



SAKARYA  
UYGULAMALI BİLİMLER  
ÜNİVERSİTESİ

2023 - 2024 | BAHAR

# Görüntü İşleme Final Ödevi

## Öğrencinin

Adı Soyadı : M. Fatih SARIBİYIK – Ezgi GÜL

Bölümü : Bilgisayar Mühendisliği

Okul No : B210109013 - B210109004

Ödev Konusu : PCB'de Hata Tespiti



## 1. PROJENİN AMACI VE ÖZETİ:

Projemiz PCB devresi üzerindeki hataları tespit etmek için Python ve openCV kütüphanelerini kullanarak hatalı yerleri bulup karşılaştırma yapmaktadır.

**Kusur Tespiti:** Projede kusur tespiti genel olarak 2 tane input fotoğraf alınır . biri orijinal diğeri kusurludur. Kusurlu olan fotoğraf yamuksa düzelttilir ve gri formata çevrilir. Orijinal olanda gri tonlamaya çevirilir. Aynı boyutta olan iki fotoğraf üst üste konularak tonlamanın olduğu bölgeler kusurdan dolayı kaynaklandığı anlamına gelir ve bu hatanın merkezi tespit edilerek hatanın türüne göre işaretlemeler yapılmaktadır.

Doğru bölgelerin (gerçek kusurun olduğu yerler) tespit edilmesi için bir eşik değer ataması yapılmıştır. Bu atama bir değer aralığı olarak belirledik. Bu değer aralığında ise hata işaretleme yapıp ekranda gösterilir.

Genel olarak hata tespit mantığı aynı olsa da missing hole, open circuit ve mouse bite hatalarını bulurken küçük farklılıklar vardır. Ne gibi farklılıklar olduğunu aşağıda bölümlerde belirttik.

**Fotoğraf Çıktısı:** 4 adet çıktı vardır: 1. Orijinal, 2. Hatalı PCB, 3.Düz hale çevrilmiş PCB, 4. Hataları işaretlenmiş PCB.

**Konsol Çıktısı:** XML dosyasından hataları tespiti konsol ekranına bastırılarak gösterilmektedir.

## 2. PROBLEMİN ÇÖZÜM TASARIMI:

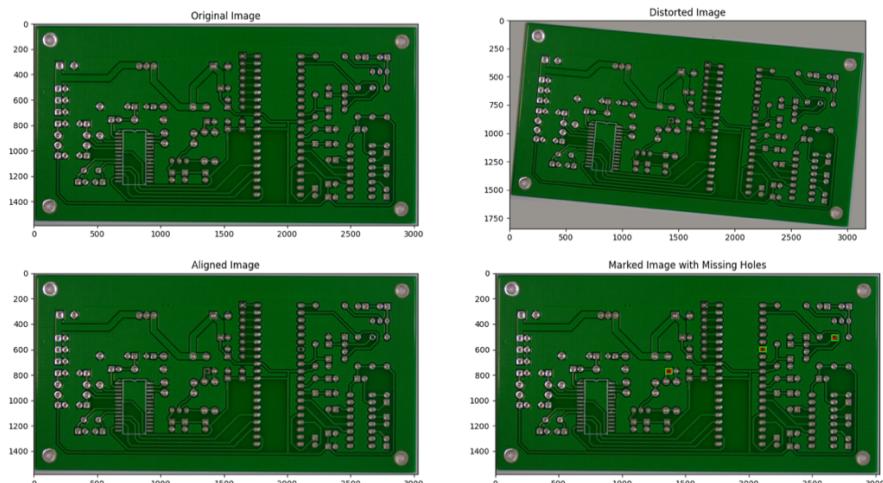
### 2.1. Missing Hole Problemi:

**2.1.1. Probleme Çözüm Önerisi:** İlk olarak Hatalı PCB devresini işlemlerden geçiriyoruz bu yamuk ise düzgün olması için çevreme işlemi ve görüntü işleme yapabilmek için gri tonlama yapıyoruz. ORB dedektörü, resimlerdeki ortak noktaları bularak hizalamaya yapmamızı sağlar(karşılaştırma için). Sonra hata tespiti için karşılaştırma yapılır. Farklılıkların bulunması sırasında eşik değere bakılır. Aynı olan kısımlar atılır.

Diğer kodlardan farklı olarak eşik değerimiz yani countur  $>100$  işlemi bir aralık olarak belirlenmemiştir. Sadece çok küçük ayrıntıları görmemesi yeterlidir. Bu yüzden bir değer aralığı değil bir değerden büyük olması işlemini yaptık.

