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

Açıklama otomatik olarak oluşturuldu

**KIRKLARELİ ÜNİVERSİTES****İ MÜHENDİSLİK FAKÜLTESİ**

**YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ**

**ALGORİTMA ANALİZİ VE TASARIMI DERS****İ 2.KISA SINAV 2.ÖDEV**

**Buğra GÖRMÜŞ1, Recep Umut AKPINAR2**

**KIRKLARELİ – 2023**

1 Öğrenci No: 1200505068

2 Öğrenci No: 1190505805

**OpenCV Template Matching**

OpenCV Template Matching, bir görüntüdeki belirli bir şablonun yerini bulmak için kullanılan bir görüntü işleme tekniğidir. Bu yöntem, bir şablonun bir hedef görüntü içindeki konumunu bulmak için benzerlik ölçütleri kullanır.

Template Matching işlemi aşağıdaki adımları içerir:

1. **Hedef Görüntü ve Şablon Görüntüsünün Yüklenmesi:**

* Template Matching işlemi için öncelikle hedef görüntü ve şablon görüntüsü yüklenir.
* Hedef görüntü, şablonun aranacağı ana görüntüdür.
* Şablon görüntüsü ise hedef görüntü içinde bulunması istenen deseni, nesneyi veya şekli temsil eder.

1. **Benzerlik Ölçütlerinin Belirlenmesi:**

* Template Matching işlemi için benzerlik ölçütleri belirlenir.
* Benzerlik ölçütleri, hedef görüntü üzerindeki şablonun ne kadar benzediğini ölçer.
* OpenCV'de kullanılabilecek bazı benzerlik ölçütleri şunlardır: cv2.TM\_CCORR, cv2.TM\_CCORR\_NORMED, cv2.TM\_CCOEFF, cv2.TM\_CCOEFF\_NORMED, cv2.TM\_SQDIFF, cv2.TM\_SQDIFF\_NORMED.
* Benzerlik ölçütü seçimi, uygulama senaryosuna ve veri setine bağlı olarak değişebilir. Farklı ölçütler, farklı eşleşme sonuçlarına yol açabilir.

1. **Template Matching İşleminin Gerçekleştirilmesi:**

* cv2.matchTemplate() fonksiyonu kullanılarak template matching işlemi gerçekleştirilir.
* Bu işlev, hedef görüntü üzerinde şablonu kaydırarak benzerlik ölçütlerini hesaplar.
* Her bir kaydırma konumu için benzerlik ölçütleri hesaplanır ve bir benzerlik haritası oluşturulur.

1. **En İyi Eşleşmenin Bulunması:**

* Oluşturulan benzerlik haritası üzerinde, en yüksek benzerlik ölçütüne sahip konumu bulmak için cv2.minMaxLoc() fonksiyonu kullanılır.
* Bu fonksiyon, en yüksek benzerlik değerini (max\_val) ve konumunu (max\_loc) döndürür.
* En yüksek benzerlik değeri, en iyi eşleşmeyi temsil eder.

1. **Eşleşme Sonucunun Kullanılması:**

* En iyi eşleşme sonucunu kullanabilirsiniz.
* Örneğin, en iyi eşleşme konumunu işaretleyebilir ve görsel üzerinde bir dikdörtgen veya başka bir şekil kullanarak görsel üzerindeki eşleşmeyi vurgulayabilirsiniz.
* Eşleşme sonucunu kullanarak nesnenin konumunu, boyutunu veya diğer özelliklerini belirleyebilirsiniz.
* Eşleşme sonuçlarını filtreleyebilir ve belirli bir benzerlik eşiği üzerindeki eşleşmeleri alabilirsiniz.
* Birden fazla şablon kullanarak çoklu eşleşmeleri bulabilirsiniz. Bu durumda, farklı şablonlar için ayrı ayrı template matching işlemleri gerçekleştirilir.
* Eşleşme sonuçlarını sıralayabilir ve benzerlik derecesine göre en iyi eşleşmeleri elde edebilirsiniz.

