**logo, amblem, simge, sembol, ticari marka içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**KIRKLARELİ ÜNİVERSİTESİ**

**MÜHENDİSLİK FAKÜLTESİ**

**YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ**

**ALGORİTMA ANALİZİ VE TASARIMI DERSİ**

**ALGORİTMA ANALİZİ VE TASARIMI**

**KISA SINAV 2 KISIM 2**

**HAZIRLAYANLAR**

1210505035 – Emirhan Şimşek

1210505017 – Ali Arslan

1210505033 – Muhammed Eren Yüksel

**MAYIS 2023**

Görsellerin birbirine olan benzerliğini belirlemek için çeşitli yöntemler bulunmaktadır. İşte en yaygın kullanılan yöntemlerden bazıları:

1. **Ortak özellikler:** Görsellerin benzerliklerini belirlemek için ortak özelliklerini kullanabilirsiniz. Örneğin, aynı nesneleri veya benzer renkleri içeren görseller birbirine benzeyebilir.
2. **Renk histogramları**: Görsellerin renk dağılımlarını analiz etmek için renk histogramları kullanabilirsiniz. Benzer renk dağılımlarına sahip görseller birbirine benzeyebilir.
3. **Özellik çıkarma:** Görsellerin özelliklerini çıkarmak için makine öğrenimi veya derin öğrenme tekniklerini kullanabilirsiniz. Örneğin, Convolutional Neural Network (CNN) gibi bir derin öğrenme modeli kullanarak görsellerin özellik vektörlerini çıkarabilirsiniz. Benzer özellik vektörlerine sahip görseller birbirine benzeyebilir.
4. **Kontrast ve benzerlik ölçüleri:** Görsellerin kontrast ve benzerlik ölçülerini kullanarak benzerliklerini değerlendirebilirsiniz. Örneğin, SSIM (Structural Similarity Index) veya MSE (Mean Squared Error) gibi ölçümler kullanılabilir.
5. **İçerik tabanlı arama:** Görsellerin içeriğini analiz ederek benzerliklerini belirleyebilirsiniz. Örneğin, görüntü tanıma algoritmaları kullanarak benzer nesneleri veya benzer kategorideki görselleri bulabilirsiniz.

Bu yöntemlerden herhangi birini kullanarak 150 adet görselin birbirine en çok benzeyenlerini bulabilir ve listeyebilirsiniz. Ancak, bu işlemi gerçekleştirmek için bir programlama dilinde uygun bir kod yazmanız gerekecektir. Görsel benzerlik algoritmalarını uygulamak için popüler kütüphaneler arasında OpenCV, scikit-image, TensorFlow ve PyTorch bulunmaktadır. Bu kütüphanelerden birini kullanarak, görsel benzerlik analizini gerçekleştirebilir ve sonuçları listelebilirsiniz.

**OpenCV**

OpenCV (Open Source Computer Vision Library), bilgisayar görüşü ile ilgili uygulamaları geliştirmek için kullanılan ve bir dizi C/C++, Python ve Java arayüzü sunan popüler bir yazılım kitaplığıdır. OpenCV, özgür ve açık kaynaklı bir kitaplık olup, hem akademik hem de ticari kullanım için tasarlanmıştır.

OpenCV'nin sağladığı işlevler arasında yüz ve nesne tanıma, görüntü işleme, optik akış, özellik tespiti, makine öğrenmesi ve derin öğrenme gibi birçok alan bulunur. Ayrıca, 2D ve 3D görüntü alanlarında gerçek zamanlı görüntü işleme yetenekleri sağlar.

Görüntü analizi ve manipülasyonu, robotik, yapay zeka, otomatik algılama ve birçok diğer teknoloji alanında kullanılır. Ayrıca, OpenCV, birçok platformda (Windows, Linux, MacOS, Android, iOS) çalışacak şekilde tasarlanmıştır ve çok geniş bir kullanıcı ve geliştirici topluluğuna sahiptir.

OpenCV, çeşitli görüntü işleme ve bilgisayar görüşü işlevlerine sahip olduğu için birçok uygulamada kullanılabilir.

