**تحليل خوارزمية Watershed: فن فصل الكائنات المتلامسة**

**مقدمة: تحدي رؤية الحدود**

في عالم معالجة الصور الرقمية، يكمن أحد أبرز التحديات في قدرة الحاسوب على "رؤية" وفصل الكائنات المتجاورة. كيف يمكننا أن نعلّم برنامجًا أن يرى الحدود الدقيقة بين مجموعة من الخلايا الحية المتلاصقة، أو أن يميّز كل قطعة نقدية في كومة متراصة؟

هنا يأتي دور خوارزمية **Watershed**، وهي تقنية ذكية مستوحاة من الطبيعة، تم تصميمها خصيصًا لمواجهة هذا التحدي. تبرز قوتها الحقيقية عندما تفشل الخوارزميات الأخرى، حيث تتخصص في رسم الخطوط الفاصلة في أصعب الظروف، محولةً مشهداً من الكائنات المتشابكة إلى مجموعة من العناصر المحددة والواضحة.

**جوهر الخوارزمية: رحلة عبر تضاريس الصورة**

لفهم الآلية العبقرية وراء Watershed، دعنا نتخيل الصورة ليس كسطحٍ مستوٍ، بل **كخريطة تضاريس ثلاثية الأبعاد**. في هذه الخريطة:

* المناطق الداكنة (ذات السطوع المنخفض) هي **الوديان العميقة**.
* المناطق الساطعة (ذات السطوع العالي) هي **قمم الجبال الشاهقة**.

الآن، تبدأ رحلة "الفيضان":

1. نبدأ بصب الماء الافتراضي في أدنى نقطة في كل "وادي".
2. مع ارتفاع منسوب المياه، يبدأ كل وادٍ بالامتلاء ليشكل ما نسميه "حوض تجميع" أو بحيرة.
3. عندما توشك مياه حوضين مختلفين على الالتقاء عند نقطة ما، نقوم فورًا ببناء **"سد"** لمنعها من الاندماج.
4. نستمر في هذه العملية حتى تغمر المياه الخريطة بأكملها.

في النهاية، شبكة "السدود" التي قمنا ببنائها ليست سوى **الحدود الدقيقة** التي تفصل بين الكائنات المختلفة في الصورة.

**من النظرية إلى التطبيق: استراتيجية التنفيذ**

تطبيق هذه الفكرة مباشرة على صورة حقيقية قد يؤدي إلى فوضى؛ فالضوضاء الطفيفة في الصورة ستخلق آلاف الوديان الصغيرة، مما ينتج عنه "تجزئة مفرطة". لذلك، نتبع استراتيجية مدروسة من خمس خطوات لضمان نتيجة دقيقة:

1. **أولاً، نقوم بتهيئة المشهد (المعالجة الأولية):** نحول الصورة إلى الأبيض والأسود باستخدام تقنية (Otsu's Thresholding) لعزل الكائنات (باللون الأبيض) عن الخلفية (باللون الأسود) بشكل فعال.
2. **بعد ذلك، نحدد ما هو "خلفية" بالتأكيد:** نقوم بتوسيع مساحة الكائنات البيضاء قليلاً. ما يتبقى من مساحة سوداء بعد هذا التوسيع هو بالتأكيد جزء من الخلفية.
3. **ثم، نحدد "قلب" كل كائن:** نستخدم تقنية **"تحويل المسافة" (Distance Transform)** التي تحدد مركز كل كائن أبيض. هذه المراكز هي نقاط البداية المؤكدة التي سيبدأ منها "الفيضان".
4. **نترك منطقة للحيرة (المنطقة المجهولة):** المنطقة الرمادية بين "الخلفية المؤكدة" و"قلب الكائن المؤكد" هي المنطقة التي لا نعرف تماماً أين تنتهي حدودها. هذه هي المنطقة التي ستعمل عليها الخوارزمية.
5. **أخيراً، نبدأ الفيضان:** نُعطي لكل "قلب كائن" علامة فريدة، ونطلب من خوارزمية Watershed أن تبدأ عملية الملء من هذه العلامات، وأن تبني "السدود" في المنطقة المجهولة.