Belirlenen yerleri openCV kullanarak işaretledik. İşaretlenen alanları XML dosyasındaki gerçek hatalarla ile karşılaştırarak programımız ne kadar iyi çalışıyor tespit edip konsola yazdırıldı. Bunun için xml dosyasına parse işlemi yapıp değerleri listeledik. Daha sonrasında bizim bulduğumuz değerler, listeye attığımız nesnelere yakın mı kontrol edeceğiz. Son olarak kullanıcının görebilmesi için ekranda hatalı bölgeler işaretli olarak gösterilmektedir.

### 2.1.2. Ekran Çıktıları:

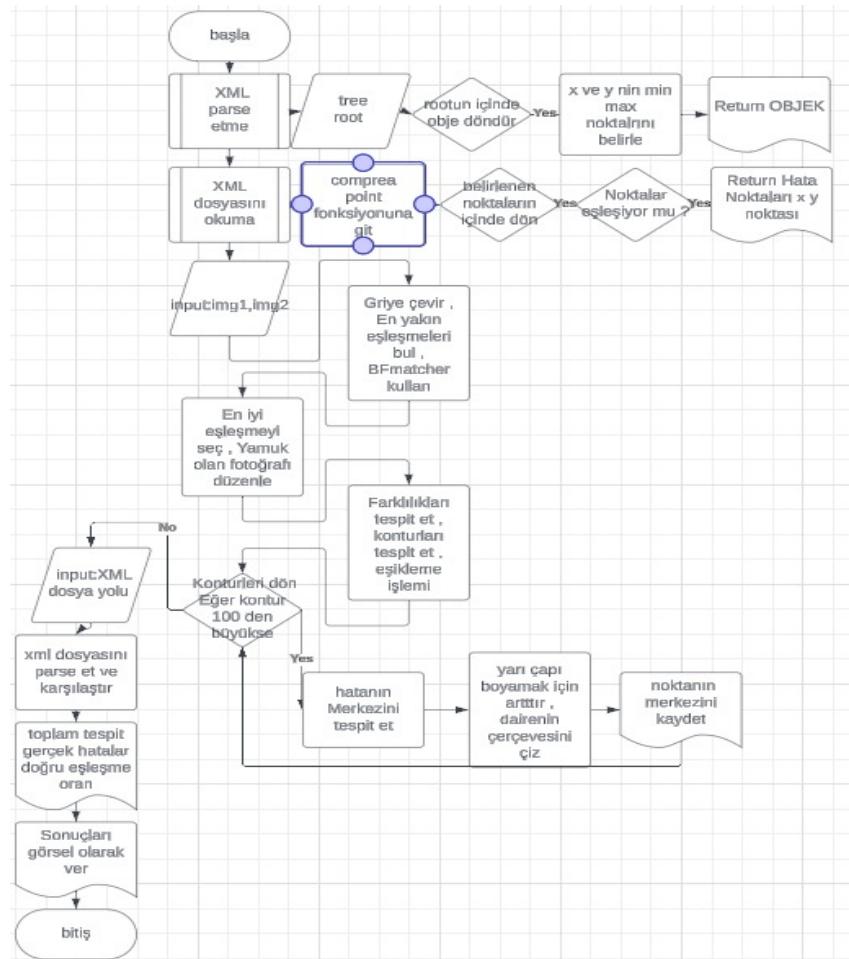


Şekil 2.1.2.1: 3. Test Fotoğrafının Çıktısı

```
> /usr/local/bin/python3 /Users/ezgigul/Desktop/goruntu_isleme_final/missingh.py
Toplam Tespit Edilen: 3
Toplam Gerçek: 3
Doğru Eşleşmeler: 3
Doğruluk Oranı: 100.00%
```