OpenCV Template Matching, nesne takibi, görüntü sınıflandırma, yüz tanıma gibi birçok uygulamada kullanılan önemli bir görüntü işleme tekniğidir. Ancak, bazı zorlukları da beraberinde getirebilir.

Örneğin, aydınlatma değişiklikleri, dönme veya ölçek değişiklikleri gibi faktörler template matching performansını etkileyebilir. Bu nedenle, uygulama senaryosuna bağlı olarak diğer daha gelişmiş yöntemler tercih edilebilir.

Aşağıda, OpenCV Template Matching işlemini gerçekleştirmek için basit bir örnek kod bulunmaktadır:

metin, ekran görüntüsü, yazılım, bilgisayar içeren bir resim

Açıklama otomatik olarak oluşturuldu

**OpenCV Template Matching vs. Diğer Yöntemler**

OpenCV Template Matching yöntemi ve diğer eşleştirme yöntemleri arasında bazı temel farklar bulunmaktadır. İşte bu farklardan bazıları:

1. **Template Matching:**

* Template Matching yöntemi, belirli bir şablonun bir görüntü içindeki konumunu bulmak için kullanılır.
* Şablon görüntüsü, aranan deseni veya nesneyi temsil eder.
* Template Matching işlemi, şablonu hedef görüntü üzerinde kaydırarak benzerlik ölçütlerini hesaplar.
* Benzerlik ölçütleri, şablon ile hedef görüntü arasındaki benzerlik derecesini temsil eder.
* Template Matching yöntemi, basit ve doğrudan bir yaklaşımdır ve hesaplama açısından hızlıdır.
* Ancak, aydınlatma değişiklikleri, dönme veya ölçek değişiklikleri gibi faktörler template matching performansını etkileyebilir.

1. **Özellik Tabanlı Eşleştirme:**

* Özellik tabanlı eşleştirme yöntemleri, görüntülerdeki özellikleri algılayarak eşleşmeleri bulur.
* Özellikler, görüntülerdeki benzersiz desenler veya yapılar olarak düşünülebilir.
* Özellikler, nokta, kenar, köşe gibi çeşitli şekillerde olabilir ve genellikle tanımlayıcı vektörler veya tanımlayıcılar olarak temsil edilir.
* Özellik tabanlı eşleştirme yöntemleri, önceki öğrenme aşamaları veya özellik çıkarımı adımları gerektirebilir.
* Eşleşme aşamasında, özellik tabanlı yöntemler genellikle eşleşme puanı veya benzerlik ölçüsü kullanır.
* Özellik tabanlı eşleştirme yöntemleri, daha genel ve karmaşık eşleştirme senaryolarında daha iyi performans gösterebilir.

1. **Yapılandırılmış Işık Eşleştirme:**

* Yapılandırılmış ışık eşleştirme yöntemleri, görüntülerin yüzey yapısını ve yüzeyin yansıma özelliklerini kullanarak eşleşmeleri bulur.
* Bu yöntemler, genellikle 3D nesnelerin eşleştirilmesinde kullanılır.
* Yapılandırılmış ışık eşleştirme yöntemleri, özel aydınlatma düzenekleri kullanır ve yüzeyin farklı açılardan alınan görüntülerini analiz eder.
* Bu yöntemler, yüzeyin şekil, doku ve yansıma özellikleri gibi bilgileri kullanarak eşleşmeleri bulur.

