

**T.C.
SAKARYA ÜNİVERSİTESİ
BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ**

BSM 498 BİTİRME ÇALIŞMASI

**GÖRÜNTÜ İŞLEME TABANLI
KUMAŞ KALİTESİ
TEST YAZILIMI**

G151210307 – Zeynep TORUN

**Fakülte Anabilim Dalı : BİLGİSAYAR MÜHENDİSLİĞİ
Tez Danışmanı : Doç. Dr. Devrim AKGÜN**

2018-2019 Bahar Dönemi

T.C.
SAKARYA ÜNİVERSİTESİ
BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

**GÖRÜNTÜ İŞLEME TABANLI
KUMAŞ KALİTESİ
TEST YAZILIMI**

BSM 498 - BİTİRME ÇALIŞMASI

Zeynep TORUN

Fakülte Anabilim Dalı : BİLGİSAYAR MÜHENDİSLİĞİ

Bu tez .. / .. / ... tarihinde aşağıdaki jüri tarafından oybirliği / oyçokluğu ile kabul edilmiştir.

.....
Jüri Başkanı

.....
Üye

.....
Üye

ÖNSÖZ

Endüstriyel firmalarda hız ve kalite çok önemli bir konudur. Çoğu durumlarda manuel yapılan işlemlerde meydana gelen hatalar sistemdeki üretim hızını ve müşteri memnuniyetini oldukça etkiler. Bu durumla ilgili olarak manuel işlemleri azaltmak, teknolojik gelişmelere uygun şekilde düzenleme yapmak üretim kalitesini, hızını ve buna bağlı olarak maliyetini olumlu yönde etkiler. Amaç hız ve kalite iken firmalar inceleme araçları, robotlar ve görüntü işleme gibi teknolojik gelişmeleri takip ederler. Görüntü işleme teknolojisi çoğu insanın göremediği, hızla hesaplayamacağı veya dikkatsizlik-anlık dalgınlıkla kaçıracağı hesaplamalarda yüksek oranda olumlu sonuçlar çıkardığı görülmektedir.

İÇİNDEKİLER

ÖNSÖZ.....	iii
İÇİNDEKİLER.....	iv
ŞEKİLLER LİSTESİ.....	vii
TABLolar LİSTESİ.....	viii
ÖZET.....	ix
BÖLÜM 1.	
GİRİŞ.....	1
BÖLÜM 2.	
GÖRÜNTÜ İŞLEME İLE KUMAŞ TESTİ.....	2
2.1. Kumaş Testi.....	2
2.2. Kumaşta Çekmezlik Nedir?.....	3
2.3. Çekmezlik Testi Yapılışı.....	3
2.4. Çekmezlik Testinde Görüntü işleme.....	6
BÖLÜM 3.	
TEKNOLOJİLER VE YÖNTEMLER.....	8
3.1. Matlab Daire Tespiti.....	8
3.1.1. Resme Uygulanan Filtreler ve Morfolojik işlemler.....	9
3.2. Daireler Arasındaki Mesafenin Hesaplanması.....	10
3.3. Akış Diyagramı ve Matlab Kodu.....	14
3.4. Uygulamanın Gerçeklenmesi.....	16
3.5. Projenin Geliştirilmesi.....	24
BÖLÜM 4.	
SONUÇLAR VE ÖNERİLER.....	26

KAYNAKLAR.....	27
EK A.....	29
ÖZGEÇMİŞ.....	31

BSM 498 BİTİRME ÇALIŞMASI DEĞERLENDİRME VE SÖZLÜ SINAV TUTANAĞI.....	32
---	----

ŞEKİLLER LİSTESİ

Şekil 2.1.	Vaskator şablonu.....	4
Şekil 2.2.	Vaskator cetveli.....	4
Şekil 2.3.	Ölçüm işlemi temsili.....	5
Şekil 2.4.	Takip formu örneği.....	6
Şekil 2.5.	UML diyagramı.....	7
Şekil 3.1.	Örnek resim.....	10
Şekil 3.2.	Resmin algoritmaya göre ekran çıktısı.....	10
Şekil 3.3.	Daireler arasındaki sıralamanın gösterilmesi.....	11
Şekil 3.4.	Programda kullanılacak dizi şeması.....	11
Şekil 3.5.	Örnek ekran çıktısı- Testi Geçer.....	13
Şekil 3.6.	Örnek ekran çıktısı- Testten Kalır.....	13
Şekil 3.7	Matlab kodunun akış diyagramı ilk kısım.....	14
Şekil 3.8	Matlab kodunun akış diyagramı ikinci kısım.....	15
Şekil 3.9	Java kodunda dairelerin sıralanmamış hali.....	19
Şekil 3.10	Java masaüstü uygulaması.....	22
Şekil 3.11	Java masaüstü uygulaması -2.....	23
Şekil 3.12	Android Mockup görüntüsü.....	24
Şekil 3.13	Kenarlar arası ölçüm.....	25

TABLÖLAR LİSTESİ

Tablo 2.1.	Bazı kumaş test standartları.....	2
Tablo 3.1.	X değerine göre sıralamak.....	12
Tablo 3.2.	Diziyi parçalara ayırarak sıralamak.....	12

ÖZET

Anahtar kelimeler: Görüntü işleme, Matlab, Java, OpenCV, Eclipse

Görüntü işleme, görüntüyü dijital form haline getirmek ve bazı işlemleri gerçekleştirmek için geliştirilmiş, spesifik görüntü elde etmek veya ondan bazı yararlı bilgiler çıkarmak için kullanılan bir yöntemdir. Bu yöntemin girdisi video kesiti veya fotoğraf gibi bir görüntüdür. Çıktısı ise görüntünün istenilen ya da dikkat edilmesi gereken bölümüne karşılık gelir.

Bu tez çalışmasında her sektörde uygulama alanı bulan görüntü işleme tekniklerini, endüstriyel bir sorun olan kumaş testinde kullanılması konusunda çalışma yapılmıştır. Bu amaçla kumaş üzerinde yapılan çekmezlik testini ve nasıl yapıldığını anlatıp, sorunları hakkında bilgi verdikten sonra bu bilgiler doğrultusunda görüntü işleme teknikleri ile daha net sonuçlar elde eden bir sistem hazırlanmıştır. Bu sistemin algoritması, Matlab'ın yaygın kullanılan görüntü işleme komutlarıyla hazırlanıp daha sonra Eclipse platformunda Java ile OpenCV (Açık Kaynak Bilgisayarlı Görüntü Kitaplığı) kütüphanesi kullanılarak gerçekleştirilmiştir.

Bu çalışma sonucunda test işleminin tamamen dijital ortama taşınabileceği ve manuel yapılan hatalardan kaçınılabileceği görülmüştür.

BÖLÜM 1. GİRİŞ

Dokuma kumaşlar çok değişik özelliklerde ve çok çeşitli kullanım alanlarında karşımıza çıkarlar. Bu kadar farklı özelliğe ve kullanım alanına hitap eden dokuma kumaşların yapıları da birbirinden farklıdır. Kumaş yapısı, hem kumaşın özelliklerini etkilemesi açısından hem de kumaşın yüzey görünümünü belirlemesi açısından son derece önemlidir.

Kullanım amacına göre dokuma kumaşlarda bulunması istenilen bir takım özellikler vardır. Bunların arasında sağlamlık, esneklik, yumuşaklık, ısı tutma, nem çekme, dökümlülük, hava geçirgenliği, çekmezlik gibi özellikler sıralanabilir. Kumaşlarda bu istenilen özelliklerini olabilmesi için kumaşı oluşturan unsurların ve bunlar arasındaki ilişkilerin çok iyi incelenmesi gerekir.

Çekmezlik testi kumaşın yıkama öncesi ve sonrası değişen boyutlarının hesaplanmasıdır. Bu testin gerçekleştirilmesi sırasında yapılan hataları minimuma indirmek adına görüntü işleme teknikleri kullanılarak masaüstü bir program hazırlanmıştır.

Matlab kullanılarak yazılan algoritma ile gerekli hesaplama ve görüntü işleme işlemlerinin algoritması çıkartılmıştır. Bu algoritmaya göre masaüstü programı OpenCV ve Java ile gerçekleştirilmiştir.

BÖLÜM 2. GÖRÜNTÜ İŞLEME İLE KUMAŞ TESTİ

2.1. Kumaş Testi

Tekstil ürünleri, büyük ölçüde ne kadar dayanıklı olduklarını görmek için test edilirler. Buna bağlı olarak firma için bu test kalite demektir. Kumaş esneklik, dökümlülük, çekmezlik, dönme vb. özellikler kumaşın kalitesini belirler.

Dayanıklılığını ölçmek için yapılan testlerden bazıları ise şunlardır: Yırtılma dayanıklılığı, patlama mukavemeti, dikiş dayanımı, dikiş açılması ve iplik kaymasına karşı mukavemet, boncuklanma (pilling), aşınma dayanımı, hava geçirgenliği, su iticilik, buruşmazlık, statik elektriklenme ve sertlik gibi testler yapılmaktadır.

Bu testler üretim tesislerinin laboratuvarlarında hammadde girdi ve mamulleri kontrol altında tutabilmek için fiziksel uygunluğu test edilip, kalite ve kontrolleri onaylandıktan sonra müşteriye veya bir sonraki aşamada farklı bir departmana sevk edilir.

Aşağıda belirtilen testler her firmada farklılık göstermemesi açısından bazı standartlara uygun yapılır. Kalite kontrol proseslerinden geçemeyen kumaşlar için uygun olmayan ürün prosesi uygulanarak, sonraki aşamaya gönderilmesi engellenir [1].

Tablo 2.1. Bazı kumaş test standartları.

Test Adı	Standart	Test Amacı
Boyut Değişimi (Dimensionel Change)	ISO 6330	Kumaşlarda yıkamadan sonraki boyut değişimi kontrol edilir. Yıkabilir kumaşlarda, kılıflarda vs. talep edilir.
Boncuklanma (Pilling)	ISO12945-2	Kumaşın zamanla boncuklanma eğiliminin tespiti yapılır.

İplik Sıklık Tayini (Determination of number of threads)	ISO 7211-2	Kumaşta kullanılan atkı ve çözgü iplik sayılarının tespiti yapılır.
pH Tayini (pH Value)	ISO 3071	Kumaşların pH seviyesinin tespiti yapılır.

Tabloda belirtilen standartlara göre kumaşlara testler uygulanır. Bu testler arasında bizim üzerinde çalışma yapacağımız test boyutsal değişim olan çekmezlik testidir. Çekmezlik testinin nasıl yapıldığını öğrendikten sonra geliştirme aşamasına geçilmektedir.

