Bu proje, bir otoparkın gerçek zamanlı olarak izlenmesi, boş park yerlerinin tespit edilmesi ve araçların bu boş yerlere etkin bir şekilde yönlendirilmesi için görüntü işleme tabanlı bir sistem geliştirmeyi amaçlamaktadır. Sistem, video akışından gelen kareleri analiz ederek park alanlarının doluluk durumunu belirler ve kullanıcıya anlık ve konsol tabanlı yönlendirmeler sağlar.

**2. Materyal ve Metot**

**2.1. Veri Seti (Video Kaynağı ve İlk Kare)**

Bu projede, gerçek bir otoparkın CCTV kamerasından elde edilmiş bir video akışı (video.mp4) kullanılmaktadır. Sistemin park alanlarını ve giriş noktalarını tanımlayabilmesi için videodan alınan ilk kare (first\_frame.png) referans olarak kullanılmıştır. Bu ilk kare üzerinde kullanıcı, park yerlerinin ve araç giriş noktalarının konumlarını manuel olarak belirler.

**2.2. Görüntü İşleme ve Algoritma**

Sistem, aşağıdaki adımları izleyerek çalışmaktadır:

 **Park Alanı ve Giriş Noktası Belirleme:**

* parking\_space\_picker.py betiği kullanılarak first\_frame.png görüntüsü üzerinde park alanları (dikdörtgen bölgeler) ve araç giriş noktaları manuel olarak işaretlenir.
* Park yerleri için width = 27 ve height = 13 piksel boyutları kullanılırken, giriş noktaları için entry\_point\_width = 50 ve entry\_point\_height = 20 piksel boyutları tanımlanmıştır.
* Bu konum bilgileri (CarParkPos ve EntryPoints dosyaları) pickle modülü ile kaydedilerek ana uygulamada kullanılır.

 **Video Akışı ve Ön İşleme:**

* parking\_space\_counter.py betiği, video.mp4 dosyasını okuyarak video akışını başlatır.
* Her kare gri tonlamaya dönüştürülür (cv2.cvtColor).
* Gürültüyü azaltmak için Gauss bulanıklaştırma (cv2.GaussianBlur) uygulanır.
* Park alanı doluluk tespiti için adaptif eşikleme (cv2.adaptiveThreshold), medyan bulanıklaştırma (cv2.medianBlur) ve genişletme (cv2.dilate) işlemleri uygulanarak ikili (siyah-beyaz) bir görüntü elde edilir.

 **Park Alanı Doluluk Tespiti (checkParkSpace):**

* Tanımlanan her bir park alanı için, ön işlemden geçmiş ikili görüntüdeki o bölgedeki sıfır olmayan piksel (beyaz piksel) sayısı (cv2.countNonZero) sayılır.
* Bu sayı (count) park\_threshold (150) değerinden küçükse, park alanı boş kabul edilir ve yeşil bir dikdörtgenle işaretlenir; aksi takdirde dolu kabul edilir ve kırmızı bir dikdörtgenle işaretlenir.
* Toplam boş park yeri sayısı ekran üzerinde gösterilir.

 **Araç Algılama ve Yönlendirme (detect\_car\_in\_entry\_point, find\_nearest\_empty\_spot, provide\_guidance):**

* **Hareket Algılama:** Video akışındaki hareketli nesneleri algılamak için mevcut kare ile videonun ilk karesi arasındaki mutlak fark (cv2.absdiff) hesaplanır. Bu fark, eşikleme (cv2.threshold) ve genişletme (cv2.dilate) ile işlenerek hareketli bölgeleri belirleyen bir maske oluşturulur.
* **Kontur Algılama:** Hareket maskesi üzerinde kontur algılama (cv2.findContours) yapılarak hareket eden nesnelerin sınırları belirlenir.
* **Araç Tespiti:** Algılanan konturların alanı car\_min\_area (500) değerinden büyükse ve konturun merkezi tanımlanan giriş noktalarından birinin içindeyse, bir araç algılanmış kabul edilir.
* **Video Duraklatma ve Boş Yer Arama:** Bir araç algılandığında, sistem videoyu duraklatır ve konsola "Yeni arac algilandi! Boş yer aranıyor..." mesajını yazdırır. Ardından, algılanan aracın konumuna en yakın boş park yerini (find\_nearest\_empty\_spot) bulur.
* **Yönlendirme:** En yakın boş yer bulunduğunda, "Hedef boş yer bulundu. Yönlendiriliyor... Devam etmek için tıklayın." mesajı görüntülenir ve konsola yazdırılır. Kullanıcı tıklayarak videoyu devam ettirdiğinde, provide\_guidance fonksiyonu aracın hedef park yerine doğru hareketine göre dinamik yönlendirme mesajları (örn: "Saga dogru ilerle!", "Asagiya (ileri) dogru ilerle!", "Park et!") sağlar ve bunları hem görüntü üzerinde hem de konsolda gösterir.
* **Park Etme Tamamlanması:** Araç, hedef park yerine parking\_completion\_distance (20 piksel) mesafesine geldiğinde, "Arac park edildi! Video devam ediyor." mesajı gösterilir ve video tekrar otomatik olarak oynatılmaya başlanır.

