Optik akış ile video sabitleme (video stabilization with optical flow) algoritması, bir video çerçevesinin hareketini izleyerek istikrarlı bir video oluşturmayı amaçlayan bir görüntü işleme tekniğidir. Staj sürecim sırasında geliştirilen Optik Akış ile Video Sabitleme algoritması adım adım açıklanmaktadır.

Genel bakış olarak algoritma şu şekilde çalışmaktadır;

Girdi videosu seçilir ve videodaki kareler (frame) alınır. Sonrasında ardışık kareler üzerinde feature deteciton algoritması (sift feature detection) ile feature detecting işlemi yapılır ve bir dizi feature elde edilir. Elde edilen iki kare arasındaki featurelar birbirleriyle eşleştirilir (flann fature match ile). Eşleştirme sonucu elde edilen sonuçlardan belli bir eşik değerden uzak olan eşleştirmeler elenerek daha stabil bir eşleştirme sonucu elde edilir. Eşleştirilen featureların piksel koordinatları alınır ve her iki pikselin x ve y koordinatları arasında çıkarma işlemi yapılarak birbirlerine olan uzaklıkları elde edilir. Elde edilen uzaklık değerlerinin hepsi toplanarak ortalama bir hareket değeri elde edilir. Bu hareket değeri iki kareyi birbirine yaklaştırma, sabitleme için kullanılacak değerdir. Sonradan gelen ikinci kare bu değere zıt bir şekilde kaydırılır ve hareket eden iki kare daha sabit hale getirilir.

Girdi videosunu almak için açılan dosya seçme ekranı ekledim ve seçilen dosyayı yazdığım karelerine ayrıma fonksiyonuna göndererek dönüt olarak kareleri içeren bir dizi elde edilir. Sonrasında tüm işlemlerin yapılacağı ana fonksiyon olan “opticalflow” fonksiyonuna kareler dizi halinde gönderilir.

video\_path = fd.askopenfilename()

video\_name = video\_path.split("/")[-1].split(".")[0]

print(video\_name)

#read frames

frames = framesFromVid(video\_path)

#send frames to optical flow function for detect features

#match features , shift frames opposite of movement and fill the blank areas

shiftedframes = opticalflow(frames)

Videodaki kareleri ayıklamak için OpenCV kütüphanesi kullanılmakta ve aşağıdaki fonksiyonla kareler bir diziye aktarılmaktadır.

def framesFromVid(path):

    readed\_frames = []

    vidcap = cv2.VideoCapture(path)

    success,image = vidcap.read()

    count = 0

    while success:

        readed\_frames.append(image)

        success,image = vidcap.read()