2.2. Kumaşta Çekmezlik nedir?

Çekme testinin amacı kumaşın kesim ya da dikimden sonra oluşabilecek çekmelerinin en baştan önüne geçmek için alınacak önlemleri sağlamaktır. Çekmesi bozuk olan kumaş derhal terbiye işlemlerinin yapıldığı işletmeye gönderilir ve çekmesinin sağlanması istenir. Çekme testi yapılmadan kesilen ya da dikilen kumaş olması gereken beden ölçülerinden daima farklı sonuçların çıkmasına neden olur. Mesela yeni alınan bir giysinin ilk yıkamadan sonra daralması ya da bollaşması çekmezlik denen işlemin sağlıklı yapılmamasından kaynaklanmaktadır [2].

2.3. Çekmezlik Testi Yapılışı

Her gün işletmeden gelen kumaş parçalarının, kumaş örneği takip formunda verilen bilgileri okunur. Firma elemanlarına verilen takip formunda (Şekil 2.4.) müşteri ismi, parti numarası, kumaş kalitesi, renk, kumaş giriş eni, tüm en, besleme değeri bulunmaktadır. Bu bilgiler ile test sonucu yorumlanır.

Kumaş üzerine konulan şablonun (Şekil 2.1) 35x35 cm olan çevresine ve içindeki noktalara boyalı bir kalem ile işaretlenir. Daha sonra kumaş çevresi işaretlenen yerden taşarak kumaş kesilir. Kesilen kenarlar dağılmaması amacıyla makinada dikilir ve kumaşlar yıkama makinesine gönderilir. [3]



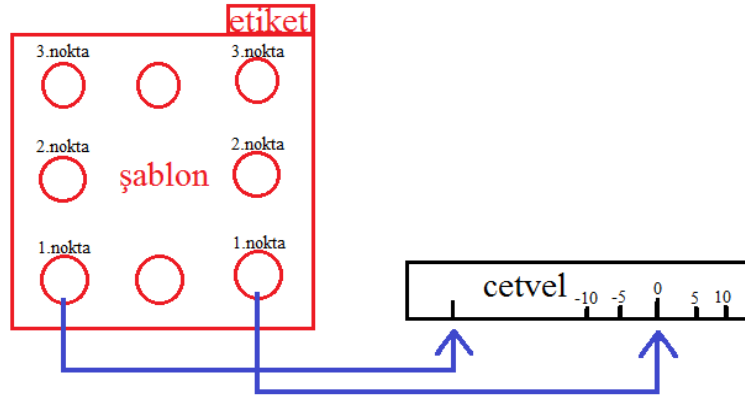
Şekil 2.1. Vaskator şablonu

Yıkamaya gitmeden önce kumaşlardaki numaralandırılmış noktalar ile onların hizasındaki (en sağdaki) noktalar arası ölçüldüğünde sıfıra sıfırdır. Yıkama işleminden sonra +12 saat beklemeye alınan kumaş kurutma sonunda vaskator cetveli (Şekil 2.2.) ile tekrar ölçülür. Ne kadar sapma olduğu cetveldeki değerden okunarak kayıtlara geçer.



Şekil 2.2. Vaskator cetveli

Takip formunda belirtilen birinci nokta için kumaşın soluna yaslı halde vaskator cetveli (Şekil 1.2) düz bir şekilde konulur, sağ nokta cetvelin 0 noktasında ise kumaş sağlamdır. -15 - 0 arasında ise çekmiş, +15 - 0 arasında ise genişlemiştir.



Şekil 2.3. Ölçüm işlemi temsili

Bu işlem etiket sağ üstte olduğunda **atkı** değerini, kumaştaki etiket sağ yanında olduğunda (kumaş 90 derece sağa döndürüldüğünde) **çözü** değerlerini verir. Her ikisi için de 3'er noktadan ölçüm yapılarak takip formundaki yerlerine yazılır.

Atkı ve çözgü değerlerine göre mutlak değerin 3'den büyük olması kumaşın testi geçemediğini gösterir (Bazı kumaş türleri için bu değer değişebilir). Eğer kumaş yıkamadan çıktığında çok çekmişse (+15'den büyük veya -15'den küçükse) her bir nokta için *demir cetvel* kullanılarak milimetrik ölçüm yapılır ve şu formül kullanılır: (Denklem 2.1)

$$350 - (\text{ölçülen değer}) = \text{Sonuc}$$

$$\left(\frac{\text{sonuç}}{350} \right) \times 100 = X. \text{ nokta değeri} \quad (2.1)$$

Yıkama Sonrası Boyutsal Değişim - Örne (ISO 6330:2012,ISO3759:2011)

Bölüm Adı	Örne Testleri	Test Tanımı	Uygulama Yeri
Test Talep No		1 Gramaj - Örne (ISO 3801:1977)	
Uygulama		2 Dönme- Örne (B3-01/TM6)	
Talep Tipi	Ön Üretim Bileşen	3 Yıkama Sonrası Boyutsal Değişim - Örne (ISO 6330:2012,ISO3759:2011)	
Termin Tarihi	12.09.2018	4 Patlama(ISO 13938-2:1999)	
Gramaj	150,00	5 Boncuklanma(ISO 12945-1:2000)	
Değişken	40°C 4N TK		
Performans Kodu	H TRH02 F TOFT03		
Yaş Grubu	YENİDOĞAN		
Elyaf Karşımı	%100 Pamuk		
Açıklama	PINK - 16166- 1.1		

Bileşen Kodu	Kullanım Yeri	Ek Bilgi	Barkod	Miktarı	Açıklama	Sonuç Değeri
Bileşen Adı Jersey , 30/1 Combed , 150.0 gsm , 28 Fine , 100% Cotton , Solid Dyed , Reactive Dyed , Silicone, Enzyme						
	ANA BEDEN					

() Kondisyonuz test edildi

Yıkama Öncesi Kondisyon Giriş Tarih / Saat	12 Eylül 2018 18:00	Yıkama Sonrası Kondisyon Giriş Tarih / Saat	13 Eylül 2018 12:50			
Yıkama Sonrası Ölçüm Sonuçları						
	1.Nokta %	2.Nokta %	3.Nokta %	ORTALAMA %	Mutlak Değer	Testi Yapan İmza/Tarih
ATKI	-6.0	-5.0	-5.0	-5.3	1.8	13 Eylül 2018
ÇÖZGÜ	-3.5	-3.0	-4.0	-3.5		27

Testi Hazırlayan İmza / Tarih	Onaylayan İmza / Tarih
12 Eylül 2018 SMK	13 Eylül 2018

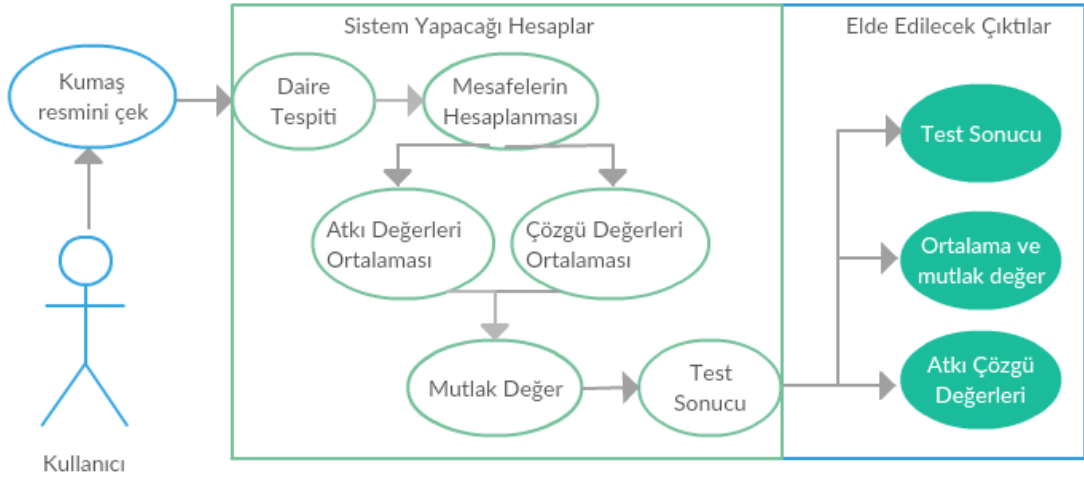
Şekil 2.4. Takip formu örneği

2.4. Çekmezlik Testinde Görüntü İşleme

Anlatılan test adımları sırasında manuel işlemlerde yapılan hatalar nedeniyle testten sağlıklı bir sonuç alınamaz. Yıkama sonrası kumaş üzerinde şablon ile konulmuş noktalar görüntü işleme teknikleri kullanılarak yazılan program ile bulunduktan sonra manuel testte olduğu gibi noktalar arası mesafe hesaplatılması, bu mesafe ölçümlerine göre yapılan kumaşın geçer ve ya kalır sonucu elde edilmesi sağlanmıştır. Hesaplanan değerlere göre çekmezlik hesaplamaları takip formuna doğru ölçümler olarak yazılıp bir sonraki adıma kolaylıkla geçilebilecektir.

Şablon üzerindeki noktaları bulmak amacıyla uygun algoritmayı yazmak ve işleyişi görmek için öncelikle Octave kullanılmıştır. Octave, Matlab benzeri olan open source ücretsiz GPL lisanslı program olup, kullanımı daha kolaydır. Fakat kaynakların yeterli olmaması nedeniyle Octave'dan vazgeçilmiştir. Matlab'ın görüntü işleme konusunda daha fazla kaynağının olması ve daha yaygın olması sebebiyle tercih edilmiştir.

Kumaş testi için yazılacak olan algoritmanın UML diyagramı Şekil 2.5.'de gösterilmiştir.



Şekil 2.5. UML Diyagramı

BÖLÜM 3. TEKNOLOJİLER VE YÖNTEMLER

3.1. Matlab ile Daire Tespiti

Resimdeki dairelerin algılanması işlemlerini sırasıyla anlatmak gerekirse öncelikle cisimlerin ayırt edilebilmesi için resmi tek renk haline getirip üzerindeki gürültüden (anlamsız-küçük pixellerden) arındırmak gerekir. Resim üzerinde düzeltmeler yapılarak daha net şekiller elde edilir.

Daha sonra for döngüsüyle matristeki her daire elemanının sınırlarını çizer. Döngü, birinci daireden başlayarak `length(B)` değerine (sıra ve sütun arasından büyük olan sayıya) kadar döner.

