****

**KIRKLARELİ ÜNİVERSİTESİ**

**MÜHENDİSLİK FAKÜLTESİ**

**YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ**

**ALGORİTMA ANALİZİ VE TASARIMI DERSİ**

**Kısa Sınav 2. Ödevi Görsel Tanıma/Tespit Etme**

**HAZIRLAYAN**

1210505042 – Emirhan Tanrıverdi

**28 MAYIS 2023**

**İÇİNDEKİLER**

[**İÇİNDEKİLER** 1](#_Toc136197448)

[**Görüntü Tanıma/Tespit Etme** 2](#_Toc136197449)

[**Koda Giriş** 3](#_Toc136197450)

[**Programın Yapısı** 3](#_Toc136197451)

[**Programın Çalışması** 3](#_Toc136197452)

[getSimilarity() Metodu 4](#_Toc136197453)

[Collections.sort() Metodu 4](#_Toc136197454)

[toString() Metodu 5](#_Toc136197455)

[**Kodun Zaman Karmaşıklığı** 5](#_Toc136197456)

[**Sonuç** 5](#_Toc136197457)

Bu dokümanda, Görüntü Tanıma/Tespit Etme ve Java dilinde görsel benzerliği hesaplamak için yazdığım bir kodu anlatacağım. Kod, verilen bir dizindeki PNG formatındaki görselleri okuyarak benzerliklerini hesaplar ve sonuçları benzerlik derecesine göre sıralar.

# **Görüntü Tanıma/Tespit Etme**

Görüntü tanıma ve tespit, bilgisayarın dijital görüntülerde nesneleri algılayabilmesi ve sınıflandırabilmesi sürecidir. Bu teknoloji, bilgisayarın görsel verileri işleyerek nesneleri tanımasına ve belirli özellikleri algılamasına olanak tanır. Bu yöntem, birçok alanda kullanılan ve hızla gelişen bir araştırma alanıdır. Otomotiv, güvenlik, sağlık, eğlence ve robotik gibi birçok endüstride yaygın olarak kullanılmaktadır.

Bu yöntem süreci genellikle aşağıdaki adımlardan oluşur:

**1-)** **Veri Edinme:** İlk adım, görüntü verilerini elde etmektir. Bu işlem kameralar, tarayıcılar veya sensörler aracılığıyla gerçekleştirilebilir. Veriler, tek bir görüntüden veya görüntü akışından oluşabilir.

**2-)** **Önişleme:** Veri edinme aşamasından elde edilen görüntüler, önişleme adımında işlenir. Bu adımda, gürültü azaltma, boyutlandırma, dönüşüm ve kontrast düzenlemesi gibi işlemler yapılır. Amacı, verileri daha uygun bir biçimde analiz etmek için hazırlamaktır.

**3-)** **Özellik Çıkarımı:** Görüntü verilerinden anlamlı özelliklerin çıkarılması önemli bir adımdır. Bu adımda, desen tanıma algoritmaları kullanılarak görüntüdeki özellikler çıkarılır. Örneğin, kenarlar, köşeler, renk histogramları veya derin öğrenme tabanlı özellikler gibi farklı yöntemlerle özellik çıkarılabilir.

**4-)** **Sınıflandırma/Algılama:** Özellik çıkarımı aşamasından elde edilen veriler, sınıflandırma veya algılama algoritmalarıyla işlenir. Sınıflandırma, belirli bir nesneyi veya sınıfı tanımlamayı içerirken, algılama, birden fazla nesnenin konumunu ve sınıfını tespit etmeyi içerir. Bu aşamada, makine öğrenimi veya derin öğrenme algoritmaları kullanılabilir.

**5-)** **Sonuç İşleme:** Sınıflandırma/algılama sonuçları, görüntüdeki nesnelerin tanınmasını veya tespitini temsil eder. Bu sonuçlar daha sonra farklı amaçlar için kullanılabilir.

