SAKARYA
UYGULAMALI BİLİMLER
ÜNİVERSİTESİ

T.C.

SAKARYA UYGULAMALI BİLİMLER ÜNİVERSİTESİ

TEKNOLOJİ FAKÜLTESİ

GÖRÜNTÜ İŞLEME PROJE ÖDEVİ

2023-2024 BAHAR DÖNEMİ

AD-SOYAD:	***Seymen Can Aydoğan***
NUMARA:	***B210109063***
BÖLÜM:	***Bilgisayar Mühendisliği***

Programın Genel Çalışma Yapısı ve Kütüphaneler: Bu program, video içindeki kırmızı topların hareketlerini izlemek ve çarpışmalarını tespit etmek için kullanılan bir görüntü işleme uygulamasıdır. Genel olarak, kullanılan kütüphaneler, sınıf yapıları ve işlevler aşağıda açıklanmıştır.

Kullanılan Kütüphaneler:

1. **OpenCV (cv2):** Görüntü işleme ve video okuma/yazma işlemleri için kullanılır. OpenCV, bilgisayarla görme görevlerinde yaygın olarak kullanılan bir kütüphanedir.
2. **Numpy:** Sayısal işlemler ve matris hesaplamaları için kullanılır. Numpy, çok boyutlu diziler ve matrislerle çalışmak için temel araçlar sunar.
3. **OrderedDict:** Nesnelerin sıralı şekilde saklanması için kullanılır. Python'un collections modülünde bulunan OrderedDict, ekleme sırasını koruyan bir sözlük veri yapısıdır.

MerkezTracker Sınıfı, nesnelerin merkez noktalarını ve izlerini takip eder. Sınıfın ana bileşenleri şunlardır:

1. **__init__ fonksiyonu:** Nesne ID'lerini, merkezlerini, izlerini ve renklerini saklamak için OrderedDict yapıları oluşturur. Ayrıca kaybolan nesneleri takip eder ve maksimum kaybolma süresini belirler.
2. **kaydet fonksiyonu:** Yeni bir nesnenin merkezini, izini ve rengini kaydeder.
3. **kayitsil fonksiyonu:** Kaybolan bir nesneyi listeden siler.
4. **_renk_olustur fonksiyonu:** Her nesne için rastgele bir renk oluşturur.
5. **guncelle fonksiyonu:** Yeni merkezlere göre nesnelerin izlerini günceller ve kaybolan nesneleri kontrol eder. Yeni merkez yoksa mevcut nesnelerin kaybolma süresini artırır ve süreyi aşanları siler. Mevcut nesneler ve yeni merkezler arasındaki mesafeleri hesaplayarak en yakın merkezleri eşleştirir ve izleri günceller. Kullanılmayan merkezler için yeni nesneler kaydeder.

top_bul Fonksiyonu, belirli bir renk aralığındaki topları tespit eder. Görüntüyü HSV renk uzayına çevirir ve verilen renk aralığına göre maske oluşturur. Kırmızı ve beyaz toplar için maskeler oluşturur ve maske üzerindeki konturları bulur, her konturun merkezini hesaplar.

Video İşleme ve Çarpışma Tespitinde, video kaydını okuyarak her kare için topların yerlerini tespit eder ve MerkezTracker ile kırmızı topların izlerini günceller. Beyaz top ile topların çarpışmasını daha sonra kırmızı topların birbirleriyle çarpışmasını kontrol eder. Çarpışma tespit edilirse, bu bilgileri videoda gösterir ve çarpışma sayısını kaydeder.

Sonuç olarak, videoda topların hareketini izleyip kırmızı topların çarpışmalarını tespit eder ve bu bilgileri videoda gösterir. Videonun sonunda izlenen nesnelerin izleri ayrı ayrı görüntü dosyalarına kaydedilir ve çarpışma sayısı ekrana yazılır. Bu sayede, görüntü işleme ve nesne takibi konularında pratik bir uygulama gerçekleştirilmiş olur.