Son işlem ise hangi nesnelerin yuvarlak olduğunu belirlemektir. Nesnelerin ağırlık merkezlerine göre alan ve çevre hesaplamaları yapılır. Bu hesaplamalarla hangi orana sahip olduğu nesnenin üstüne yazılmıştır.

```
RGB = imread('C:\Users\ZEYNEPTORUN\Desktop\dairesetpit.png' );

%resme filtre uygulamak ve gürültülerden temizlemek;
I = rgb2gray(RGB);
bw = im2bw(I);
imshow(bw)
bw = bwareaopen(bw,50);
se = strel('disk',2);
bw = imclose(bw,se);
bw = imfill(bw, 'holes');
imshow(bw);

%daire sınırlarını bul
[B,L] = bwboundaries(bw, 'holes');
imshow(label2rgb(L, @jet, [.5 .5 .5]))

hold on

for k = 1:length(B)
    boundary = B{k};
    plot(boundary(:,2), boundary(:,1), 'w', 'LineWidth', 2)
end
...
```



```

nesne_verileri = regionprops(L, 'Area', 'Centroid', 'Perimeter' );
threshold = 0.85; %Daire sayılabilmesi için ağırlık merkezi eşiği
total=0; %daire sayısı sayacı
for k = 1:length(B) %her bir daire için;
    boundary = B{k}; %Boundary; k. dairesinin x,y sınır
    koordinatlarını tutan 2 boyutlu matrisi tutar
    delta_sq = diff(boundary).^2;%delta_sq değişkenine dairenin
    tahmini çevresini hesaplayıp atadık
    perimeter = sum(sqrt(sum(delta_sq,2)));%çevre = toplam(kök içinde
    (toplam delta_sq,5))
    area = nesne_verileri(k).Area; %k. daireye karşılık gelen alan
    hesaplaması
    metric = 4*pi*area/perimeter^2;
    metric_string = sprintf('%2.2f',metric);%sonuçları değişkende
    tutmak
    if metric > threshold %eğer metrik hesap büyükse ağırlık
    merkezi eşiğinden dairedir
        centroid = nesne_verileri(k).Centroid;
        plot(centroid(1),centroid(2),'ko');
        total=total+1;
    end
    text(boundary(1,2)-35,boundary(1,1)+13,metric_string,
        'Color','g','FontSize',14,'FontWeight','bold');
End
title(['Daire olmaya en yakın cisimler', num2str(total)]);

```

3.1.1. Resme Uygulanan Filtreler ve Morfolojik işlemler

rgb2gray() : Gri filtreleme işlemidir.

im2bw() : Siyah-beyaz (bitwise) dönüşümü, beyaz(1) ile siyah(0) olarak yeniden çizer. (Matlab 2016 sürümündeki adı imbinarize)

bwareaopen(bw,30) 30 pixele kadar oluşan gürültüyü kadar gidermek için.

strel('disk',2): Fonksiyonu nesnenin dairesel yapısını korumak için kullanılan 2pixel yarıçapında yapılandırma elemanıdır.

imclose(): Görüntü üzerinde morfolojik bir kapatma işlemi gerçekleştirir.

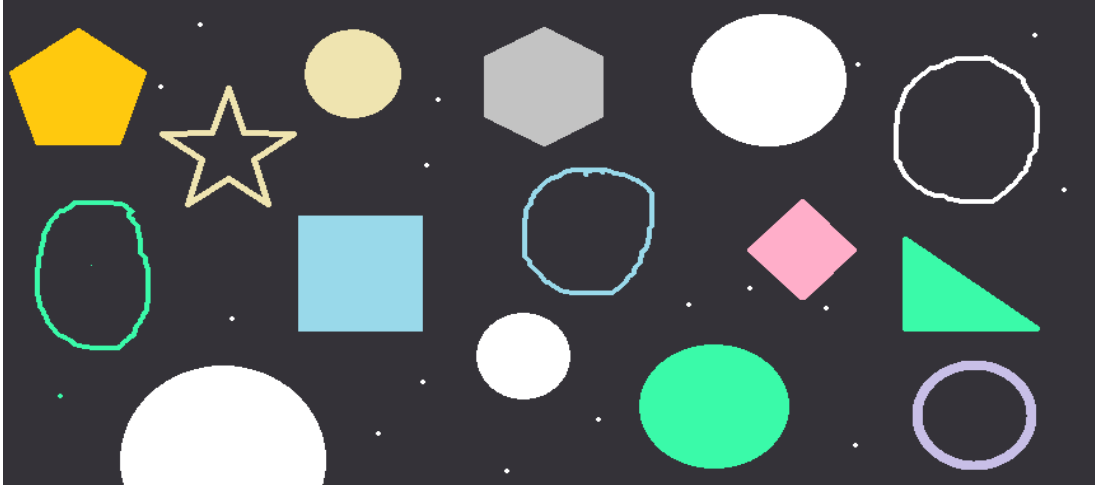
imfill(bw,'holes'): metodu ile yapılandırma işlemi sırasında oluşan boşlukları doldurur.

bwboundaries(): Fonksiyonu resimdeki nesneleri bulmak için kullanılır. Nesnelerin dış sınırlarını ve bu nesneler içindeki deliklerin sınırlarını ikili görüntüde izler. Aldığı iki parametreden biri pixel sınır konumlarını içeren b, diğeri nesne içindeki deliklerin sınırları hücre dizisidir. Parametre olarak holes değerinin alınması, nesnelerin içindeki sınırların dahil edilmek istendiği ya da noholes değerinin alınması, dahil edilmek istenmediği anlamına gelir.

label2rgb(): Resmi renk paleti şeklinde renklendirmeyi sağlar.

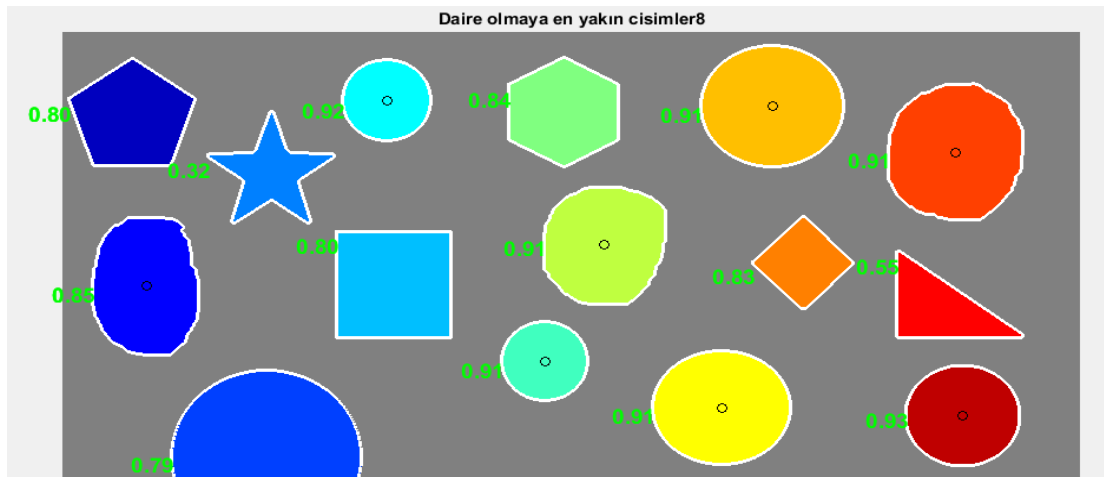
regionprops(L, 'Area', 'Centroid', 'Perimeter'): Sırasıyla ilk parametre nesne, sonraki alanda bulunan piksel sayısı, bölgenin kütle merkezi, bölge sınırı etrafındaki mesafeyi döndürerek kısmen çevreyi hesaplar.

plot() (nesne(:,2), nesne(:,1), çerçeve rengi, hat genişliği, çerçeve çizgi boyutu) belirtilen parametrelere göre nesneye çerçeve çizer [4]-[9].



Şekil 3.1. Örnek resim

Örnek resim (Şekil 3.1) üzerinden kod ekran çıktısını gösterilmiştir. (Şekil 3.2)



Şekil 3.2. Resmin algoritmaya göre ekran çıktısı

3.2. Noktalar Arasında Mesafenin Hesaplanması

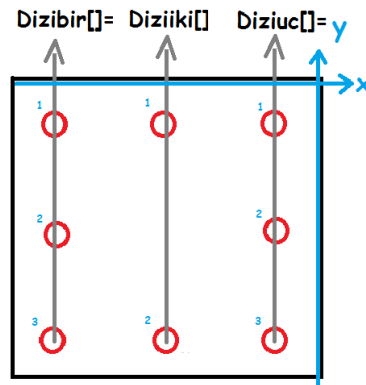
Daire tespiti yaparken bulunan ağırlık merkezlerinden x ve y koordinatları bulunur. Birinci dairenin x koordinatı ile ikinci noktanın x koordinatları arasındaki fark iki dairenin birbiri arasındaki mesafeyi verir.

Sekiz daire için bu işlem yapılmak istenirse, koordinatlar bir diziye atılıp doğru fark elde edilebilmesi için en küçük x ve y koordinat değerine göre sıralama yapılması gerekir. Bu işlemi yaparken Matlab yazma işlemine en küçük x koordinatını yazarak başlamaktadır (Şekil 3.3’de mavi yazılı sıralama ile gösterilmiştir). Yani birinci noktadan bahsedildiğinde sürekli aynı yerdeki daireyi vermesi gerekirken, ortanca veya en alt noktanın x’e olan yakınlığı yüzünden sıralama her resimde değişmektedir. Bizim ihtiyacımız olan ise x ve y’si en küçük olan daireyi bulmaktır. (Şekil 3.3’ de kırmızı yazılı olan sıralama ile olması gereken gösterilmiştir.)



Şekil 3.3. Daireler arasındaki sıralamanın gösterilmesi

Daireler x koordinatlarına göre sıralanmışken diziyi 3, 2, 3 elemanlı sayılara ayırarak yeni diziler elde edip, y koordinatlarına göre sıralayarak hesaplanmıştır. Bu işlemi 8 noktaya uygulamak istendiğinde şu şema göre göz önüne alınabilir;



Şekil 3.4. Programda kullanılacak dizinin şeması

Her biri dizi iki boyutlu olup dairelerin x ye koordinatlarını içermektedir. Her dizi grubu da kendi içinde y koordinatlarına göre sıralandığında örnek rakamlarla şu şekilde doğru sıralanış elde edildiği gösterilebilir: Soldaki tablo (Tablo 3.1.) parçalama işleminden önce x koordinatlarına göre oluşturulan dizi sıralaması iken, sağdaki tablo (Tablo 3.2.) parçalara ayrıldıktan sonraki halinin sıralamasıdır.