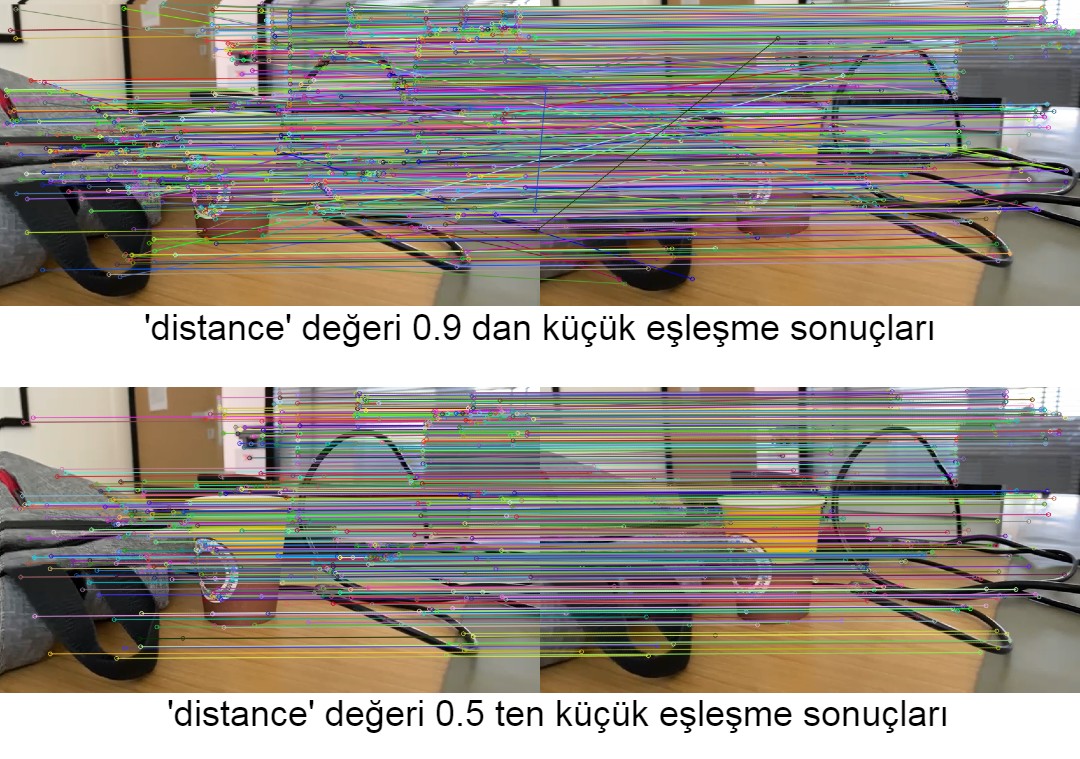
        count += 1

    print("readed frames :",count)

    return readed\_frames

Karelerin kaydırılması gereken değerin hesaplanması için ardışık kareler arasında feature detection işlemi gerekli bunun için OpenCV kütüphanesinin içindeki SIFT algoritması kullanılmıştır. SIFT algoritmasını kullanılmasının sebebi SURF algoritmasına göre daha etkili olmasıdır. İki algoritmayı analiz eden ve karşılaştıran makaleler incelendi ve sonucunda bu karara varılmıştır.

Tespit edilen featureların birbirleriyle eşleştirilmesi için FLANN algoritması kullanılmıştır. Eşleştirmelerde daha stabil ve verimli sonuçlar için eşleşen featureların ‘distance’ değerleri üzerinden bir eleme yaprak stabil olmayan yani ‘distance’ değeri yüksek olan eşleşmeler elenir. Böylelikle daha stabil doğruluğa daha yakın eşleşmeler elde edilmiş olur. Aşağıdaki Şekil 3.2.13 ‘de görüleceği üzere ‘distance’ değerindeki değişimin eşleşmeler üzerindeki etkisi bir örnekle verilmiştir. Eşik arttırıldığında sapmış eşleşmelerin gözle görülür şekilde arttığı gözlemlenebilir. ‘distance’ değeri azaldıkça eşleşmelerin doğruluk oranının arttığı görülür. Ne çok fazla eşleşmeyi eleyerek ne de çok fazla eşleşmeyi tutarak en optimum eşik değeri ile eleme sonucunda stabil en doğru eşleşmeler ayıklanmıştır.



İki kare arasındaki hareketin ne tarafa ve ne kadar hareket ettiğinin hesaplanması için bu eşleşmeler kullanılır. Eşleşen featureların yeni ve eski karelerdeki koordinat değerleri birbirlerinden çıkartılarak x ve y koordinatlarında ne kadar ne tarafa hareket ettikleri elde edilir ve her bir feature için bulunan hareket miktarı toplanır ardından ortalama bir hareket değeri hesaplanır.

Ortalama hareket değerini doğruluğa yaklaştırmak, stabil bir değer elde etmek ve karelerdeki hareket eden veya yanlış eşleşen değerleri elemek için tekrar bir eleme gerçekleştirilir. Bu eleme ortalama değerin belli bir miktar altı ve üstündeki değerlerin yani ortalamadan çok uzaklaşan değerlerin elenmesiyle gerçekleşir. Eleme için ortalama değerin %80’inden fazla veya %20 ‘sinden az olmak üzere iki farklı eşik değer hesaplanır ve bu eşik değerlere göre her eşleşmenin hareket değeri kontrol edilir. Bu aralıkta olmayan değerler eşleşmelerin olduğu dizilerden silinir ve en sonunda ortalama stabil değerlerle bir daha hesaplanır.

Bu işlemler sonucunda her iki kare arasındaki hareket değerlerini hesaplamış oluruz. Bu adımların hepsinin yapıldığı kod kısımları aşağıdaki gibidir.