**2.3. Performans Metrikleri**

Bu proje, bir makine öğrenmesi sınıflandırma modeli içermediğinden, doğruluk, kesinlik, duyarlılık gibi metrikler doğrudan uygulanamaz. Sistem performansı, aşağıdaki operasyonel ve niteliksel ölçütler üzerinden değerlendirilmiştir:

* **Algılama Doğruluğu:** Araçların giriş noktalarında ve park yerlerinde ne kadar güvenilir bir şekilde algılandığı.
* **Doluluk Tespit Doğruluğu:** Park yerlerinin dolu veya boş durumunun ne kadar doğru belirlendiği.
* **Yönlendirme Etkinliği:** Verilen yönlendirmelerin sürücülerin boş park yerine ulaşmasına ne kadar yardımcı olduğu.
* **Gerçek Zamanlılık:** Sistemin video akışını herhangi bir gözle görülür gecikme olmaksızın işleyebilme yeteneği.
* **Kullanıcı Deneyimi:** Sistemle olan etkileşimin (duraklatma, devam etme, mesajların netliği) genel kolaylığı ve sezgiselliği.

**3. Deneysel Bulgular**

Geliştirilen akıllı park alanı sayacı ve yönlendirme sistemi, belirtilen video.mp4 üzerinde başarıyla test edilmiş ve aşağıdaki bulgular elde edilmiştir:

* **Park Alanı ve Giriş Noktası Tanımlama:** parking\_space\_picker.py aracıyla belirlenen park alanları ve giriş noktaları, first\_frame.png üzerinde doğru bir şekilde yansıtılmış ve ana uygulamada sorunsuz bir şekilde yüklenmiştir. Görselleştirilen giriş alanları, algılama hassasiyetini artırmak için yeterince büyük bir bölgeyi kapsamıştır.
* **Park Alanı Doluluk Tespiti:** checkParkSpace fonksiyonu, her bir park yerindeki piksel yoğunluğunu (count) doğru bir şekilde hesaplamış ve bu değere göre park yerlerinin dolu/boş durumunu başarıyla ayırt etmiştir. Park yerlerinin renkleri (kırmızı/yeşil), gerçek zamanlı olarak araçların giriş çıkışına göre dinamik olarak güncellenmiştir. Boş park yeri sayısı her zaman güncel olarak ekranda gösterilmiştir.
* **Araç Algılama ve Yönlendirme Akışı:**
  + Video giriş alanına bir araç girdiğinde, sistem anında bir araç algılamış, videoyu duraklatmış ve hem konsolda hem de uygulama penceresinde "Yeni arac algilandi! Boş yer aranıyor..." mesajını göstermiştir.
  + En yakın boş park yeri (find\_nearest\_empty\_spot tarafından) doğru bir şekilde belirlenmiş ve sarı bir dikdörtgenle işaretlenmiştir.
  + Kullanıcının ekran üzerine tıklamasıyla video akışı devam ettiğinde, sistem aracın hedefe olan göreceli konumuna göre "Saga dogru ilerle!", "Asagiya (ileri) dogru ilerle!" gibi dinamik yönlendirme mesajlarını sürekli olarak hem konsolda hem de görsel arayüzde sunmuştur. Bu mesajlar, aracın hareketine anında tepki vermiştir.
  + Araç, belirlenen hedef park yerine yaklaştığında ("Park et!" mesajı ile), sistem aracın park ettiğini tespit etmiş ve videoyu otomatik olarak devam ettirmiştir.
* **Sistem Tepkiselliği:** Sistem, video akışını gerçek zamanlıya yakın bir performansla işlemiş ve gözle görülür bir gecikme yaşanmamıştır. Kullanıcı tarafından p tuşuyla yapılan manuel duraklatma/devam ettirme ve fare tıklamasıyla video akışını kontrol etme işlevleri sorunsuz çalışmıştır.