Bu yöntem alanında birçok farklı teknik ve algoritma bulunmaktadır. Bazı popüler yöntemler arasında destek vektör makineleri (SVM), yapay sinir ağları, konvolüsyonel sinir ağları (CNN), R-CNN (Bölge tabanlı CNN), YOLO (You Only Look Once) ve SSD (Single Shot MultiBox Detector) yer alır.

Sonuç olarak, bu yöntem bilgisayarların dijital görüntülerde nesneleri tanımasına ve sınıflandırmasına olanak sağlayan bir teknolojidir. Bu süreç; veri edinme, önişleme, özellik çıkarımı, sınıflandırma/algılama ve sonuç işleme aşamalarından oluşur. Bu yöntem alanı hızla gelişmekte olup birçok endüstride yaygın olarak kullanılan bir araştırma alanıdır.

# **Koda Giriş**

Benzerlik, renkler, desenler, şekiller veya diğer özelliklerin karşılaştırılmasıyla belirlenebilir. Bu program, basit bir benzerlik hesaplama yöntemi kullanır ve RGB renk uzayında, renk farklarını ödevde istenen şekilde hesaplayarak görsel benzerlik oranını belirler.

# **Programın Yapısı**

Program, Java'nın BufferedImage ve File sınıflarını kullanarak görselleri okur ve benzerlik hesaplama işlemlerini gerçekleştirir. Aşağıda programın genel yapısını gösteren bir akış şeması bulunmaktadır:

1-) Klasördeki tüm .png uzantılı dosyaları bulunur.

2-) Bu dosyaların her biri okunarak `BufferedImage` nesnelerine dönüştürülür.

3-) Her bir görsel çifti için benzerlik oranı hesaplanır ve `ImagePair` nesnelerine kaydedilir.

4-) Tüm `ImagePair` nesneleri benzerlik oranlarına göre sıralanır.

5-) Sıralanmış `ImagePair` nesne çiftleri, benzerlik oranlarına göre büyükten küçüğe olacak şekilde sıralanarak ekrana yazdırılır.

# **Programın Çalışması**

Program, belirtilen dizindeki PNG görsellerini alır ve her birini BufferedImage nesnelerine dönüştürür. Daha sonra, tüm görsel çiftlerini oluşturarak ödev dosyasında istenen şekilde benzerliklerini hesaplar. Son olarak, benzerlik skorlarına göre çiftleri sıralar ve sonuçları ekrana yazdırır.

Programın çalışma adımları şunlardır:

İlk adımda, programın çalışacağı dizini belirtiyoruz. Örneğin, "C:\Users\HP\Desktop\Görseller" gibi bir dizin yolu kullanabilirsiniz. Bu dizin, içindeki PNG görsellerini tarayacağımız dizindir.

İkinci adımda, belirtilen dizindeki PNG görsellerini listeleyerek her bir görseli temsil eden File nesnelerini alıyoruz.

Üçüncü adımda, her bir File nesnesini BufferedImage nesnesine dönüştürerek görselleri hafızada temsil ediyoruz. Bu, görsellerin işlenmesi için gereklidir.

Dördüncü adımda, her bir görsel çiftini oluşturarak benzerliklerini hesaplıyoruz. Bu adımda, getSimilarity() metodunu kullanarak iki görselin benzerlik skorunu buluyoruz (Bu metodun nasıl çalıştığını ileriki kısımda anlatacağım). Skor, her bir görsel çifti için ayrı ayrı hesaplanır.

Beşinci adımda, benzerlik skorlarına göre görsel çiftlerini sıralıyoruz. Bu, Collections.sort() metodunu kullanarak gerçekleştirilir (Bu metodun da nasıl çalıştığını ileriki kısımda anlatacağım). Sıralama, benzerlik skoruna göre azalan sırada yapılır.