Tablo 3.1. X değerine göre sıralamak

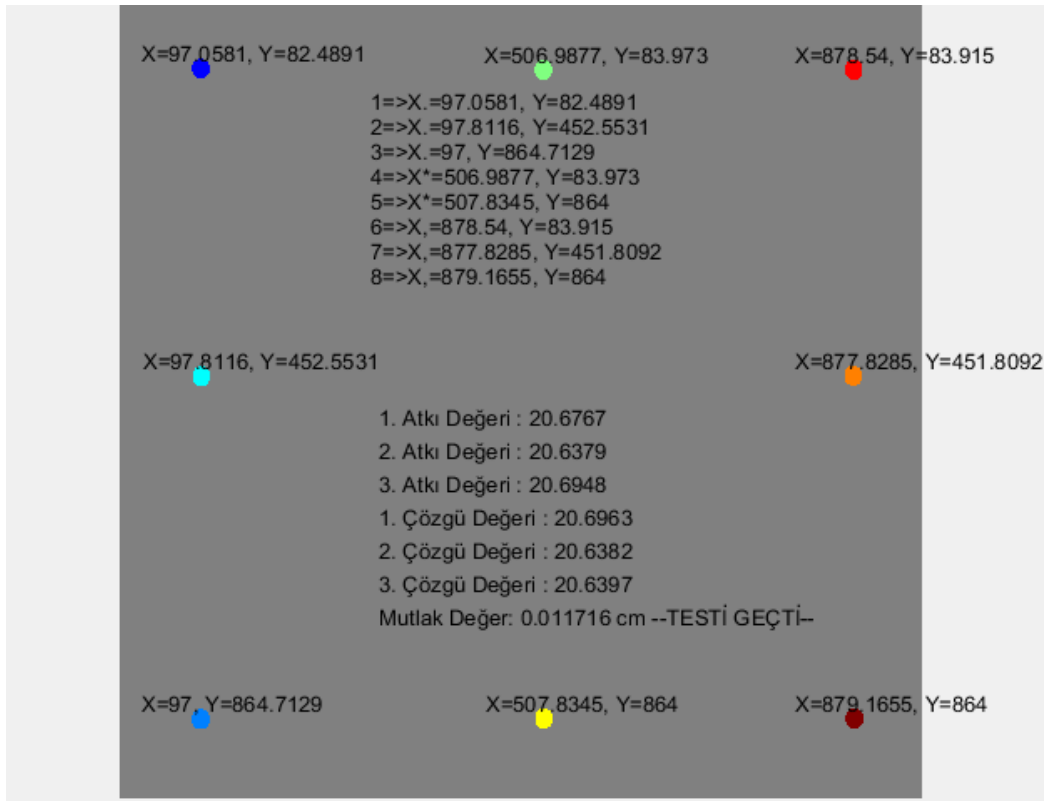
Sıra	X Koordinat	Ykoordinat
1	97,0581	82,4891
2	97	864,7129
3	97,8116	452,5531
4	506,9877	83,973
5	507,8345	864
6	877,8285	451,8092
7	878,54	83,915
8	879,1655	864

Tablo 3.2. Diziyi parçalara ayırarak sıralamak

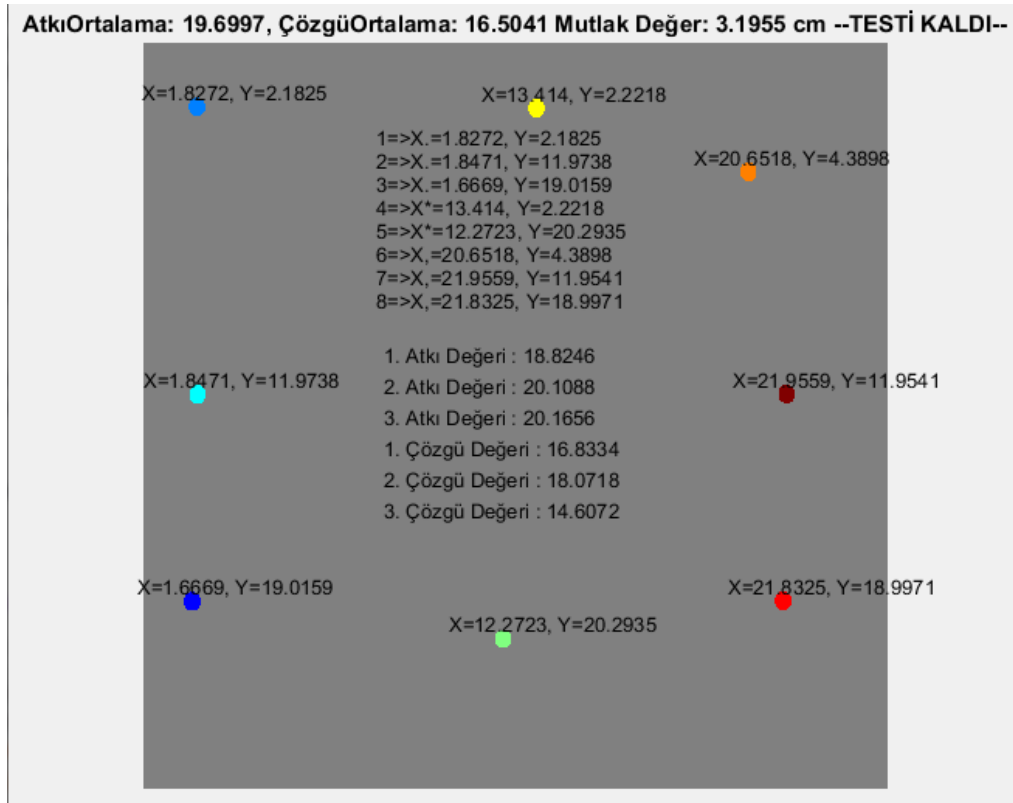
Sıra	X Koordinat	Ykoordinat
1	97,0581	82,4891
2	97,8116	452,7129
3	97	864,5531
4	506,9877	83,973
5	507,8345	864
6	878,54	83,915
7	877,8285	451,8092
8	879,1655	864

Bu elde edilen değerler sonucunda atkı ve çözgü olarak isimlendirilen değerler bulunur. Gerçek test yapılırken atkı değerleri en sağdaki üç noktadan kendilerine karşılık gelen en soldaki üç noktanın çıkartılmasıyla elde edilir, çözgü değerleri ise 90 derece sağa döndürülen kumaşın yine en sağdaki üç noktadan en soldaki üç noktanın çıkarılması işlemi uygulanarak bulunur. Bu işlemler koda dökülürken; atkı değerleri için, “diziuc” isimli diziden “dizibir” isimli dizinin birbirlerine karşılık gelen x değerleri çıkartılırken, çözgü değerleri için, her dizinin son elemanının y değerinden, ilk elemanının y değeri çıkartılarak sonuc bulunur. Bu farklar piksel-cm dönüşümü yapılarak ekrana yazdırılır. (1piksel = 0,02645833 cm)

Test formunda yer alan atkı ve çözgü değerlerinin ortalamaları ve birbirlerinden çıkartılarak elde edilen mutlak değer işlemi sonucunda eğer 3 cm’den küçük bir rakam elde edilmişse kumaş testi geçmiştir demektir, değilse kumaş testi geçemez.