**4. Tartışma ve Sonuçlar**

Bu çalışma, temel görüntü işleme tekniklerinden yararlanılarak geliştirilmiş, etkili bir akıllı park alanı sayacı ve yönlendirme sistemi uygulamasını ortaya koymuştur. Proje, otoparklardaki boş yer bulma sorununa pratik ve gerçek zamanlı bir çözüm sunarak, hem sürücülerin zamanını optimize etme hem de otopark yönetiminin verimliliğini artırma potansiyeline sahiptir. Sistem, araçları algılama, boş park yerlerini tespit etme ve sezgisel yönlendirmeler sağlama yeteneğini başarılı bir şekilde göstermiştir.

**Geliştirme Alanları ve Gelecek Çalışmalar:**

Mevcut sistemin temel fonksiyonları başarılı olsa da, daha sağlam ve kapsamlı bir çözüm için aşağıdaki alanlarda geliştirmeler yapılabilir:

* **Çevresel Koşullara Dayanıklılık:** Sistemin performansı, farklı ışıklandırma koşulları (gündüz/gece, parlak güneş ışığı, gölgeler), hava durumu değişiklikleri (yağmur, kar) veya kamera açısındaki değişiklikler gibi çevresel faktörlerden etkilenebilir. Gelecekte, daha gelişmiş arka plan çıkarma algoritmaları (örn. MOG2, GMG) veya adaptif eşikleme yöntemleri ile bu tür durumlara karşı dayanıklılık artırılabilir.
* **Gürültüye Karşı Direnç:** Park yeri içerisindeki küçük hareketler veya yansımalar "dolu" olarak algılanabilir. Daha sofistike filtreleme teknikleri veya alanın belirli bir yüzdesinin hareketli olması gibi kriterler eklenerek yanlış pozitifler azaltılabilir.
* **Çoklu Araç Takibi:** Mevcut sistem, yeni bir araç algılandığında bir yönlendirme süreci başlatır ve bu süreç tamamlanana kadar başka bir aracın algılanmasını beklemez (ya da videoyu duraklatır). Birden fazla aracın aynı anda otoparka girdiği senaryolar için, nesne takibi algoritmaları (örneğin Kalman Filtresi, SORT veya DeepSORT) entegre edilerek eş zamanlı çoklu araç takibi ve her araç için ayrı ayrı yönlendirme sağlanabilir.
* **Derin Öğrenme Entegrasyonu:** Daha gelişmiş ve güvenilir araç tespiti için derin öğrenme tabanlı nesne algılama modelleri (örneğin YOLOv5/v8, SSD) kullanılabilir. Bu modeller, farklı araç tiplerini daha doğru bir şekilde ayırt edebilir ve değişen koşullara daha iyi adapte olabilir. Ayrıca, park yeri doluluk tespiti için de CNN tabanlı sınıflandırma modelleri (boş/dolu) kullanılabilir.
* **Kullanıcı Arayüzü (GUI) Geliştirme:** Mevcut arayüz, OpenCV'nin temel pencereleme yeteneklerini kullanmaktadır. Daha kullanıcı dostu ve işlevsel bir grafik kullanıcı arayüzü (örneğin PyQt, Tkinter veya web tabanlı bir arayüz) geliştirilerek kullanıcının sistemle etkileşimi zenginleştirilebilir, geçmiş verilerin görüntülenmesi veya sistem ayarlarının kolayca yapılandırılması sağlanabilir.
* **Park Yeri Yerleşim Optimizasyonu:** Büyük otoparklarda, sadece en yakın boş yeri göstermek yerine, sürücüyü otoparkın daha az yoğun olan bir bölgesine yönlendirme gibi stratejiler de uygulanabilir.

Bu proje, görüntü işleme alanındaki temel prensiplerin gerçek dünya problemine nasıl uygulanabileceğine dair somut bir örnek teşkil etmektedir ve gelecekteki akıllı şehir altyapıları için potansiyelini ortaya koymaktadır.

**KAYNAKLAR**

[1] OpenCV Resmi Belgelendirmesi: https://docs.opencv.org/

[2] NumPy Resmi Belgelendirmesi: <https://numpy.org/doc/>

[3] Python Resmi Belgelendirmesi: <https://docs.python.org/>

[4] ChatGPT

[5] Gemini