Altıncı ve son adımda, sıralanmış görsel çiftlerini ekrana yazdırarak sonuçları görüntülüyoruz. Bu adımda, her bir ImagePair nesnesinin toString() metodunu kullanarak çiftin adını ve benzerlik skorunu elde ediyoruz. (Bu metodun nasıl çalıştığını ileriki kısımda anlatacağım).

## getSimilarity() Metodu

Programın en önemli bölümlerinden biri getSimilarity() metodu. Bu metot, iki BufferedImage nesnesini alır ve benzerlik skorunu hesaplar. İki görsel arasındaki benzerlik, renk farklarına dayanarak hesaplanır.

Metodun çalışma adımları şunlardır:

İki görselin genişlik ve yükseklik değerlerini alarak boyutlarını karşılaştırır. Eğer boyutlar farklı ise bir hata mesajı verir ve işlemi sonlandırır.

Görsellerin benzerlik skorunu tutmak için bir değişken tanımlarız ve başlangıç değerini sıfır olarak ayarlarız.

İki görselin piksellerini 20 birimlik aralıklarla tararız. Bu, görsellerin büyük boyutlarda olması durumunda daha hızlı bir hesaplama yapılmasını sağlar.

Her bir piksel için, RGB renk değerlerini alırız. Bu değerler, her pikselin kırmızı, yeşil ve mavi bileşenlerini temsil eder.

İki görsel arasındaki renk farklarını hesaplar ve bu farkı 255'e bölerek bir benzerlik oranı elde ederiz. Bu adımda, renk farkının hesaplanması için Euclidean Distance formülünü kullanırız.

Her bir piksel için hesaplanan benzerlik oranını toplarız.

Toplam benzerlik oranını, piksel sayısına (20 birimlik aralıklarla tarandığı için) böleriz ve sonuç olarak elde edilen ortalama benzerlik skorunu döndürürüz.

## Collections.sort() Metodu

Collections.sort() metodunun kullanımı, bir liste üzerinde sıralama yapmayı sağlar. Bu programda, ImagePair nesnelerini içeren bir liste olan "pairs" kullanılır ve bu liste benzerlik skorlarına göre sıralanır.

Aşağıdaki adımlar, Collections.sort() metodunun çalışma mantığını açıklar:

Collections.sort() metodunu kullanarak, sıralama yapılacak olan "pairs" listesini ve bir Comparator nesnesini parametre olarak veriyoruz. Comparator nesnesi, sıralamanın nasıl yapılacağını belirler. Biz, ImagePair nesnelerini benzerlik skorlarına göre sıralamak istediğimiz için bir Comparator kullanırız.

Comparator.comparing() metodunu kullanarak bir Comparator oluşturuyoruz. comparing() metodu, bir nesnenin özelliğine göre karşılaştırma yapmamızı sağlar. Biz, ImagePair nesnelerini benzerlik skorlarına göre karşılaştıracağımız için ImagePair::getSimilarity ifadesini kullanırız. Bu ifade, ImagePair sınıfında tanımlanan getSimilarity() metodu ile benzerlik skoruna erişmemizi sağlar.

Sıralama işlemi, benzerlik skorlarına göre azalan sırada yapılması için reversed() metodunu kullanarak tersten sıralama yaparız. Yani, en yüksek benzerlik skorları en başta olacak şekilde sıralama yapılır.

Collections.sort() metodunun ardından, "pairs" listesi benzerlik skorlarına göre sıralanır.

Bu adımlar sonucunda, "pairs" listesi benzerlik skorlarına göre sıralanmış olur. Sıralama tamamlandıktan sonra, for-each döngüsü kullanarak sıralanmış görsel çiftlerini elde eder ve ekrana yazdırırız.

Bu şekilde, Collections.sort() metodunu kullanarak benzerlik skorlarına göre görsel çiftlerini sıralayabilir ve sonuçları elde edebilirsiniz.