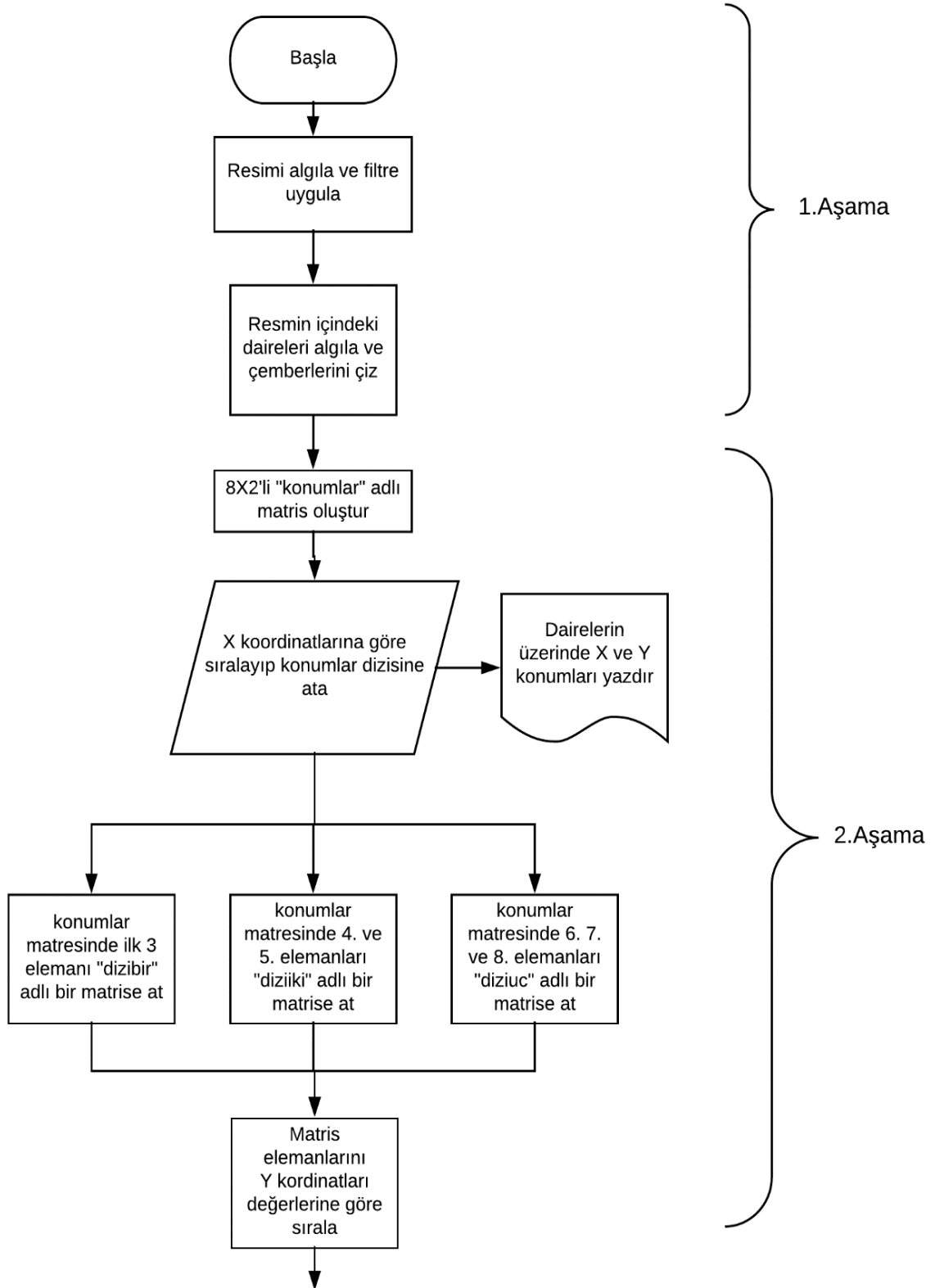


Şekil 3.5. Örnek ekran çıktısı- Testi Geçer

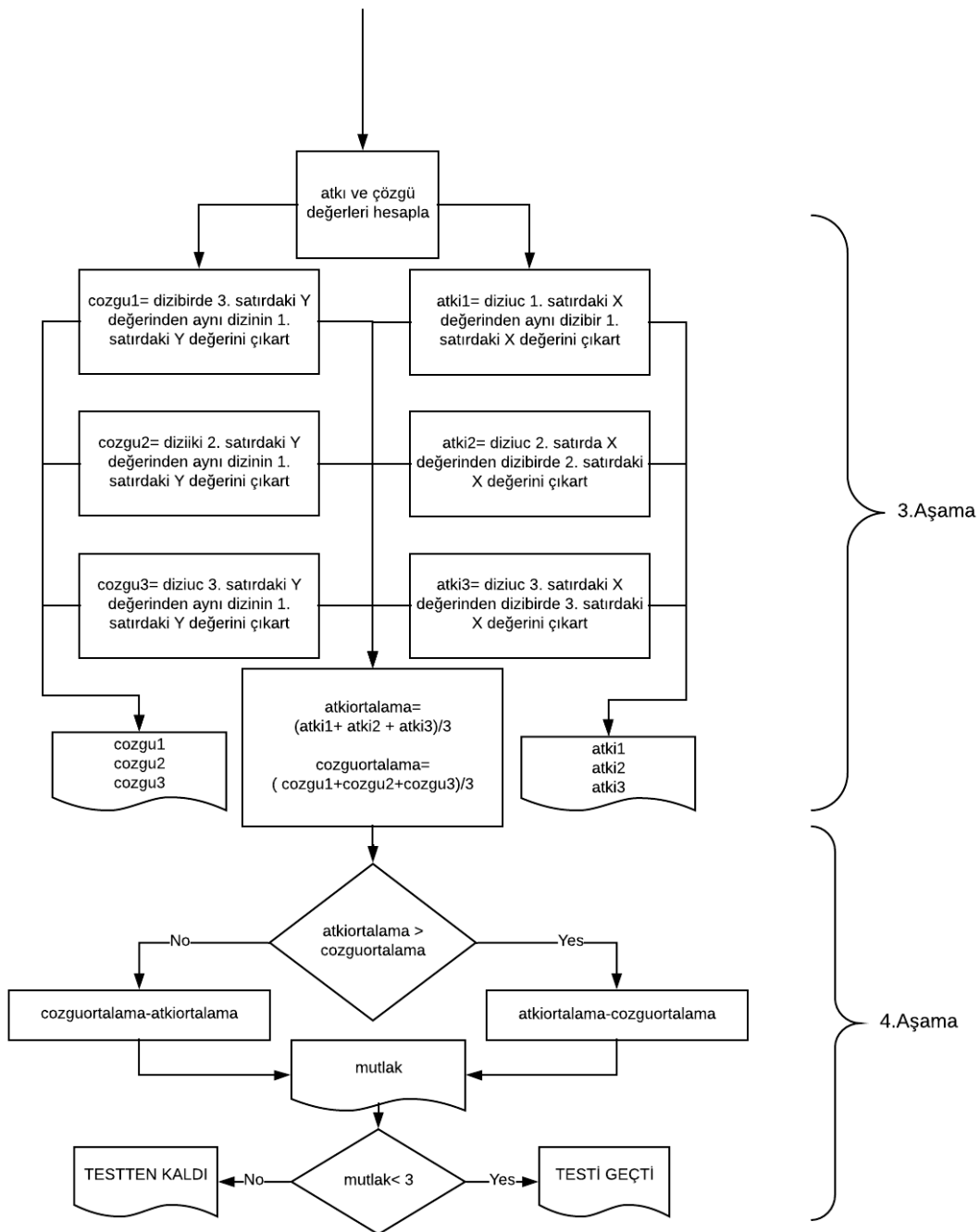


Şekil 3.6. Örnek ekran çıktısı- Testten Kalır

3.3. Akış Diyagramı ve Matlab kodu



Şekil 3.7. Matlab kodunun akış diyagramı -ilk kısım



Şekil 3.8. Matlab kodunun akış diyagramı ikinci kısım

1. Aşama:

Program çalıştırılır ve kullancıdan resim alınır. Alınan resim üzerindeki daireleri tespit etmek ve merkez koordinatlarını daha doğru bir şekilde bulunması amacıyla filtre ve morfolojik işlemler (gri filtre, bitwise, imclose, strel, imfill vs.) uygulanıp daireler bulunur. Dairlerin etrafı çizilip daha belirgin hale getirilir.

2. Aşama:

Kullancıdan alınan her resimde dairelerin konumları belli bir sıralaması olması gerekmektedir. Resimde bulunan dairelerin merkez koordinatlarını tespit edip hem X hem Y koordinatı en küçük olandan başlayarak sıralama yapılmalıdır.

8x2 “konumlar” matrisi oluşturulduğunda X eksenine göre Matlab’de sıralı haldedir. Y koordinatı birbirine yakın olan matris elemanlarının sıralarının sabit olmasından emin olmak için Y koordinatına göre de sıralama yapılmalıdır. Fakat X’e göre sıralamayı bozmamak için “konumlar” matrisin ilk üç elemanı “dizibir”, dördüncü ve beşinci elemanları “diziiki”, ve son üç elemanları da “diziuc” matrislerine atılır. Bu üç matris için Y koordinat değerlerine göre sıralama yaparak istenilen sıralama elde edilir.

3. Aşama:

Oluşturulan dizibir, diziiki, diziuc matrislerin aracılığıyla yatayda en sol ve en sağdaki dairelerin X koordinatların farklarını hesaplatarak atkı değerleri, dikeyde en üst ve en alttaki dairelerin Y koordinatların farklarını hesaplatarak çözgü değerleri hesaplanır. Ekranı yazdıktan sonra atkı ve çözgü değerlerini toplayıp 3’e bölerek ortalaması alınır.

4. Aşama:

Atkı ve çözgü değerlerinin ortalamalarından büyük olandan küçük olanı çıkartılıp mutlak değeri elde edilir. Mutlak değeri 3’ten küçük ise kumaş testten geçer büyük ise testten kalır. Testten kalan kumaşın kullanılmasına izin verilmez.

Uygulama Matlab kodu; EK A’da yer almaktadır.

3.4. Uygulamanın Gerçeklenmesi

OpenCV java ya import edilmesi ile birlikte form ekranı oluşturabilmek için swing kütüphaneleri de eklenir. Bu işlemlerden sonra gerekli elemanlar ile form dizaynı oluşturulmuştur. (Uygulama Java kodu EK B’de yer almaktadır.)

Uygulama içinde kullanılan Opencv sınıfları:

Mat nesnesi: vektör ve matrisleri, görüntüleri, hacimlerini, vektör alanlarını depolamak için kullanılır.

Imgcodecs sınıfı: görüntüleri okumak ve yazmak için kullanılan sınıftır.

“**Imread()**, **Imwrite()**” mat nesnelerini okumak ve yazmak için kullanılır

Core sınıfı: OpenCV kütüphanesinin fonksiyon ve sınıflarının yüklenmesi için gereken temel sınıf

Point sınıfı: x, y koordinatları ve merkez eşiğini kullanabileceğimiz bir nesne oluşturulmasını sağlar.

Scalar sınıfı: renk ataması yapılmasını sağlayan sınıftır.

Imgproc sınıfı: Görüntü üzerinde kullanılacak morfolojik işlemleri barındırır.

“**cvtColor()**” fonksiyonu sırasıyla kaynak, hedef mat nesneleri ve filtre şeklinde parametre olarak renk dönüşümü sağlar.

“**medianBlur()**” kaynak mat nesnesini ve hedef mat nesnesine belirtilen sayısal değere göre bulanıklaştırır.

“**HoughCircles()**” aldığı parametrelerle Hough dönüşümü kullanarak gri tonlamalı bir görüntü üzerindeki daireleri bulur.

“**circle()**” nesne üzerine aldığı parametrelere göre daire çizdirir. Parametreleri sırasıyla, hedef ve kaynak mat nesnesi, nokta merkezi, yarıçap, renk, kalınlık, çember sınır tipi, merkez koordinatlarda kesirli bit sayısı ve yarıçap değeri.

“**putText()**” görüntü üzerine metin eklenmesini sağlar. Aldığı parametreler sırasıyla mat nesnesi, metin, yazılacak koordinat, font tipi, font boyutu, renk ve kalınlık. [10]-[14]

```
System.LoadLibrary(Core.NATIVE_LIBRARY_NAME );

String dosyaYolu = txtResimYolu.getText();

Mat src = Imgcodecs.imread(dosyaYolu, Imgcodecs.IMREAD_COLOR);
Mat gray = new Mat();
Imgproc.cvtColor(src, gray, Imgproc.COLOR_BGR2GRAY);
Imgproc.medianBlur(gray, gray, 5);
Mat circles = new Mat();
Imgproc.HoughCircles(gray, circles, Imgproc.HOUGH_GRADIENT ,1,
(double)gray.rows() /8, 200.0, 11.0, 0, 0);
```

Seçilen dosya yolundan alınan resmi “src” kaynak mat nesnesine yüklenir. Üzerinde değişiklik yapılacak olan “gray” hedef mat nesnesidir. “COLOR_BGR2GRAY” filtresi ile resme gri dönüşüm uygulanır. Daha sonra bulanıklaştırılarak gürültülerden arındırılır. Filtrelenen ve temizlenen görüntüde daire tespiti HoughCircles fonksiyonu ile yapılır. Bu fonksiyon sırasıyla hedef ve kaynak mat nesneleri, gradyan, çözünürlük ters oranı, algılanan merkezler arası minimum mesafe, kenarların belirlenmesi için üst eşik, merkez tespiti için eşik merkezi, minimum ve maximum yarıçap parametreleri alır.