**تطبيق عملي: تحدي فصل القطع النقدية**

لا يوجد مثال يوضح قوة هذه الخوارزمية أفضل من مشكلة فصل القطع النقدية المتلامسة. فمن خلال تطبيق الاستراتيجية السابقة، نرى كيف تنجح الخوارزمية في رسم الحدود بدقة متناهية حول كل قطعة، محققةً ما يصعب على العين البشرية أحيانًا تمييزه.

الكود

import cv2

import numpy as np

from matplotlib import pyplot as plt

# --- الخطوة 0: تحميل الصورة ---

# قراءة الصورة التي تحتوي على كائنات متلامسة

# تأكد من وجود صورة باسم 'coins.png' في نفس المجلد

original\_image = cv2.imread('water\_coins.png')

img = original\_image.copy() # نأخذ نسخة للعمل عليها

# التأكد من تحميل الصورة بنجاح

if img is None:

    print("خطأ: لم يتم العثور على ملف الصورة 'coins.png'. تأكد من وجوده في نفس المجلد.")

else:

    # تحويلها إلى تدرج رمادي للتحليل

    gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

    # --- الخطوة 1: المعالجة الأولية للحصول على صورة ثنائية ---

    # تطبيق Otsu's Threshold لعزل القطع النقدية عن الخلفية

    ret, thresh = cv2.threshold(gray, 0, 255, cv2.THRESH\_BINARY\_INV + cv2.THRESH\_OTSU)

    # --- الخطوة 2: تحديد الخلفية المؤكدة (sure background) ---

    # إزالة أي ضوضاء صغيرة باستخدام Opening

    kernel = np.ones((3,3), np.uint8)

    opening = cv2.morphologyEx(thresh, cv2.MORPH\_OPEN, kernel, iterations=2)

    # توسيع (dilate) مساحة الكائنات لنجد المنطقة التي هي بالتأكيد خلفية

    sure\_bg = cv2.dilate(opening, kernel, iterations=3)

    # --- الخطوة 3: تحديد الكائن المؤكد (sure foreground) ---

    # تحويل المسافة يعطينا قيم أعلى في مراكز الكائنات

    dist\_transform = cv2.distanceTransform(opening, cv2.DIST\_L2, 5)

    # بأخذ 70% من أقصى قيمة مسافة، نحصل على نوى الكائنات

    ret, sure\_fg = cv2.threshold(dist\_transform, 0.7 \* dist\_transform.max(), 255, 0)

    sure\_fg = np.uint8(sure\_fg) # تحويلها إلى نوع بيانات مناسب

    # --- الخطوة 4: تحديد المنطقة المجهولة (unknown region) ---

    # هي المنطقة بين الخلفية المؤكدة والكائن المؤكد

    unknown = cv2.subtract(sure\_bg, sure\_fg)

    # --- الخطوة 5: إنشاء العلامات (Markers) وتطبيق Watershed ---

    # إعطاء أرقام فريدة لكل نواة كائن (1, 2, 3...)

    ret, markers = cv2.connectedComponents(sure\_fg)

    # إضافة 1 لجميع العلامات حتى تكون الخلفية المؤكدة 1 بدلاً من 0

    markers = markers + 1

    # الآن، نجعل المنطقة المجهولة 0

    markers[unknown == 255] = 0

    # تطبيق خوارزمية Watershed

    cv2.watershed(img, markers)

    # تلوين الحدود باللون الأحمر على الصورة

    img[markers == -1] = [255, 0, 0]

    # --- عرض النتائج ---

    plt.figure(figsize=(12, 8))

    plt.subplot(1, 2, 1)

    plt.imshow(cv2.cvtColor(original\_image, cv2.COLOR\_BGR2RGB))

    plt.title('الصورة الأصلية')

    plt.axis('off')

    plt.subplot(1, 2, 2)

    plt.imshow(cv2.cvtColor(img, cv2.COLOR\_BGR2RGB))

    plt.title('الصورة بعد التجزئة')

    plt.axis('off')

    plt.show()