Şekil 2.1.2.2: 3. Test Fotoğrafının Terminal Çıktısı

### 2.1.3. Çözüm Önerisinin Algoritmasının Akış Diyagramı ile Açıklanması:



#### 2.1.4. Kod İyileştirmeleri ve Sonuçlar:

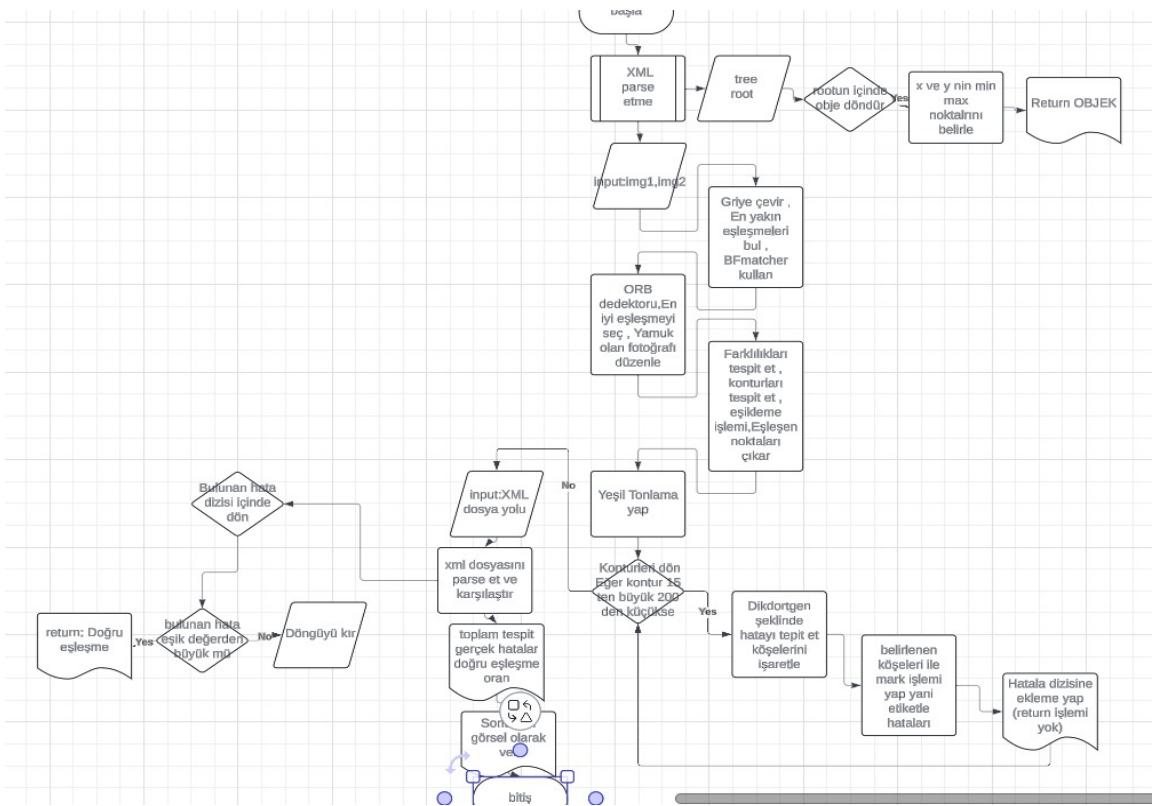
PCB devresi üzerindeki çok küçük delikleri hata olarak görmemesi için eşik değerini Countur  $>100$  yaptıktı. Program sadece büyük olan delikleri görüyor bu sayede. Resimler küçük olduğu için işaretlemeler ince bir halka ile gösteriliyor. Bunu çizgi kalınlığı ve yeşil işaretlemeler ile daha belirgin bir hale getirdik. Kodumuz çalıştırıldıktan sonra konsolda kaç tane hata bulunduğu ve yüzde hatası verdikten sonra ekranda PCB devresinin nereleri hatalı eski hali ve orijinal PCB devresinin hallerini gösterdik.