Resmin üzerinde detection edilmiş dairelerin merkezleri ve kenarlarının işaretlenir ayrıca işlem kolaylığı açısından üzerlerine koordinat ve daire numaraları yazılır. Aynı döngü içinde her dairenin x ve y koordinatları sıralama yapılacağı zaman kullanılmak üzere 8x2 ‘lik diziye atılır. X sıralamasında kullanılmak üzere “sayı” dizisi oluşturulur.

```
double[][] konumlar = new double[8][2];
double[] sayi = new double[8];

//resim üzerinde daireleri çiz
for (int x = 0; x < circles.cols(); x++) {

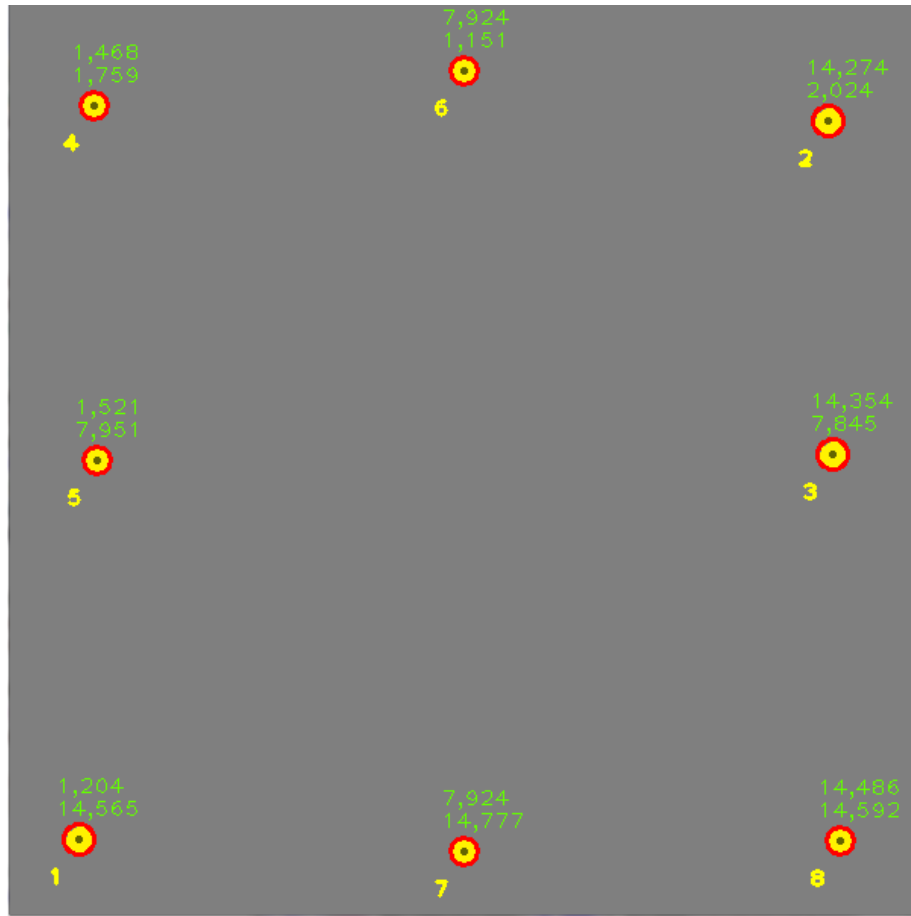
    double[] c = circles.get(0, x);
    Point center = new Point(c[0], c[1]); // daire merkezini işaretleme
    Imgproc.circle(src, center, 1, new Scalar(0,100,100), 2, 8, 0);
    int radius = (int) Math.round(c[2]); // daire dış çevresini işaretleme
    Imgproc.circle(src, center, radius, new Scalar(0,0,255), 2, 8, 0);

    //dizilere koordinatları at
    double cm=0.02645833;
    konumlar[x][0] = sayi[x]= center.x*cm;
    konumlar[x][1] = center.y*cm;

    String konumx =new DecimalFormat("#.###").format(konumlar[x][0]);
    String konumy = new DecimalFormat("#.###").format(konumlar[x][1]);

    //x ve y koordinatlarını merkez noktanın üstünde yaz
    Imgproc.putText(src, konumx, new Point((center.x-15),(center.y-30)),
        Imgproc.FONT_HERSHEY_PLAIN, 1, new Scalar(0,255,100), 1);
    Imgproc.putText(src, konumy, new Point((center.x-15),(center.y-15)),
        Imgproc.FONT_HERSHEY_PLAIN, 1, new Scalar(0,255,100), 1);

    //daire numarası
    Imgproc.putText(src,String.valueOf(x), new Point((center.x-20),
        (center.y+30)), Imgproc.FONT_HERSHEY_PLAIN, 1, new Scalar(0,255,255), 2);
}
```



Şekil 3.9. Java kadunda, dairelerin henüz sıralanmamış hali

Matlab algoritması hazırlanırken de belirtildiği gibi dairelerin Şekil 3.9 da olduğu karışık bir numaraya sahip olması istenmeyen bir durumdur. Hesaplama yapılırken her resimde daire numaralandırma sisteminin aynı olması beklenmektedir.

```
Arrays.sort(sayi); //sıralanmış x koordinatları
//sıralı koordinatları tutacak olan iki boyutlu dizi
double[][]sirali = new double[8][2];
for (int i = 0; i < 8; i++) {
    for (int j = 0; j < 8; j++) {
        if(sayi[i]==konumlar[j][0]) {
            sirali[i][0]=konumlar[j][0];
            sirali[i][1]=konumlar[j][1];
        }
    }
}
```

Matlab’de yazılan algorithmada daireler algılama sırasında x koordinatlarına göre küçükten büyüğe sıralanmıştı. Gerçekleme sırasında ise x koordinatlarına göre sıralama işlemi için Arrays.sort() fonksiyonu kullanılmıştır. Bu fonksiyon tek

boyutlu dizide çalıştığı için x koordinatları “sayı” dizisine atılıp konumlar iki boyutlu matrisindeki x koordinatları ile karşılaştırılmış ve “sıralı” adındaki iki boyutlu matriste sıralı hali tutulmuştur.

Y koordinat sıralaması için dizi üç parçaya bölünmesi yine Matlab algoritmasında karar verilen bir durumdur. Bu dizi yapısı Şekil 3.4’de belirtildiği gibidir. Her bir dizi daha sonra kendi aralarında küçükten büyüğe sıralanırlar. Bu sıralamaya bağlı olarak “sıralı” adlı iki boyutlu dizide bahsedilen y koordinatına karşılık gelen dizi elemanı konumlar adlı dizide taşınır. Konumlar dizisinin son halide hem x hem de y koordinatları sıralı olmuş olur.

```
double[] dizibir =new double[3];
double[] diziiki =new double[2];
double[] diziuc = new double[3];
//sıralı dizinin sütun elemanları bu dizilere dağıtıldı
for(int i=0; i < sirali.length ;i++) {
    if (i < 3)
        dizibir[i] = sirali[i][1];
    else if(i==3 || i==4)
        diziiki[i-3] = sirali[i][1];
    else
        diziuc[i-5] = sirali[i][1];
}
// Y koordinat sıralaması
Arrays.sort(dizibir); Arrays.sort(diziiki); Arrays.sort(diziuc);
for (int i = 0; i < 8; i++) { // kontrol
    for (int j = 0; j < 8; j++) {
        if (i<=2)
        {
            if(dizibir[i]==sirali[j][1]) {
                konumlar[i][0]=sirali[j][0];
                konumlar[i][1]=sirali[j][1];
            }
        }
        else if (i==3 || i==4)
        {
            if(diziiki[0]==sirali[i][1]) {
                konumlar[i][0]=sirali[i][0];
                konumlar[i][1]=sirali[i][1];}
            else
            {konumlar[i][0]=sirali[i][0];
                konumlar[i][1]=sirali[i][1];
            }
        }
        else if(i>=5) {
            if(diziuc[i-5]==sirali[j][1]) {
                konumlar[i][0]=sirali[j][0];
                konumlar[i][1]=sirali[j][1];
            }
        }
    }
}
```

Bu sıralama ile birlikte form ekranında gerekli text alanlarına koordinat değerleri yerleştirilmiştir. Atkı ve çözgü değerleri yine algoritma da belirlendiği gibi sağdaki noktaların kendilerine karşılık gelen noktadan çıkarılmasıyla bulunur. Mutlak değer atkı ve çözgü değerleri arasındaki farktır. Eğer 3cm den büyük veya eşitse kumaş testten kalır. Bu sonuç yine form ekranında gösterilmiştir. [15]-[17]

```
double atki1=(konumlar[5][0]-konumlar[0][0]);
double atki2=(konumlar[6][0]-konumlar[1][0]);
double atki3=(konumlar[7][0]-konumlar[2][0]);
double cozgu1=(konumlar[2][1]-konumlar[0][1]);
double cozgu2=(konumlar[4][1]-konumlar[3][1]);
double cozgu3=(konumlar[7][1]-konumlar[5][1]);

txtAtki1.setText(String.valueOf(new
DecimalFormat("##.###").format(atki1)));
txtAtki2.setText(String.valueOf(new
DecimalFormat("##.###").format(atki2)));
txtAtki3.setText(String.valueOf(new
DecimalFormat("##.###").format(atki3)));
txtCozgu1.setText(String.valueOf(new
DecimalFormat("##.###").format(cozgu1)));
txtCozgu2.setText(String.valueOf(new
DecimalFormat("##.###").format(cozgu2)));
txtCozgu3.setText(String.valueOf(new
DecimalFormat("##.###").format(cozgu3)));
double aOrt= (atki1+atki2+atki3)/3;
double cOrt= (cozgu1+cozgu2+cozgu3)/3;
txtAtkiOrtalama.setText(String.valueOf(new
DecimalFormat("##.###").format(aOrt)));
txtCozguOrtalama.setText(String.valueOf(new
DecimalFormat("##.###").format(cOrt)));

double mutlakdeger = Math.abs(aOrt-cOrt);
txtMutlakDeger.setText(String.valueOf(new
DecimalFormat("##.###").format(mutlakdeger)));
if(mutlakdeger>=3)
{
    lblSonuc.setText("Kumaş Testten Kaldı.");
    txtAreaSonuc.setText("Mutlak değerin 3cm'den büyük\n
    olması sebebiyle kumaş testi\n geçememiştir.");
}
else
{
    lblSonuc.setText("Kumaş Testti Geçti.");
    txtAreaSonuc.setText("Mutlak değerin 3cm'den küçük\nolması
    sebebiyle kumaş testi\nbaşarı ile geçmiştir.");
}
```

KUMAŞ TESTİ

Resim Yolu:

Resim Ekle

Test Et

DAİRE KOORDİNATLARI:

	X:	Y:
1.	<input type="text" value="1,468"/>	<input type="text" value="1,759"/>
2.	<input type="text" value="1,521"/>	<input type="text" value="7,951"/>
3.	<input type="text" value="1,23"/>	<input type="text" value="14,592"/>
4.	<input type="text" value="7,845"/>	<input type="text" value="1,283"/>
5.	<input type="text" value="7,924"/>	<input type="text" value="14,777"/>
6.	<input type="text" value="14,274"/>	<input type="text" value="2,024"/>
7.	<input type="text" value="14,354"/>	<input type="text" value="7,898"/>
8.	<input type="text" value="14,459"/>	<input type="text" value="14,565"/>

Atkı Değerleri: **Çözümlü Değerleri:**

1.	<input type="text" value="12,806"/>	<input type="text" value="12,832"/>
2.	<input type="text" value="12,832"/>	<input type="text" value="13,494"/>
3.	<input type="text" value="13,229"/>	<input type="text" value="12,541"/>

Atkı Ortalama: **Çözümlü Ortalama:**

<input type="text" value="12,956"/>	<input type="text" value="12,956"/>
-------------------------------------	-------------------------------------