**İşte bazı örnekler:**

**1. Yüz Tanıma:** OpenCV, yüz tanıma teknolojisi oluşturmak için kullanılabilir. OpenCV, yüzlerin belirlenmesine ve tanınmasına yardımcı olan Haar cascades ve deep learning modelleri gibi çeşitli teknikler ve algoritmalar sağlar.

**2. Nesne Tespiti:** OpenCV, belirli nesnelerin bir görüntü veya video içinde belirlenmesine yardımcı olabilir. Bu, hedef nesnenin belirli özelliklerini veya tanımlayıcılarını kullanarak gerçekleştirilir.

**3. Görüntü İşleme:** OpenCV, görüntülerin manipüle edilmesine ve işlenmesine yardımcı olabilir. Görüntü döndürme, çevirme, yeniden boyutlandırma, kesme, renk uzayı dönüşümleri ve bulanıklık gibi bir dizi farklı işlemi gerçekleştirebilir.

**4. Gerçek Zamanlı Video İşleme:** OpenCV, gerçek zamanlı video işlemeyi destekler. Bu, bir video akışında yüz veya nesne tanıma, hareket tespiti veya arka plan ayırma gibi gerçek zamanlı işlemleri gerçekleştirmeyi içerebilir.

**5. Makine Öğrenmesi ve Derin Öğrenme:** OpenCV, makine öğrenmesi ve derin öğrenme uygulamalarını destekler. Bu, görüntülerdeki belirli özellikleri öğrenmek ve tanımak için makine öğrenmesi modellerini eğitmeyi ve kullanmayı içerebilir.

**6. 3D Görüntü İşleme:** OpenCV, 3D görüntüler ve videoların işlenmesini destekler. Bu, 3D nesnelerin tespiti ve tanınması veya stereoskopik görüntülerin oluşturulması gibi işlemleri içerebilir.

Bu geniş fonksiyon yelpazesi, OpenCV'yi bilgisayar görüşü ve görüntü işleme alanındaki en popüler araçlardan biri haline getirir. Geliştiriciler, OpenCV'yi kendi uygulamalarına ve projelerine entegre etmek için bir dizi API ve arayüzden yararlanabilirler.

OpenCV'nin bir diğer önemli özelliği, çoklu platform desteğidir. Bu, OpenCV'nin farklı işletim sistemlerinde (Windows, Linux, macOS) ve farklı cihazlarda (PC, mobil cihazlar) çalışabilmesi anlamına gelir. Ayrıca, OpenCV'nin Python, C++, Java gibi farklı programlama dilleri ile uyumlu arayüzleri mevcuttur.

**OpenCV'nin bazı ileri düzey özelliklerine değinelim:**

1. **Panorama Oluşturma**: Birden çok görüntüyü birleştirerek tek bir panoramik görüntü oluşturabilir.
2. **Optik Akış**: Hareketin, hareketli nesnelerin ve kameraların analiz edilmesi için görüntü dizileri üzerinde optik akış hesaplamaları gerçekleştirir.
3. **Stereoskopik Görüntüleme**: İki ayrı görüntüden üç boyutlu görüntüler oluşturur.
4. **Makine Öğrenmesi ve Derin Öğrenme**: OpenCV, Support Vector Machines (SVM), K-Nearest Neighbors (KNN), Random Forests ve diğer popüler makine öğrenmesi algoritmalarını içerir. Ayrıca, Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN) gibi derin öğrenme modellerini destekler ve TensorFlow, PyTorch gibi popüler derin öğrenme çerçeveleriyle entegrasyon sağlar.
5. **Arka Plan Ayırma**: Arka planın bir video dizi veya hareketli bir görüntüden çıkarılmasına yardımcı olur.
6. **Augmented Reality (AR) ve Virtual Reality (VR)**: OpenCV, gerçek dünyanın görüntülerine dijital öğeler eklemek veya tamamen sanal bir ortam oluşturmak için kullanılabilir.

Bu geniş işlevsellik, geliştiricilere ve araştırmacılara, çok çeşitli uygulamalar oluşturmak için büyük bir esneklik sağlar. Bu uygulamalar güvenlik, otomotiv, robotik, tıp, eğitim ve daha birçok sektörde kullanılır. Yine de, OpenCV'nin her türlü bilgisayar görüşü uygulamasının gereksinimlerini karşılaması beklenemez, bu nedenle bazen daha özelleştirilmiş veya gelişmiş teknikler ve araçlar gerekebilir.