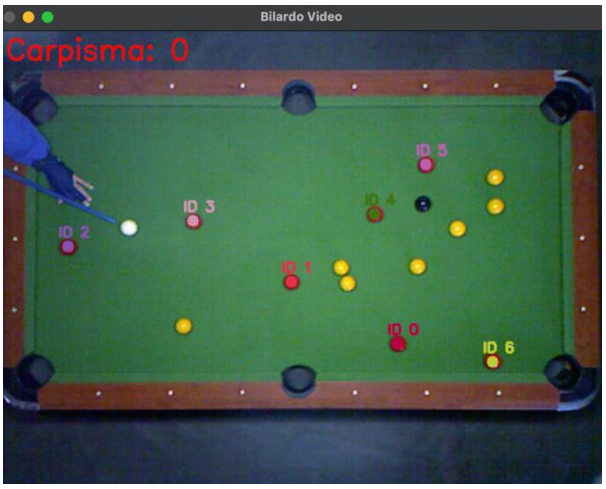
Algoritma:

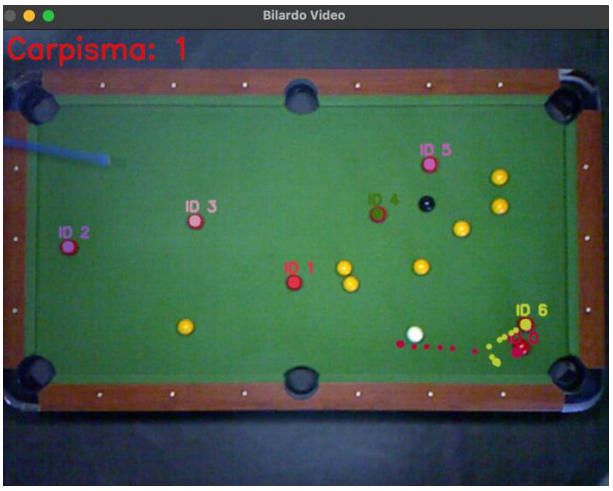
1) Video dosyasını okunur.
2) Bir döngü *(`while True:`)* başlatılır.
3) Kareden ilgi alanı (ROI) seçilir.
4) Kırmızı ve beyaz toplar tespit edilir.
5) Kırmızı ve beyaz topların pozisyonları tespit edilir.
6) Kırmızı topların pozisyonları kullanılarak nesne izleyici güncellenir.
7) Her bir izlenen nesne için, merkezi hesaplanır ve güncellenir. Daha sonra, iz noktaları güncellenir ve çizilir. Yeni iz noktaları da bir listeye atanır.
8) Eğer beyaz top pozisyonları varsa, beyaz topun merkezi hesaplanır. Beyaz top ile diğer topların mesafesi kontrol edilir. Mesafe belirli bir eşikten küçükse, beyaz topun diğer topa çarptığı tespit edilir.
9) Eğer beyaz top kırmızı topa çarpmışsa, çarpışma çiftleri kontrol edilir. Tüm toplar durana kadar, her çarpışma çifti için mesafe kontrol edilir ve eğer eşikten küçükse çarpışma kaydedilir. Çarpışma çizgileri çizilir ve çarpışma sayısı artırılır.

10) Çarpışma sayısı ekrana yazdırılır.
11) Döngü sonlandırılır ve kaynaklar serbest bırakılır.

Programın Çalışması: Programı vid_1.avi ile programladım şimdi **vid_2.avi** ile test edip görüntüleri adım adım açıklayalım.

1. Program başladığında her topa ayrı bir renk ve ID'i atanacaktır bu programın çıktısını elde ederken bize daha efektif ve görsel bir sonuç üretecektir (Görsel 1.0). Hamle başladığında yani beyaz topla kırmızı top vurulduğu anda artık toplarda hareketlenmeler başladığına göre kırmızı toplar arasında çarpışmaları hesaplayabiliriz ve resimde görüldüğü üzere ID 0 ve ID 6 çarpışıyor ve çarpışma sayıları 1'e yükseliyor (Görsel 1.1).