#### 2.2. Mouse Bite Problemi:

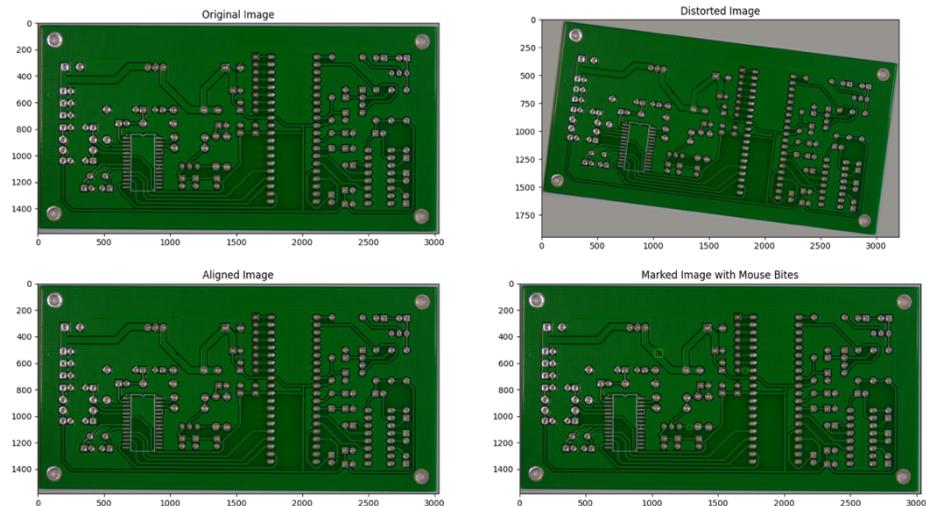
- 2.2.1. **Probleme Çözüm Önerisi:** İlk olarak hatalı ve orijinal PCB devresini ekliyoruz. Hatalı PCB devresini işlemlerden geçiriyoruz: bu yamuk ise düzgün olması için çevirme işlemi ve görüntü işleme yapabilmek için gri tonlama, ORB dedektörü kullanarak resimlerdeki ortak noktaları bularak hizalama yapmamızı sağlar (karşılaştırma için). Sonra hata tespiti için karşılaştırma yapılır. Farklılıkların bulunması sırasında eşik değere bakılır. Aynı olan kısımlar atılır. Buraya kadar her şey aynıdır. **FARKLI OLAN ŞEY** ise:  
 Countur bir değerden büyük değil iki değer arasında seçilmişdir. Bizim projemizde bu aralık 15 ile 200 aralığındadır. Bu sayede bulunan hatalar daha tutulabilir bir şekilde bize gösterilir.

Sonrasında yapılan OpenCV ve Xml karşılaştırma işlemleri aynıdır.

#### 2.2.2. Çözüm Önerisinin Algoritmasının Akış Diyagramı ile Açıklanması



### 2.2.3. Ekran Çıktıları:



**Şekil 2.2.3.1: 3. Test Fotoğrafının Çıktısı**

**Toplam Gerçek:** 3  
**Doğru Eşleşmeler:** 2  
**Doğruluk Oranı:** 66.67%

**Şekil 2.2.3.2: 3. Test Fotoğrafının Terminal Çıktısı**

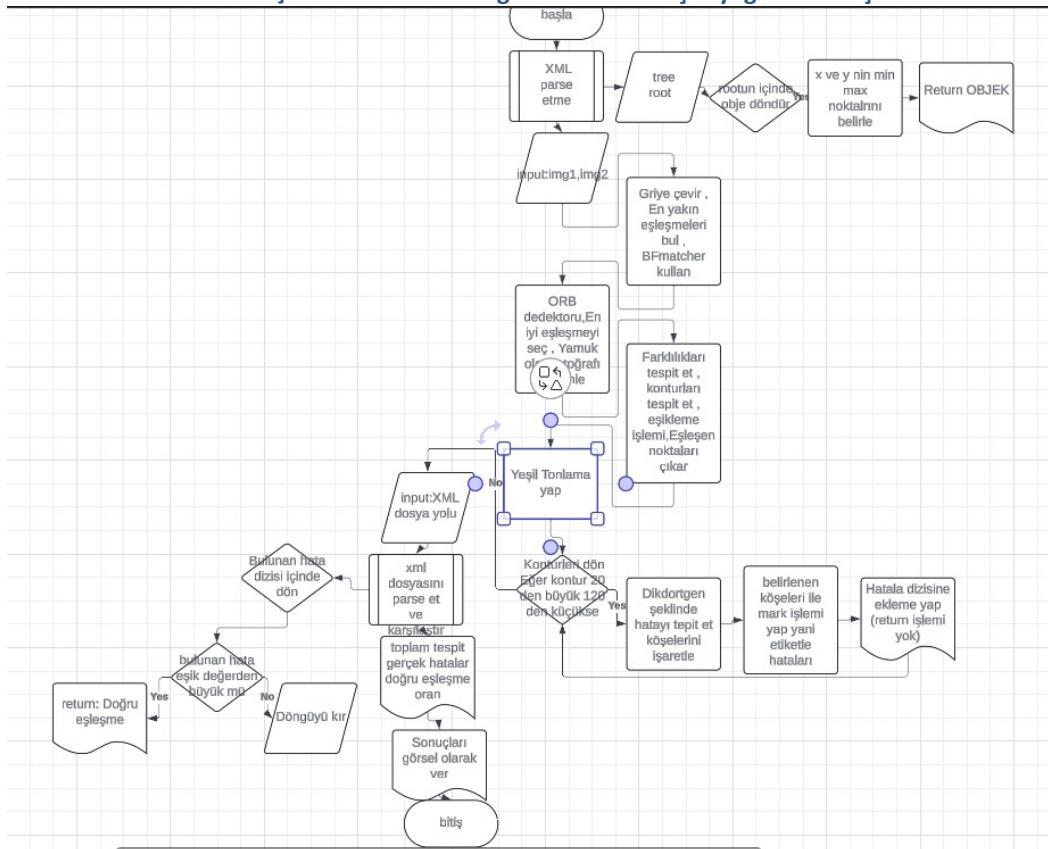
- 2.2.4. Kod İyileştirmeleri:** Kodumuzun yanlış yerleri tespit etmemesi için eşik değerini 15 ile 200 arasında tutuk. En iyi sonucu bu aralıkta aldık. Karşılaştırma işleminde yeşil maske kullandık ve vida kısmındaki parlaklıkları hata olarak algılıyor. Bu hataları yeşil maskeleme yöntemini kullanarak giderdik. Hata hizalanmış ve orijinal görüntü arasındaki farkları yeşil maske ile birleştirerek çözüldü.

