CSE 463 COMPUTER VISION

HW02

ÖDEV RAPORU

Mehmet Satılmış 101044011

1. Rapor İçeriği

Bu rapor CSE 463 dersi kapsamında verilmiş olan HW02 adlı, finger trip detection ödevini açıklamak için hazırlanmıştır.

2. Algoritmanın Açıklanması

Algoritma arka fona siyah bir kartonun (mukavva) bulunması ve bunun üzerinde hareket edecek elin parmak uçlarını tanıyıp, bu uçların takip edilebilmesi üzerine kurulmuştur.

Elin arka fondan ayrıt edilebilmesi için frame in RGBtoGray değeri üzerinde binary threshold kullanılmıştır. Threshold değerinin ışık farklı koşullarda tespit edilebilmesi için k-means algoritması kullanılmıştır.

Parmakların uçlarının ayırt edilebilmesi için ise elin doğal yapısına dayanılarak 2 farklı duruma bakılmıştır. Bunlar 2 parmak arasındaki açının ölçülmesi, bunun min. ve max. değerler arasında olması ayrıca ölçülmekte olan parmağın boyutunun belirli bir uzunlukta olması, 2. durum ise bir parmağın uç noktasından kendine komşu olan 2 çukur noktası ile oluşturmuş olduğu açısal değer şeklindedir.

Parmak uçlarının takip edilmesi için ise parmak uçlarının ilk framede adlandırılmasından sonra yeni gelen framelerde bulunmuş olan yeni konumlar arasında en kısa mesafeler ilk adlandırılma ile eşleştirilmesi şeklinde gerçekleştirilmiştir. İlk framede tüm ellerin olmaması gibi durumlar bazı kısıtlmalarla handle edilmiştir.

3. Programın Çalıştırılması

Program upload edilen klasörde bulunan makefile ile başlatılmasından sonra ilk 5 saniye kullanıcının kamerayı arkafona göre ayarlaması ve elini fonun önüne koyması için bir işlem yapmadan beklemektedir. Bunun ardından myKmeans fonksiyonunun hesapladığı threshold değeri ile algoritma başlatılmaktadır. Eğer kamera görüntüsünde arka fon harici gürültüye sebeb olacak durumlar varsa elin kamera görünen kısmının gürültülerden daha büyük olması beklenmektedir. Parmak uçları kırmızı ile çizilmiş bir çember ile gösterilir ve her parmağa bir id verilmektedir. Programın sonlandırılması için ESC tuşuna basılmalıdır.

Upload edilmiş klasör içerisinde örnek bir ekran görüntüsü bulunmaktadır.

4. Fonksiyonların Açıklanması

1-) **myKmeans(Mat src_my, Mat src_gray_my)** : RGB halinde verilen kamera görüntüsünden opencv nin kmeans algoritması implementasyonunu kullanarak 2 sınıflandırılmış bilgi elde eder. Bu görüntüleri RGB görüntünün intensity değerlerini içeren src_gray üzerinde2 sınıfın pozisyonlarına bakarak ortalama intensity değerlerini ölçer. Çıkan sonuçların ortalama değerini alarak binary threshold da kullanılacak değeri çıkarmış olur.

2-) **kmeans** : OpenCv nin implement ettiği bir fonksiyondur. Kmeans algoritması ilk olarak merkezde rastgele n nokta seç (n tane cluster), geri kalan bütün noktaların bu noktalara olan uzaklıklarını bul, her nokta için uzaklık merkezde seçilen noktalardan hangisine yakınsa onunla grupla, gruplanmış noktalar arasında her grup için, grupların avarage larını hesapla ve bu avarageları yeni seçilmiş noktalar olarak set et. Şeklinde adımlar içermektedir. Algoritma fixed pointlere erişince veya belirli iterasyona ulaşınca sonuçlanır.

3-) **runAlgorithm** : kendi yazdığım bu fonksiyon bütün algoritmayı süren fonksiyondur. Gray e convert edilmiş görüntü üzerinden binary threshold alır, sonra findContours fonksiyonu ile el görüntüdeki contourlar bulunur. Bu contourlar içerisinden en büyük contour seçilerek üzerinde convexHull algoritması çalıştırılır. En büyük contour ile convex hull arasındaki kırılmaları tespit edebilmek için convexityDefects fonksiyonu çağırılır. Bu kırılmaların nerelerde olduğu anladıktan sonra algoritmanın geri kalan fonksiyonları çağırır.