1. **İteratif En Yakın Nokta Eşleştirme (Iterative Closest Point - ICP):**

* İteratif En Yakın Nokta Eşleştirme (ICP), 3D nokta bulutları veya 3D modeller arasında eşleştirmeleri bulmak için kullanılan bir yöntemdir.
* ICP, nokta bulutlarının birbirine hizalanması için iteratif olarak çalışır.
* İlk olarak, başlangıç hizalaması yapılır ve ardından en yakın nokta eşleştirmeleri bulunur.
* Bu eşleştirmeler kullanılarak iki nokta bulutu arasındaki dönme ve çevirme dönüşümleri tahmin edilir ve hizalama geliştirilir.
* İşlem, eşleşme hatası minimize edilene kadar tekrarlanır.
* ICP yöntemi, 3D nesne tarama, nesne takibi veya haritalama gibi uygulamalarda kullanılır.

1. **Hesaplama Karesel Eşleştirme (Normalized Cross-Correlation - NCC):**

* Hesaplama Karesel Eşleştirme (NCC), iki görüntü arasındaki benzerliği bulmak için kullanılan bir yöntemdir.
* NCC, iki görüntü arasındaki her bir pikselin ortalamasını ve standart sapmasını kullanarak benzerlik ölçütünü hesaplar.
* Bu yöntem, genellikle görüntü hizalaması, stereo görüntü eşleştirmesi veya görüntü kalibrasyonu gibi uygulamalarda kullanılır.
* NCC, görüntülerin ışık değişikliklerine karşı dirençli olabilir, çünkü piksel değerlerinin normalizasyonu kullanılır.
* Yüksek benzerlik değeri, iki görüntünün birbirine daha çok benzediğini gösterir.

Bu farklı eşleştirme yöntemleri, farklı uygulama senaryolarına ve görüntü işleme problemlerine yönelik çeşitli avantajlar ve dezavantajlar sunar. Her yöntemin performansı ve uygunluğu, kullanım senaryosuna, veri tipine ve istenen sonuçlara bağlı olarak değişebilir. Uygulama gereksinimlerine göre doğru yöntemin seçilmesi önemlidir. OpenCV, farklı eşleştirme yöntemlerini destekler ve bu yöntemleri uygulamanızda kullanabilmeniz için işlevler sağlar.