### 2.3. Open Circuit Problemi:

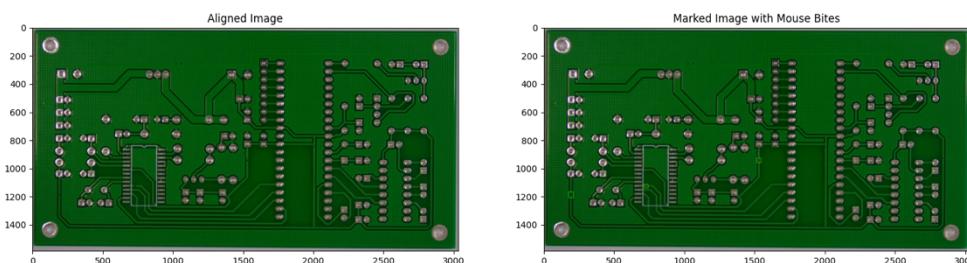
- 2.3.1. Probleme Çözüm Önerisi:** İlk olarak Hatalı PCB devresini işlemlerden geçiriyoruz: bu yamuk ise düzgün olması için çevirme işlemi ve görüntü işleme yapabilmek için gri tonlama, ORB dedektörü kullanarak resimlerdeki ortak noktaları bularak hizalama

yapmamızı sağlar(Karşılaştırma için). Sonra hata tespiti için karşılaştırma yapılır. Aynı olan kısımlar çıkarılır. Vidaların parlaklıından kaynaklı hatalar oluşmasının diye yeşil bölge maskelemesi yapılır. Morfolojik işlemler yapılarak daha doğru bir kontur bulunması sağlanır. Open circuitleri kare içerisinde almak için bir eşik değeri belirlenir(20 ile 120 arası), XML dosyasındaki değerlerle karşılaştırılıp doğruluk oranı hesaplanır. Bu hesaplamayı yaparken xml dosyasındaki değerlerin oluşturduğu karenin belli bir px kadar içinde mi diye kontrol ettik. Daha rahat görebilmek için buraları görselleştirip çıktıyı yansittık.

### 2.3.2. Çözüm Önerisinin Algoritmasının Akış Diyagramı ile Açıklanması:



### 2.3.3. Ekran Çıktıları:



**Şekil 2.3.3.1: 6. Test Fotoğrafının Çıktısı (original image ve distorted image kısmı kırıldı siğması amacıyla)**

```
> /usr/local/bin/python3 /Users/ezgigül/Desktop/goruntu_isleme_final/open_circuit.py
Toplam Tespit Edilen: 2
Toplam Gerçek: 3
Doğru Eşleşmeler: 2
Doğruluk Oranı: 66.67%
```

**Şekil 2.3.3.2: 6. Test Fotoğrafının Terminal Çıktısı**

**2.3.4. Kod İyileştirmeleri:** Kare içine almak için kullandığımız eşik değer önceden 20 ile 120 arasında değil, daha genişti. Bu nedenle doğru noktaları bulmakta zorlanıyorduk. Eşik değerinin aralığını azaltmamız open circuit hatalarının doğru boyutuna ulaşmamızı sağladı, seçtiğimiz başka bir fotoğraf(5. Örnek) için %100 doğruluk oranı yakalıyoruz. Ancak hala diğer örnekler için yeterli değil. Farklı eşik değerleriyle daha doğru sonuçlara ulaşabiliriz. Yeşil maskelemeyi de beyaz noktalarda da değişiklik görüyordu, düzeltmek için ekledik.