4-) **threshold** : Farklı parametrelerle farklı yollarla çalışabilmekle birlikte, binary threshold kendisine verilen sınır değeri altındaki intensity değerine sahip noktaları karanlık, üstündekileri ise aydınlık haline çevirmektedir.

5-) **findContours** : Bu fonksiyonun çalışması için istenilen image üzerinde önceden threshold alınmalı veya cenny edge detector ü çalıştırılmalıdır. Bu işlemler yapıldıktan sonra kapalı bir sistemde devamlı çizgiler üzerinde yürüyerek şeklin yapısını açığa çıkarır. İmage üzerinde bir biri ardına gelen tüm noktalar için çalışmaya çalışacaktır.

6-) **findBiggestContour** : findContours fonksiyonunun bulmuş olduğu contorlar içerisinde size 'ı en büyük olanının indexini döndürür.

7-) **convexHull** : Bu fonksiyon ise verilen noktalar içerisinde ilk önce 2 küme oluşturur. Her kümede bir birine en uzak olan 2 noktayı bulur ve bunların arasına bir line çizer, bu line en uzak olan 3. bir bulmaya çalışır, bulduğu takdirde burda oluşan üçgen dışında nokta varmı diye bakmak için kendini tekrar çağırır. Bu sayede bir image üzerinde belirtilen tüm noktaları içeren yapı açığa çıkmış olur.

8-)**convexityDefects** : Bu fonksiyon ise şeklin etrafını çeviren convex hull ile şeklin çevresini veren contouru alarak bu ikisi arasında bir kırılma varsa tespit etmektedir. Kırılma convex hull ile contourun ayrıldığı nokta ile, ki bu noktayı fonksiyon bize pStart olarak vermektedir, birleştikleri nokta ki bu noktayıda pEnd olarak vermektedir arası olarak söylemektedir.

9-) **calculateAngle()** : Attribute sınıfının bir fonksiyonudur. Attribute sınıfı probleme yaklaşımımız olan parmaklar arası açı olmasından dolayı bu açıyı ve açıyı oluşturan noktaları tutan bir sınıftır. Bu fonksiyon ise **convexityDefects** fonksiyonundan alınmış olan başlangıç, orta ve son noktalar arasındaki 2 line ın eğimlerini hesaplayıp bunları birbirinden çıkararak aralarındaki açıyı bulmaktadır.

10-) **findDistanceBetweenTwoPoint** : AttributeManager sınıfının bir fonksiyondur. Bu sınıf Attribute sınıfının objelerini bir listede tutar ve bunun üzerinde işlemler yapar. Bu fonksiyon ise 2 nokta arasındaki uzaklığı hesaplar.

11 -) **checkIsNear** : Elimizdeki görüntüde parmak uçlarının genişliği ve yapısının curve olması dolayısıyla bu uç noktalarda birden fazla kırılma bulunmaktadır. Birbirine çok yakın olan bu gürültüleri pozisyonlarına bakarak elemek için yazılmıştır.

12 -) **checkIsFound** : checkIsNear ın yaptığının yanı sıra bir attribute bir başlangıç ve sonuç noktasına sahip olduğu için 2 attribute arasında aslında ortak olan bir nokta bulunmaktadır. Bu sonucu handle edebilmek için yazılmıştır. Noktaların pozisyon yakınlıklarına bakılır. Yakınlık ise biggiest contour un size ına bağlı olarak değişmektedir.

13 -) **findCenterOfHand_Handful:** Moments fonksiyonunu kullanmaktadır. Bu fonksiyonla elin ağırlık merkezi bulunması amaçlanmıştır. Bu noktadan attribute lardan bildiğimiz orta noktalara (çukur noktalara) olan uzaklıkları hesaplar ve en yakın olan uzaklığı yarıçap olarak kabul ederek bir circle çizer. Bu circle'ın elin avucu olduğu kabul edilir.

14-) **eliminateAndDraw** : Bu fonksiyonda AttributeManager sınıfının bir fonksiyonudur ve elimizdeki attributelardan hangilerinin parmak belirttiğini tespit etmek için yazılmıştır. Attributeların parmak belirtmesini tespit etmek için 2 yaklaşımda bulunulmuştur. Birincisi eğer bir attribute un başlangıç, bitiş ve frab noktaları arasında oluşan açı 5 derece ve 45 derece arasında ise ve boyuda tolerans değerinden daha büyükse parmaktır demek, ikinci yöntem ise bir parmağın ucundan kendine komşu olan 2 çukur noktaya çizilecek olan line ların arasındaki açıda belirli dereceler arasında olmasıdır. Bu fonksiyon bunları tespit ettikten sonra parmak olarak belirlenen değerleri fingerList adında bir listeye eklemektedir.