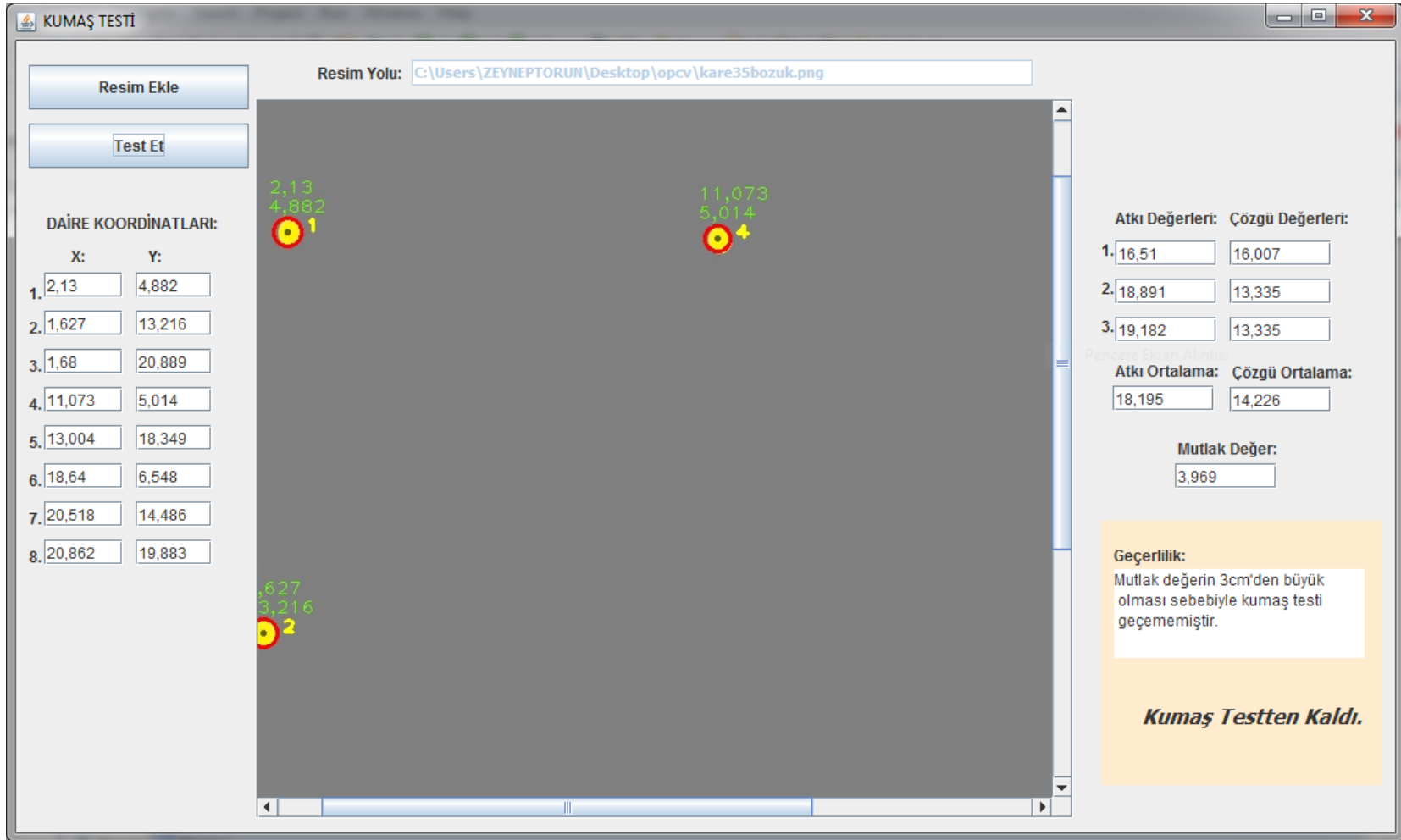
Mutlak Değer:

Geçerlilik:

Mutlak değer 3cm'den küçük olması sebebiyle kumaş testi başarı ile geçmiştir.

Kumaş Testi Geçti.

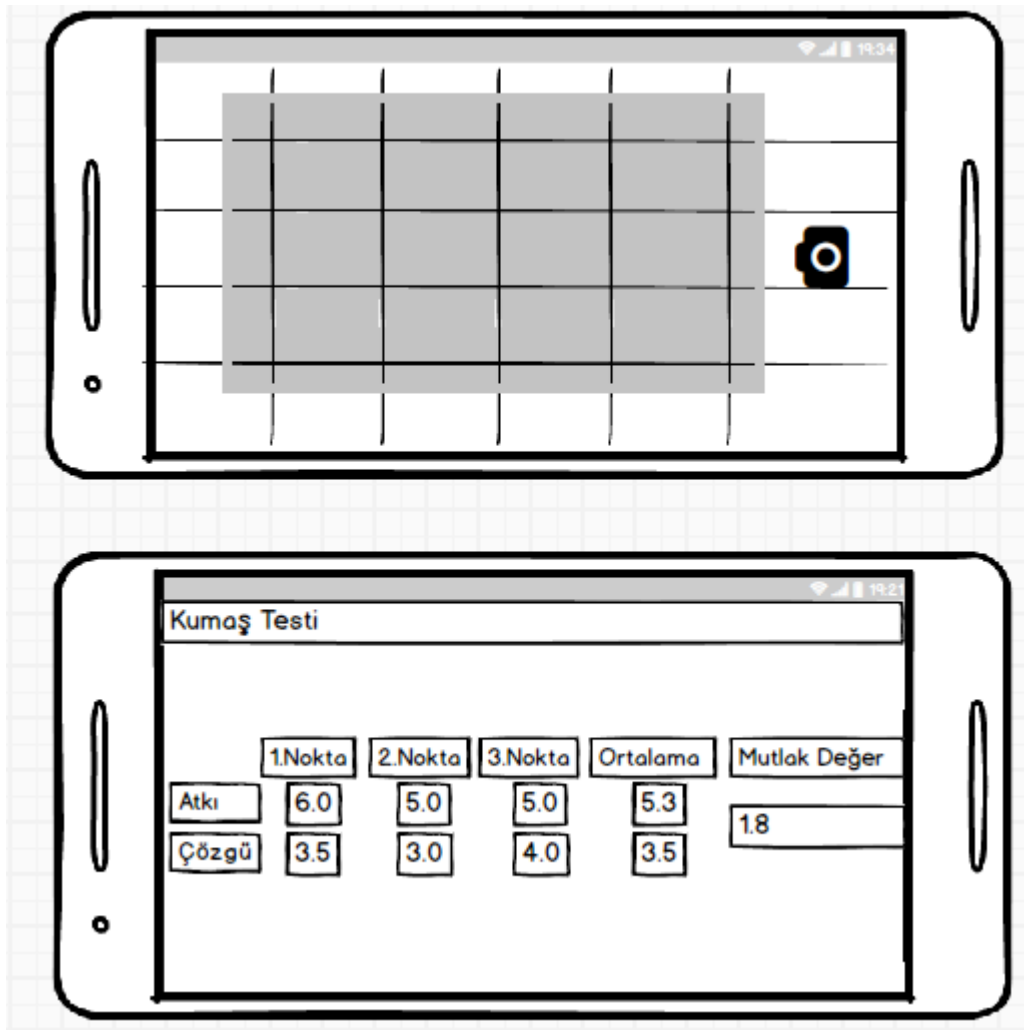
Şekil 3.10. Java masaüstü uygulaması



Şekil 3.11. Java masaüstü uygulaması -2

3.5. Projenin Geliştirilmesi

Eclipse platformunda Java dilinde yazılan uygulama rahatlıkla Android işletim sistemine geçirebileceği için Android işletiminin uygun bir seçenek olacağı düşünülmüştür. Uygulama Android 4.0 da API 14 versiyonu için yazılmasıyla çoğu mobil cihazın uyum sağlayabileceği bir uygulama yazılacaktır.



Şekil 3.12. Android Mockup görüntüsü

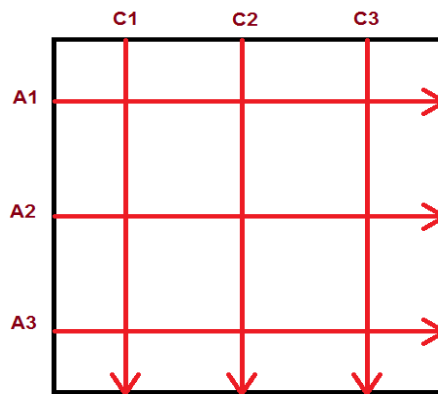
Hesaplama işlemlerinin doğruluğu, çekilen resmin çözünürlüğü ve fotoğrafı çekecek olan cihazın kumaşa olan konumu ile yakından ilgilidir. Eğer fotoğrafta kumaş herhangi şekilde yamuk veya kötü çıkarsa algılama işlemi veya sonuç buna bağlı olarak değişir. Testi uygulayacak kişilerin her biri fotoğrafı kendi çekmeye çalışırsa farklı konumlar, fotoğraf açıları, ışıklandırma, fotoğraf çözünürlüğü gibi sebeplerden

test sonuçları hatalı çıkabilir veya hiç algılama yapamayabilir. Bu duruma çözüm olarak cihaz ile kumaş arasında mesafenin sabitlenmesi amacıyla bir aparat düşünülebilir.

Aparatın alt kısmına konulan kumaş alt bölmede sabitlenmesiyle ve üste konulan cihaz (tablet, telefon vs.) fotoğrafın sabit aynı mesafeden çekilmesini sağlayacaktır. Ayrıca gerekli yerlere ışıklandırma yapılarak ortam ışığından bağımsız hale getirilebilir veya alt bölüme yerleştirilecek olan bant sistemi seri şekilde ölçüm işlemi yapıp bir sonraki işleme geçişin hızlandırılması sağlanabilir.

Geliştirme sürecinde dikkat edilmesi gereken bazı yazılımsal konular:

- Kumaş arkaplanının çok açık renkli veya beyaz gibi daire tespitini zorlaştıracak renkler olması tasarım aşamasında göz ardı edilmiştir. Bu soruna karşılık Matlab'den sonraki aşamada java kullanıldığında renkler konusunda kontroller yaptırılması, kullanılan filtrelerin tekrar gözden geçirilmesi gerektiği düşünülmüştür.
- Yazılan kodda kumaşın sabit bir renk olduğu varsayılmıştır. Fakat desen olarak daireler bulunabilir. Eğer ölçüm yaptığımız dairelerden daha büyük boyutlu daireler varsa göz ardı edilmesi kod ile sağlanabilir. Eğer desendeki daireler, şablon üzerinden konulan dairelerin boyutundaysa kod kendine göre seçeceği 8 noktaya göre yanlış sonuçlar üreten işlemler yapacaktır. Bu soruna karşılık ise daire tespitinin tamamen kaldırılması düşünülmüştür. Eğer kumaşın belirli aralıklarla bir kenarından diğer kenarına kadar olan mesafe ölçülürse atkı ve çözgü değerleri daha sağlıklı elde edilebilir.



Şekil 3.13. Kenarlar arası ölçüm

BÖLÜM 4. SONUÇLAR VE ÖNERİLER

Kumaş üretimindeki en önemli kalite parametrelerinden biri boyutsal değişimdir. Bu çalışmada konfeksiyon işletmelerinde üretim sürecinde yardımcı olabilmek amacıyla bir görüntü işleme algoritması hazırlanmış ve masaüstü uygulaması geliştirilmiştir. Bu uygulama ile görüntü işleme teknolojileri kullanarak kumaş üzerine konulan noktalar tespit edilip, noktalar arası mesafe hesaplanarak kumaş boyutu üzerindeki değişimler hesaplanmıştır. Boyutu değişen, sorun oluşturabilecek kumaşlar, ölçü noktalarında yapılan hesaplama hataları ve diğer olası hatalar giderilmek istenmiş, hatasız hızlı ve verimli üretimin gerçekleşmesi adına bir uygulama gerçekleştirilmiştir.