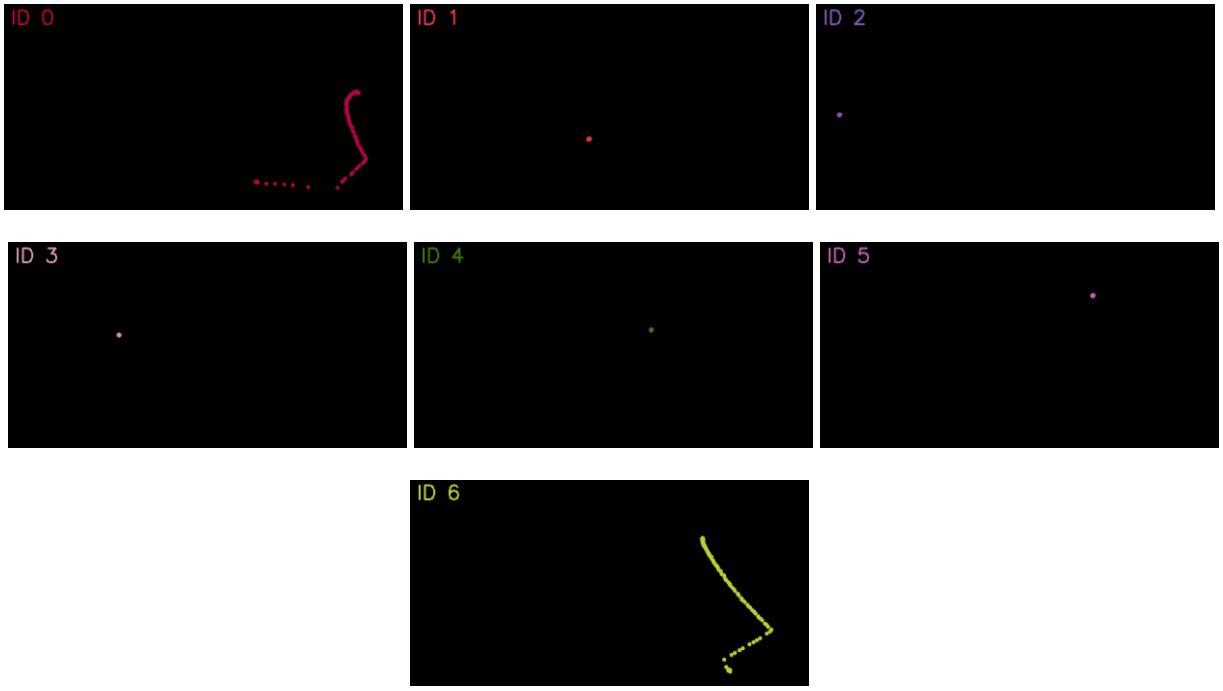




Görsel 1.0 Çarpışma Öncesi

Görsel 1.1 Çarpışma Sonrası

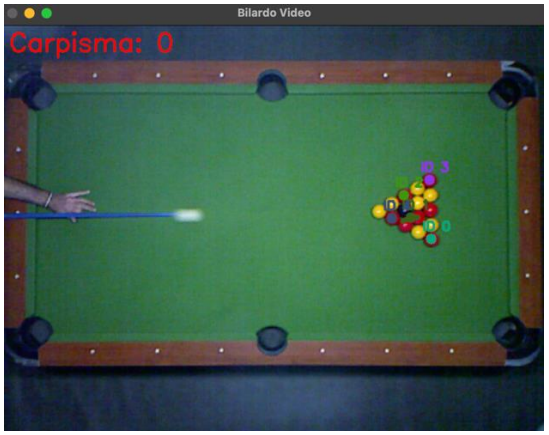
2. Hamle bittiğinde yani tüm toplar durduğunda her kırmızı topun (7 adet kırmızı top) ayrı ayrı 7 görüntüsünü aşağıdaki gibi dosyaya kaydediliyor. (Görsel 2.0).

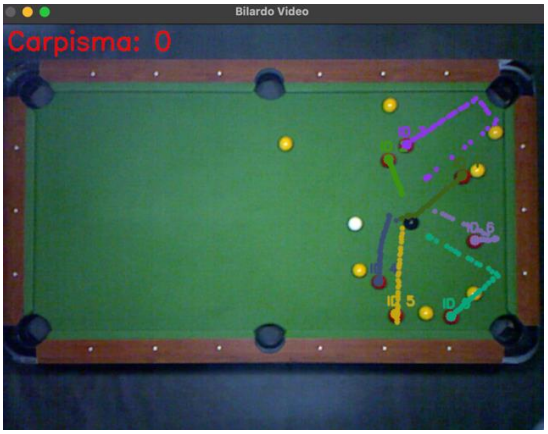


Görsel 2.0 Kırmızı Top Çıktıları

Çıkarımlar ve İyileştirmeler:

- Program ve algoritma mantığı genel olarak Programın Çalışması kısmında görüldüğü gibi **vid_2.avi ile sorunsuz çalışmaktadır.** vid_1.avi'de program çarpışmayı hesaplayamamaktadır (Görsel 3.1). Aşağıdaki resimde gördüğünüz üzere ID2 ve ID3 topları çarpıştıktan sonra 1 olmuyor bu da carpisma_mesafe_esigi = 22 olduğundan kaynaklanabilir. Optimum çalışacak mesafe eşiğini veya topun alanını 250-350 arasına çekersek bu sorun düzelebilir.

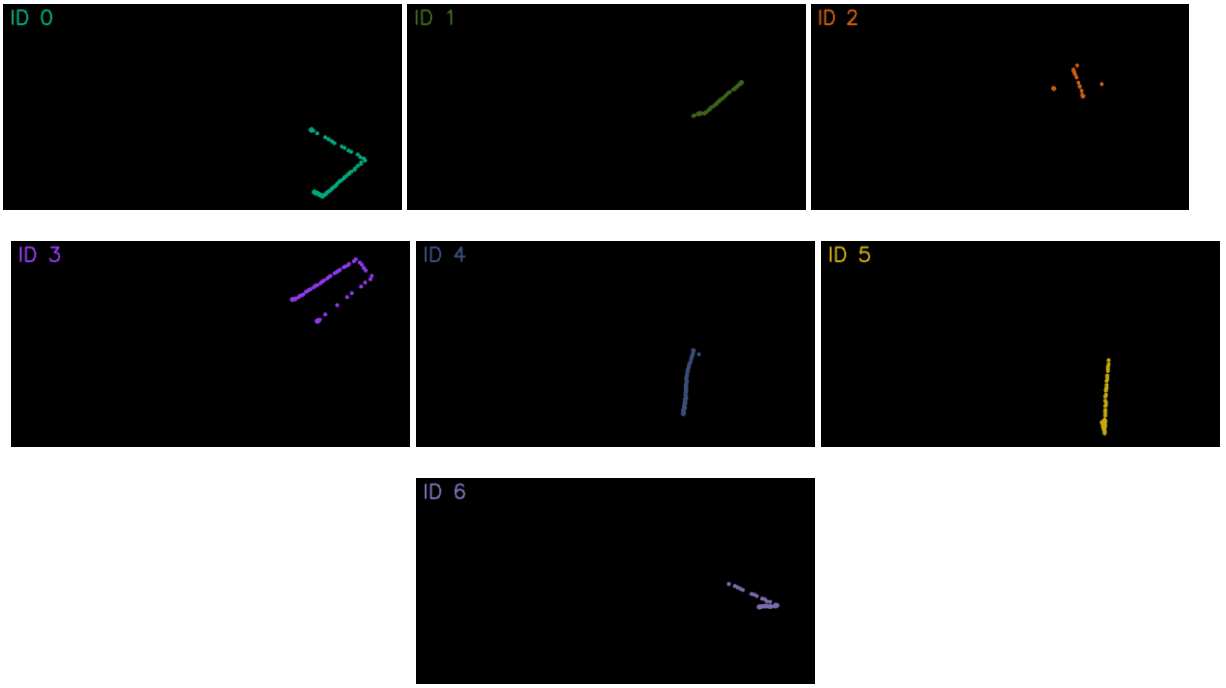




Görsel 3.0 Çarpışma Öncesi (vid_1)

Görsel 3.1 Çarpışma Sonrası (vid_1)

- Aynı zamanda, video1'de hamle bittiğinde yani tüm toplar durduğunda her kırmızı topun ayrı 7 görüntüsünü aşağıdaki gibi dosyaya sorunsuz kaydediliyor (Görsel 4.0).



Görsel 4.0 Kırmızı Top Çıktıları (vid_1)

- Ayrıca iyileştirme önerisi olarak, MerkezTracker Sınıfında izleme sırasında nesnelerin izlerini sınırlı bir şekilde saklayarak bellek kullanımını azaltabilir ve performansı arttırabiliriz. Örneğin, her nesne için sadece son birkaç konumu saklayabiliriz.

Akış Diyagramı