15-) **FingerTrackObject** : Bu sınıf ise takip edilmek istenen parmağın koşullarının neler olması gerektiğini içermektedir. Yani bir parmağı takip etmek için önceki pozisyonu, şu anki pozisyonunu içermektedir. Bunun yanında bu objeden sadece 5 tane yaratılmaktadır. Yani her bir parmağı bu sınıf temsil etmektedir.

16 -) i**dentfiyAndNameFinger** : Bu fonskiyon Tracking sınıfnın bir fonksiyonudur. Fonksiyon **FingerTrackObject den** 5 adet üretip bunkara initial adlarını atar. Bunun ardından ilk frameden alınan parmaklar burada bu 5 objeden birine set edilir. Bunun ardından gelen framelerdeki parmak uçları buradaki initial verilmiş parmak pozisyonlarına en yakın olarak hesaplanan fingerTrackObject e kendini set eder ve bunu kullanılmış olarak işler. Ancak yeni noktanın eski bir noktaya yakın olmasını bir tölerans değerine bağlanmıştır. Bu noktadan daha uzun çıkarsa kendini set edememektedir. Bu set etme işlemi bitikten sonra eşleşemeyen veya yeni bir parmak olarak sahneye girmiş parmağın set edilebilmesi için bir döngüyle set edilmemiş objeler gezilir ve sırayla yerleştirilirler. Bir parmağın sahneden kaybolması durumunda ise oradaki bulunan pozisyon objelerden hemen silinmez. FrameCounter artırılır ve bu noktanın tekrardan alınması beklenir. Ancak 3 frame geçip kaybolan parmak hala gelmemiş ise bu pozisyon boşaltılır. Bunun nedeni ise diğer parmakların bu noktadan geçerken buradaki id yi almaya kalkışa bilir.

17 -) **blur** : openCv nin implement etmiş olduğu bir fonksiyondur. Görüntü üzerindeki gürültüyü azaltmak için kullanılır. Basit bir şekilde her pixelin komşuları ile olan avarage ını hesaplayarak tüm noktaların değerlerini günceller. Hangi komşularına bakılacağını kernel veya normalized box filter olarak adlandırılan 2 boyutlu bir matrise göre bakarak alır.

5. Programın Yapamadıkları

1-) Kullanıcı ilk 5 saniyede kamerayı fona göre ayarlarken elini koymaması veya fonu tam görmeyip etrafından büyük bir gürültüyü kameraya göstermesi durumunda threshold değeri yanlış hesaplanacaktır.

2-) Elin yumruk yapılması durumunda hesaplanacak açı olarak avuç içerisinden bir nokta seçilebilir duruma gelebiliyor. Parmaklar doğru bulunmasına karşın açı hesaplamaları yanlış olacaktır.

3-) Kameradan daha küçük boyutlarda görüntü alabilmek için videocapture sınıfından .set fonksiyonu ile widht height ayarları değiştirilmeye çalışılmıştır. Ancak bu fonksiyon her kamera için çalışamaya bilir. Kendi elimdeki kamerayla **boş fon çekiminde** fps değerim 9 fbs olmaktadır (algoritma çalıştırılmadan). Bu işlem **main boşaltılarak** yapılmıştır. Bu madde kendi bilgisayarım ve kamera için geçerlidir, farklı bir makinede, farklı kamerayla farklı sonuçlar doğuracaktır.

4-) Track edilmeye çalışılan parmaklar program içerisinde devamlı hareket yaptıkları müddetçe hep aynı idleri almalarına karşın genellikle program içerinde aynı parmaklar farklı idler almaya başlayacaktır.

6. Kullanılan Kaynaklar

[1] fingertrip detection için iskelet **,http://www.codeproject.com/Articles/782602/Beginners-guide-to-understand-Fingertips-counting , Ziyaret Tarihi [8.11.2014]**

[2] Raheja, Jagdish Lal, Karen Das, and Ankit Chaudhary. "An efficient real time method of fingertip detection." arXiv preprint arXiv:1108.0502 (2011)

[3] OpenCv cpp kmeans örneği, https://github.com/Itseez/opencv/blob/master/samples/cpp/kmeans.cpp , **Ziyaret Tarihi [8.11.2014]**