**Ödev Karmaşıklık Analizi**

1. İlk olarak, verilen görüntü dizinindeki tüm dosya yollarını "image\_paths" değişkenine depolamak için "glob.glob" işlevi kullanılır. Bu işlem, "image\_directory" değişkeninde belirtilen klasördeki tüm PNG dosyalarını alır. Bu işlem O(n) karmaşıklığa sahiptir, burada "n", görüntü dizinindeki toplam dosya sayısını temsil eder.
2. Ardından, boş bir "images" listesi oluşturulur. Bu liste, işlenecek görüntülerin depolanacağı yerdir.
3. "image\_paths" listesindeki her bir dosya yolu için bir döngü başlatılır.
   1. Her döngü adımında, "cv2.imread" işlevi kullanılarak görüntü dosyası okunur ve "img" değişkenine atanır. Bu işlem O(1) karmaşıklığına sahiptir, çünkü her görüntüyü tek tek işliyoruz.
   2. Ardından, "img" görüntüsü "images" listesine eklenir. Bu işlem O(1) karmaşıklığına sahiptir, çünkü listeye bir öğe eklemek bir sabit zaman işlemidir.
   3. Döngü, tüm dosya yolları için tekrarlanır. Dolayısıyla, bu adım toplamda O(n) karmaşıklığına sahiptir.
4. "images" listesindeki görüntüler, "find\_similar\_images" fonksiyonuna ve "template\_size" parametresine iletilir.
5. "find\_similar\_images" fonksiyonu içinde, başlangıçta boş bir "similarities" listesi oluşturulur. Bu liste, benzerlik oranları ve ilgili görüntülerin indeksleriyle birlikte tutulacaktır.
6. "combinations" fonksiyonu kullanılarak, "images" listesindeki görüntülerin tüm kombinasyonları elde edilir. Bu adım, verilen n sayısına bağlı olarak O(n^2) karmaşıklığına sahiptir, çünkü her bir görüntüyü diğer tüm görüntülerle eşleştirmemiz gerekiyor.
7. İç içe döngü kullanarak, tüm görüntü kombinasyonları üzerinde işlem yapılır.
8. İlk olarak, "img1\_index" ve "img2\_index" değişkenlerine, kombinasyonun ilgili indeksleri atanır. Bu işlem O(1) karmaşıklığına sahiptir.
9. Daha sonra, "img1" ve "img2" değişkenlerine, "images" listesindeki ilgili görüntüler atanır. Bu işlem O(1) karmaşıklığına sahiptir, çünkü listeye erişim sabit zamanlıdır.
10. "cv2.cvtColor" işlevi kullanılarak, her iki görüntü de gri tonlamaya dönüştürülür. Bu işlem O(p\*q) karmaşıklığına sahiptir, burada "p" ve "q" template boyutunu temsil eder.
11. "cv2.resize" işlevi kullanılarak, her iki görüntü de belirtilen "template\_size" boyutuna yeniden boyutlandırılır. Bu işlem O(p\*q) karmaşıklığına sahiptir, çünkü görüntülerin boyutunu değiştirirken her pikseli işlemlememiz gerekiyor.
12. "cv2.matchTemplate" işlevi kullanılarak, yeniden boyutlandırılmış görüntüler arasında template eşleştirmesi yapılır. Bu işlem O(p\*q) karmaşıklığına sahiptir, çünkü her piksel için template eşleştirmesi hesaplanır.
13. Son olarak, "similarities" listesine, görüntü indeksleri ve benzerlik oranı eklenir. Bu işlem O(1) karmaşıklığına sahiptir, çünkü her kombinasyon için bir öğe eklemek sabit zamanlıdır.
14. Tüm kombinasyonlar işlendikten sonra, "similarities" listesi benzerlik yüzdelerine göre sıralanır. Bu sıralama işlemi O(m log m) karmaşıklığına sahiptir, burada "m" toplam kombinasyon sayısını temsil eder.
15. Son olarak, sıralanmış benzerlikleri içeren "similar\_images" listesi döndürülür. Bu işlem O(1) karmaşıklığına sahiptir, çünkü listedeki tüm öğeleri döndürmek bir sabit zaman işlemidir.

Ana programın karmaşıklığı:

Ana programda yapılan işlemler şunlardır:

1. "glob.glob" işlevi kullanılarak, görüntü dizinindeki dosya yolları elde edilir. Bu işlem O(n) karmaşıklığına sahiptir.
2. "images" listesi oluşturulurken, her görüntü için "cv2.imread" işlevi kullanılır. Bu işlem O(n) karmaşıklığına sahiptir.

3. "find\_similar\_images" fonksiyonu çağrılırken, görüntüler ve template boyutu parametre olarak iletilir. Bu fonksiyonun karmaşıklığı, içindeki kombinasyon işlemi nedeniyle O(n^2) şeklindedir.

1. Benzerlik hesaplaması ve sıralama işlemi, "find\_similar\_images" fonksiyonunda gerçekleşir. Bu işlemler, kombinasyon sayısı ve template boyutuna bağlı olarak O(n^2 \* p\*q \* log(n^2)) karmaşıklığına sahiptir.
2. Son olarak, "similar\_images" listesindeki benzerliklerin her biri için döngü oluşturulur ve sonuçlar ekrana yazdırılır. Bu işlem, "similar\_images" listesindeki öğelerin sayısına bağlı olarak O(n^2) karmaşıklığına sahiptir.

Bu şekilde, ana programın toplam karmaşıklığı şu şekilde ifade edilebilir:

O(n) + O(n) + O(n^2 \* p\*q \* log(n^2)) + O(n^2)

Bu karmaşıklık analizi, "find\_similar\_images" fonksiyonunun içindeki kombinasyon işlemi ve template eşleştirmesi işlemleri nedeniyle genellikle daha belirleyici olan O(n^2 \* p\*q \* log(n^2)) terimiyle temsil edilir.