**KODUN ÇALIŞMA MANTIĞI**

Bu Java kodu, bir dizi resmi okuyor ve her resmin diğer resimlere olan benzerliğini karşılaştırıyor. OpenCV kütüphanesinin birkaç özelliğini kullanır. İşte bu kodun nasıl çalıştığına dair maddeli bir açıklama:

1. **Resimlerin Okunması**: **readImages** fonksiyonu belirli bir dizindeki resimleri okur. Bu işlev, dizindeki her bir ".png" dosyasını bir Mat nesnesi (OpenCV'nin temel görüntü sınıfı) olarak okur ve bu nesneyi ve resmin adını içeren ImageData nesnelerinin bir listesini oluşturur.
2. **Resimlerin Karşılaştırılması**: **compareImages** fonksiyonu, iki görüntünün benzerliğini karşılaştırır. Bu, **Imgproc.matchTemplate** işlevini kullanarak yapılır, bu işlev bir görüntüyü diğerine göre eşleştirir ve her bir konum için bir benzerlik skoru verir. Bu işlev, en yüksek skoru bulmak için skor matrisini tarar.
3. **En Benzer Resimleri Bulma**: **findMostSimilar** fonksiyonu, her resim için en benzer resmi bulur. Her resmi, diğer tüm resimlerle karşılaştırır ve en yüksek benzerlik skoruna sahip olanı seçer.
4. **Sonuçların Yazdırılması**: Son olarak, **main** fonksiyonu en benzer resimleri bulur ve her bir resim için en benzer resmin adını yazdırır.

\*\*NOT: Eğer çıktı alması uzun sürüyorsa for (int i = 21; i < 2150; i++) kod parçasında i<…. Daha düşük bir değer girilerek çıktıya ulaşılabilir. Aynı Zamanda resimlerin bulunduğu dosyanın dosya yolu kod da değiştirilmelidir.

**ZAMAN KARMAŞIKLIĞI**

Karmaşıklık analizi, bir algoritmanın performansının belirlenmesi için kullanılan bir tekniktir. Algoritma karmaşıklığını belirlemek için genellikle zaman ve alan karmaşıklığı olmak üzere iki ana parametre kullanılır.

1. **Zaman Karmaşıklığı**: Zaman karmaşıklığı, bir algoritmanın tamamlanması için gereken zamanı temsil eder. Genellikle, algoritmanın tamamlanması için gereken adım sayısına veya işlemlerine dayanır.
2. **Alan Karmaşıklığı**: Alan karmaşıklığı, bir algoritmanın çalıştırılması sırasında kullanılan hafıza miktarını temsil eder. Bu, algoritmanın verileri saklamak ve işlemek için ne kadar hafıza alanı gerektirdiğine dayanır.

Algoritma analizi ve karmaşıklık analizi, bir problemin en iyi algoritmik çözümünü belirlemeye yardımcı olabilir. Bu, verimli algoritmaların oluşturulması ve uygulanması için çok önemlidir. Örneğin, vermiş olduğunuz kod parçası için, **findMostSimilar** fonksiyonunun zaman karmaşıklığı O(n^2) olacaktır. Çünkü burada iki döngü kullanılmış ve her bir görüntü diğer tüm görüntülerle karşılaştırılıyor. Dolayısıyla, görüntü sayısındaki bir artış, işleme süresini kare olarak artırır.

* **O(n^2) - Karesel Zamanlı:** İşlem sayısı, girdi boyutunun karesine eşittir. Girdi boyutu iki katına çıktığında, işlem süresi dört katına çıkar. Örneğin, iki döngü içeren basit sıralama algoritmaları (bubble sort, insertion sort, selection sort).

Bunun yanı sıra, bu kod parçası, algoritmanın çalışma süresi boyunca tüm görüntülerin bellekte tutulduğu için büyük bir hafıza kullanımına sahip olabilir. Bu durum, büyük bir görüntü seti ile çalışıldığında performans sorunlarına neden olabilir. Bu yüzden karmaşıklık analizi, bir algoritmanın verimliliğini değerlendirmek ve onu daha da iyileştirmek için önemlidir.