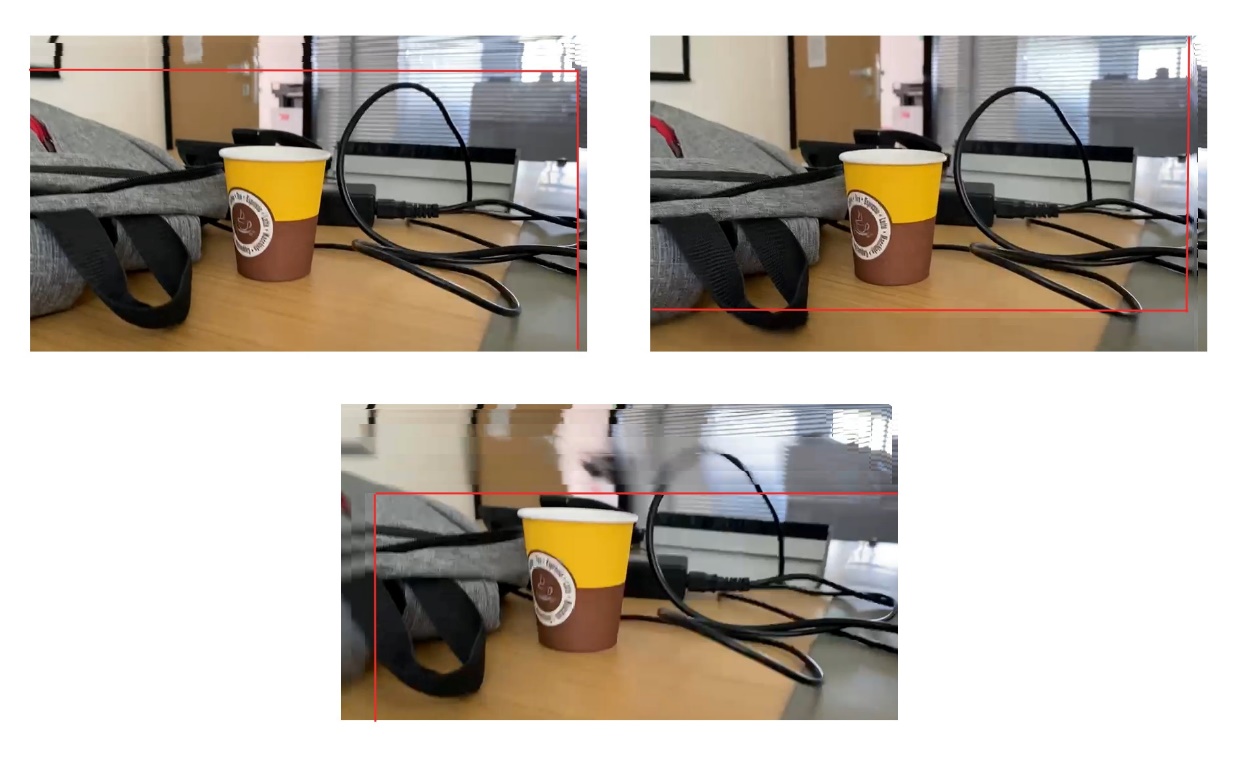
Karelerdeki hareket vektörleri hesaplandıktan ve karelerin hangi yönde ne kadar kaydırılacağı hesaplandıktan sonra kaydırma işlemini yapmak için OpenCV’nin ‘warpAffine’ fonksiyonu çağırılarak kareler kaydırılır. Bu fonksiyonun kaydırma işlemini yapabilmesi için gerekli parametreler sağlanmalıdır. Bunun için kaydırma verisi bir matrise atanır ve fonksiyona parametre olarak gönderilir.

T = np.float32([[1, 0, shiftx], [0, 1, shifty]])

        #shift the image

        img\_translation = cv2.warpAffine(newframe, T, (width , height ))

Kaydırma işlemi sonrası karelerde kaydırılan taraflarda siyah bölgeler oluşur, Şekil 3.2.12’deki gibi. Bu siyah bölgelerin doldurulması yada kesilmesi gibi farklı yaklaşımlar farklı çözümlerde kullanılmış. Bu alanlar her yönden eşit miktarlarda kesilerek videonun boyutları küçülmüş bir şekilde kaydedilebilir. Araştırdığım ve okuduğum makalelerden yola çıkarak bu sorunu çözmek için özgün çözümler üretilmeye çalışılmıştır.

Kesilen kısımları her kaydırma sonrasında orijinal kaydırılmayan karelerden keserek ve siyah alanlara ekleyerek daha doğal bir görüntü elde edilmeye çalışılmıştır. Bunun için ise yine üst taraftaki hesaplanan hangi bölgelerden ne kadar kaydırılma yapıldığını tutan verileri kullanarak kaydırılan siyah bölgeleri ve eski karelerdeki kaydırılan bölgeleri kesip eklemeler yapılmıştır. Ekleme işlemi için yukarda bulunan koddaki gibi OpenCV içerisindeki ‘vconcat’ fonksiyonu kullanılmıştır.

Sonucunda Şekil 3.19’da göründüğü gibi üst üste binmiş kareler oluşmuştur. Bir önceki yönteme göre daha doğal ve yumuşak olsa da tam istenilen ve efektif bir yöntem değildir.