## toString() Metodu

toString() metodu, bir nesnenin temsili bir dizeye dönüştürülmesini sağlayan Java'nın Object sınıfında tanımlanan bir metodudur. Bu metodun amacı, bir nesnenin içeriğini veya özelliklerini insan tarafından okunabilir bir şekilde sunmaktır.

Bu programdaki toString() metodu, ImagePair sınıfında kullanılır ve ImagePair nesnelerini bir dize olarak temsil eder. Bu metot, bir ImagePair nesnesinin "name1", "name2" ve "similarity" özelliklerini birleştirerek döndürür.

Aşağıdaki adımlar, toString() metodunun çalışma mantığını açıklar:

toString() metodu, ImagePair sınıfında Override edilir. Böylece, Object sınıfındaki varsayılan toString() metodu yerine, ImagePair sınıfındaki özel uygulama çalışır.

toString() metodu, "name1", "name2" ve "similarity" özelliklerini birleştirerek bir dize oluşturur. Bu özelliklere ImagePair sınıfının içindeki değişkenlere erişerek ulaşılır.

Oluşturulan dize, return ifadesiyle geri döndürülür. Bu dize, ImagePair nesnesinin temsilini sağlar ve örneğin "name1 - name2: similarity" gibi bir formata sahip olabilir.

Bu şekilde, toString() metodu, ImagePair nesnelerini bir dize olarak temsil eder ve bu dizeyi programın sonucunu ekrana yazdırmak için kullanırız. Bu sayede, görsel çiftlerini ve benzerlik skorlarını okunabilir bir şekilde görüntüleyebiliriz.

toString() metodu, Java'da genel olarak nesnelerin temsili ve hata ayıklama işlemlerinde kullanılır. Nesnenin içeriğini veya durumunu daha anlaşılır bir şekilde göstermek için bu metot özelleştirilebilir ve ihtiyaçlara göre düzenlenebilir.

# **Kodun Zaman Karmaşıklığı**

Bu kodun karmaşıklığı O(n^2)'dir. n, klasördeki görsel sayısına eşittir. Bu nedenle çok büyük sayıda görsel için bu kodun çalışma süresi uzayabilir.

Bu kod, bir dizindeki resimlerin benzerliğini karşılaştırır. Bu kodun zaman karmaşıklığı O(n^2 \* w \* h)'dir. Burada n dizindeki resim sayısıdır, w resimlerin genişliği ve h resimlerin yüksekliğidir. Bu, kodun her resim çiftini karşılaştırdığı (O(n^2) sürer) ve her resim çifti için renk değerlerini 20’şer piksel aralıklarla karşılaştırıldığından, zaman karmaşıklığı O(n^2 \* (w/20) \* (h/20)) olacaktır.O(n^2 \* (w/20) \* (h/20)) ifadesinde, n^2 terimi en yüksek dereceli terimdir. Sabit faktör olan w ve h, büyük O gösteriminde ihmal edilir. Bu nedenle, zaman karmaşıklığı bu kod için O(n^2) şeklinde ifade edilir.

# **Sonuç**

Program, belirtilen bir dizindeki PNG görsellerini okur, benzerliklerini hesaplar ve sonuçları benzerlik derecesine göre sıralar. Program, getSimilarity() metodunu kullanarak iki görsel arasındaki benzerlik skorunu hesaplar ve bu skorları kullanarak görsel çiftlerini sıralar.

Bu program, görsel benzerlik analizinde temel bir yöntem sunar ve özelleştirilerek geliştirilebilir. Daha karmaşık benzerlik ölçümleri veya farklı özelliklerin kullanılmasıyla daha gelişmiş bir görsel benzerlik analizi gerçekleştirilebilir.

Programı kullanırken, görsellerin yolu ve dosya uzantısı gibi parametreleri ihtiyaçlarınıza göre değiştirebilirsiniz. Ayrıca, getSimilarity() metodunu genişleterek farklı benzerlik hesaplama algoritmaları uygulayabilirsiniz.