KAYNAKLAR

- [1] Tekstil Dershanesi, “Kumaş Performans Testleri”, link:
<http://www.boyteks.com/mattress-ticking/quality/testing-standards/>,
Erişim Tarihi: Ekim 2018
- [2] Erol Kara, “Çekmezlik Testi”, link:
<http://www.erolkara.net/2015/09/sanfor-cekmezlikten-cektigimiz.html>,
Erişim Tarihi: Kasım 2018
- [3] Feriha Demirhan ve Binnaz Meriç, “Örme ve Kumaş Giysilerde Yıkama ve Kurutma Sonrası Boyut Değişimlerinin İncelenmesi”, Sayfa:383-384
Aralık 2004
- [4] Derya İşler, “Biçimsel (Morfolojik) Görüntü İşleme”, link:
<https://slideplayer.biz.tr/slide/10376908/>, Erişim Tarihi: Kasım 2018
- [5] Chris Solomon and Toby Breckon, “rgb2gray kodu kullanımı”,
Fundamentals of Digital Image Processing, Temmuz 2011
- [6] George Siogkas, Visual Media Processing Using Matlab Beginner's Guide
, Sep 24, 2013
- [7] Quanmin Zhu And Ahmad Taher Azar, “bwareaopen kodu kullanımı”,
Complex System Modelling and Control Through Intelligent Soft
Computations, Sayfa:755 ,Kasım 2014
- [8] Matlab Dokumantation, “regionprop property”, Link:
<https://www.mathworks.com/help/images/ref/regionprops.html>, Erişim
Tarihi: Kasım.2018
- [9] Aykut Gökdemir, “Matlab ile Görüntü İşleme”, Link:
<https://www.elektrikport.com/makale-detay/matlab-ile-goruntu-isleme-1-elektrikport-akademi/8196#ad-image-0> , Erişim Tarihi: Ekim.2018
- [10] Mesut Pişkin, “OpenCV Filtreler”, Link:
<http://mesutpiskin.com/blog/opencv-filtreler.html>, Erişim Tarihi:
Nisan.2019
- [11] OpenCV 2.4.9 Documentation, “Constant Field Values”, Link:

- <https://docs.opencv.org/java/2.4.9/constant-values.html#org.opencv>,
Erişim Tarihi: Mart 2019
- [12] Luigi De Russis, Alberto Sacco, “OpenCV Java Tutorials Documentation”,
Döküman Linki: <https://buildmedia.readthedocs.org/media/pdf/opencv-java-tutorials/latest/opencv-java-tutorials.pdf>, Erişim Tarihi: Mart 2019
- [13] Talha Koç, “OpenCV Eğitimi”, Link: <http://talhakoc.net/opencv-c-egitimi/>,
Erişim Tarihi: Mayıs 2019
- [14] Marvin Smith, “Introduction to OpenCV”, Link:
https://www.cse.unr.edu/~bebis/CS485/Lectures/Intro_OpenCV.pdf,
Erişim Tarihi: Mayıs 2019
- [15] Sanatsal Bilgi, “Java ile Resim İşlemleri”-“Java ile Windows Form
Oluşturma”, Link: <http://www.sanatsalbilgi.com/TEKNIKBI LGI/10/java-ile-windows-form-olusturma-81.html>, Erişim Tarihi: Mayıs 2019
- [16] Selim Karataş, “Java Swing Programlama”, Link:
<http://selimkaratas.com.tr/wp/java-swing-programlama.html#>, Erişim
Tarihi: Mayıs 2019
- [17] Mehmet Ethem Sultan, “JForm Araçları ”, Link:
<http://www.ethemsulan.com/2009/12/jtextfield-jpasswordfield-jtextareann.html>, Erişim Tarihi: Mayıs 2019

EKLER

EK A: Matlab Kodu:

```

RGB = imread('C:\Users\ZTRN\Desktop\mesafeHesapla.png');
I =     rgb2gray(RGB);
esik_degeri = graythresh(I);
bw = im2bw(I,esik_degeri);
bw = bwareaopen(bw, 50);
hold on

birlestirici = strel('square',3);
bw = imdilate(bw,birlestirici);
[B,L] = bwboundaries(bw,'noholes'); % nesneleri buluyoruz.

daireler = regionprops(L, 'Area', 'Centroid', 'Perimeter');
imshow(label2rgb(L, @jet, [.5 .5 .5]));
hold on;
konumlar=[0 0;0 0;0 0 ];
tocm=0.02645833;
for n=1:size(daireler,1)
    cent= daireler(n).Centroid;
    X=cent(1);
    Y=cent(2); %dairenin merkez koordinatlar
    if daireler(n).Area>5 %%nesnelere filtre yapmak icin
        text(X-70,Y-20,['X=',num2str(X*tocm),...
            ', Y=',num2str(Y*tocm)]);
        konumlar(n,1)= X*tocm;
        konumlar(n,2)= Y*tocm;
    end
end
% y koordinat değerlerine göre sıralama yapılması
dizibir=[0 0;0 0;0 0];
diziiki=[0 0;0 0];
diziuc=[0 0;0 0;0 0];

for i=1:size(konumlar,1)
    for k=1:2
        if i<4
            dizibir(i,k) = konumlar(i,k);
        elseif i>5
            diziuc(i-5,k) = konumlar(i,k);
        else
            diziiki(i-3,k) = konumlar(i,k);
        end
    end
end
dizibir= sortrows(dizibir,2);
diziiki= sortrows(diziiki,2);
diziuc= sortrows(diziuc,2);
...

```

```

for i=1:size(konumlar,1)
    if i<4
        text(300,90+30*i,[num2str(i),'=>','X=',' ...
            num2str(dizibir(i,1))',' Y=',num2str(diziuc(i,2))]);
    elseif i>5
        text(300,90+30*i,[num2str(i),'=>','X=',' ...
            num2str(diziuc(i-5,1))',' Y=',num2str(diziuc(i-5,2))]);
    else
        text(300,90+30*i,[num2str(i),'=>','X*=',num2str( ...
            diziiki(i-3,1))',' Y=',num2str(diziiki(i-3,2))]);
    end
end
hold on
% atk1 ve çözgü değerlerinin bulunması
atki1=(diziuc(1,1)-dizibir(1,1));
atki2=(diziuc(2,1)-dizibir(2,1));
atki3=(diziuc(3,1)-dizibir(3,1));
text(310,400,['1. Atk1 Değeri : ',num2str(atki1)]);
text(310,440,['2. Atk1 Değeri : ',num2str(atki2)]);
text(310,480,['3. Atk1 Değeri : ',num2str(atki3)]);
cozgu1=(dizibir(3,2)-dizibir(1,2));
cozgu2=(diziiki(2,2)-diziiki(1,2));
cozgu3=(diziuc(3,2)-diziuc(1,2));
text(310,520,['1. Çözgü Değeri : ',num2str(cozgu1)]);
text(310,560,['2. Çözgü Değeri : ',num2str(cozgu2)]);
text(310,600,['3. Çözgü Değeri : ',num2str(cozgu3)]);

% atk1 ve çözgü değerlerinin ortalaması ve mutlak değer işlemi
atkiort=(atki1+atki2+atki3)/3;
cozguort=(cozgu1+cozgu2+cozgu3)/3;
if atkiort > cozguort
    mutlak =atkiort-cozguort;
else
    mutlak=cozguort-atkiort;
end

if mutlak>3
    title(['Atk1Ortalama: ',num2str(atkiort),' ÇözgüOrtalama: ',...
        num2str(cozguort),' Mutlak Değer: ', ...
        num2str(mutlak),' cm --TESTİ KALDI--']);
else
    title(['Atk1Ortalama: ',num2str(atkiort),' ÇözgüOrtalama: ',...
        num2str(cozguort),' Mutlak Değer: ', ...
        num2str(mutlak),' cm --TESTTEN GEÇTİ--']);
end

```

ÖZGEÇMİŞ

Zeynep Torun 15.03.1994’de İstanbul’da doğdu. İstanbul Halkalı Ticaret Meslek Lisesi Bilgisayar Bölümü veri tabanı dalını bitirdi. Ön Lisansı Denizli’de Bilgisayar Programcılığı bölümünü bölüm birinciliğiyle bitirdi. 2014 yılında Avas Teknoloji’de stajını yaptı. 2015 yılında Sakarya Üniversitesi Bilgisayar Mühendisliği bölümü kazandı.

BSM 498 BİTİRME ÇALIŞMASI DEĞERLENDİRME VE SÖZLÜ SINAV TUTANAĞI

KONU : GÖRÜNTÜ İŞLEME TABANLI KUMAŞ KALİTESİ TEST YAZILIMI
ÖĞRENCİLER (Öğrenci No/AD/SOYAD): G151210307 ZEYNEP TORUN

Değerlendirme Konusu	İstenenler	Not Aralığı	Not
Yazılı Çalışma			
Çalışma klavuza uygun olarak hazırlanmış mı?	x	0-5	
Teknik Yönden			
Problemin tanımı yapılmış mı?	x	0-5	
Geliştirilecek yazılımın/donanımın mimarisini içeren blok şeması (yazılımlar için veri akış şeması (dfd) da olabilir) çizilerek açıklanmış mı?			
Blok şemadaki birimler arasındaki bilgi akışına ait model/gösterim var mı?			
Yazılımın gereksinim listesi oluşturulmuş mu?			
Kullanılan/kullanılması düşünülen araçlar/teknolojiler anlatılmış mı?			
Donanımların programlanması/konfigürasyonu için yazılım gereksinimleri belirtilmiş mi?			
UML ile modelleme yapılmış mı?			
Veritabanları kullanılmış ise kavramsal model çıkarılmış mı? (Varlık ilişki modeli, noSQL kavramsal modelleri v.b.)			
Projeye yönelik iş-zaman çizelgesi çıkarılarak maliyet analizi yapılmış mı?			
Donanım bileşenlerinin maliyet analizi (prototip-adetli seri üretim vb.) çıkarılmış mı?			
Donanım için gerekli enerji analizi (minimum-uyku-aktif-maksimum) yapılmış mı?			
Grup çalışmalarında grup üyelerinin görev tanımları verilmiş mi (iş-zaman çizelgesinde belirtilebilir)?			
Sürüm denetim sistemi (Version Control System; Git, Subversion v.s.) kullanılmış mı?			
Sistemin genel testi için uygulanan metotlar ve iyileştirme süreçlerinin dökümü verilmiş mi?			
Yazılımın sızma testi yapılmış mı?			
Performans testi yapılmış mı?			
Tasarımın uygulamasında ortaya çıkan uyumsuzluklar ve aksaklıklar belirtilerek çözüm yöntemleri tartışılmış mı?			
Yapılan işlerin zorluk derecesi?	x	0-25	
Sözlü Sınav			
Yapılan sunum başarılı mı?	x	0-5	
Soruları yanıtlama yetkinliği?	x	0-20	
Devam Durumu			
Öğrenci dönem içerisindeki raporlarını düzenli olarak hazırladı mı?	x	0-5	
Diğer Maddeler			
Toplam			

DANIŞMAN (JÜRİ ADINA): DOÇ. DR. DEVRİM AKGÜN
DANIŞMAN İMZASI: