



report midterm



Github <https://github.com/Quy3nsss/Image-precessing-mid-report.git>

Bài toán Finding Object detection.

Giới thiệu

Template Matching

TM_SQDIFF_NORMED + MASK

Tỉ lệ giữa template và object.

Muti scale: xác định tỉ lệ giữa template và ảnh.

kết quả khi tiền xử lý với **Gray scale**. (Template và Image)

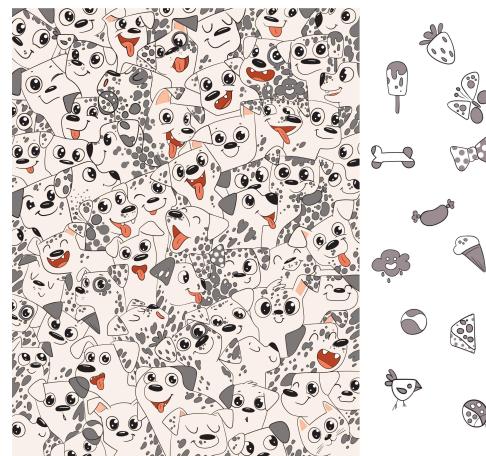
kết quả khi tiền xử lý với **Gray scale + Gaussian + Sobel Operator**. (Template và Image)

Nhận xét chung.

Bài toán Finding Object detection.



FIND 12 HIDDEN OBJECTS IN THE PICTURE



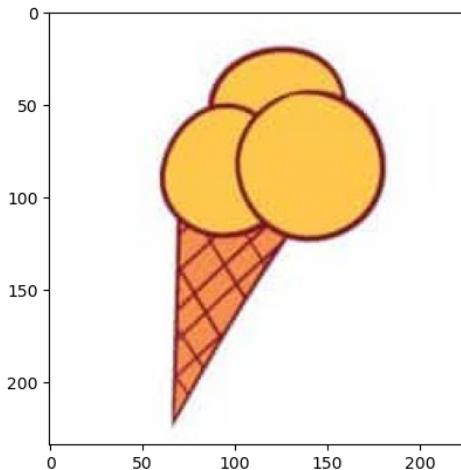
Giới thiệu

Báo cáo này trình bày một phương pháp phát hiện đối tượng sử dụng kỹ thuật template matching trong OpenCV. Thuật toán có khả năng nhận diện nhiều đối tượng đã định nghĩa trước trong một ảnh đích bằng cách so sánh chúng với một tập hợp các ảnh mẫu (template) ở các tỷ lệ khác nhau.

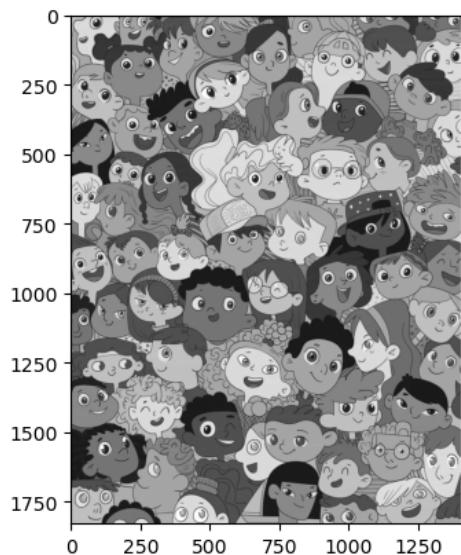
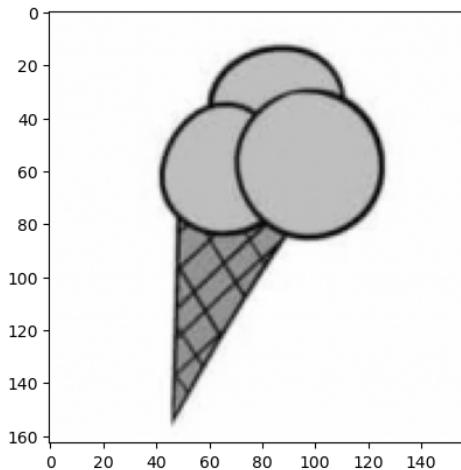
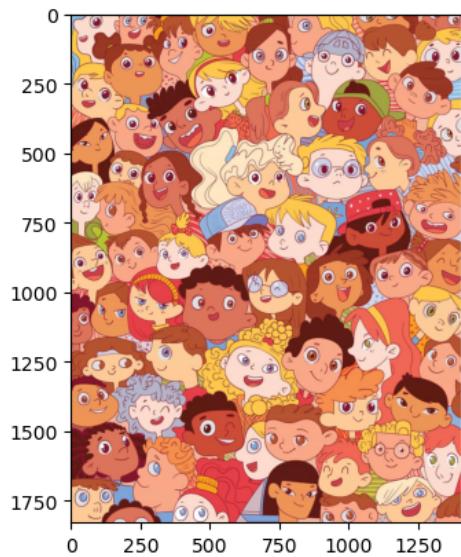


- **Ảnh đích:** Một ảnh đã cắt
- **Các template:** các template đối tượng khác nhau

- Sử dụng code để cắt các Template ra khỏi ảnh mẫu.



- Tách ảnh mục tiêu ra.



Template Matching



◆ matchTemplate()

```
void cv::matchTemplate (InputArray image,  
                      InputArray templ,  
                      OutputArray result,  
                      int method,  
                      InputArray mask = noArray() )  
  
Python:  
cv.matchTemplate( image, templ, method[, result[, mask]]) -> result
```

Template matching cho phép chúng ta phát hiện vật thể trong ảnh đầu vào bằng cách sử dụng một ảnh tham chiếu (template) chứa vật thể cần phát hiện. Cách thuật toán hoạt động:

- Trượt ảnh template trên ảnh đầu vào (giống tích chập 2D)
- Ảnh template sẽ được so sánh với cửa sổ trượt tương ứng trên ảnh đầu vào bằng một công thức.

Ưu điểm:

- Nhanh, đơn giản
- Không tốn công làm data

Nhược điểm

- Template phải rất giống với vật thể trong ảnh cả về kích thước độ nghiêng, ... Nếu khác biệt quá lớn sẽ không phát hiện được.

TM_SQDIFF_NORMED + MASK



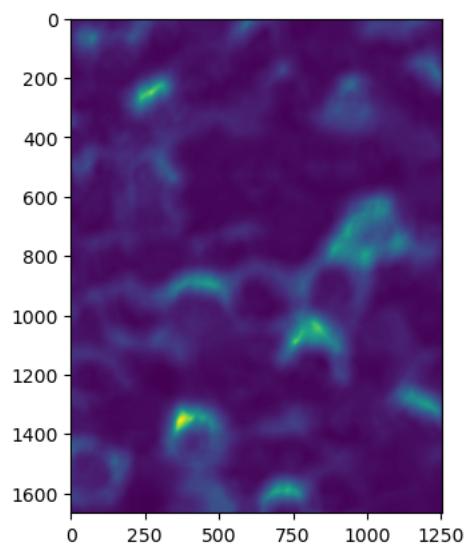
- Sử dụng phương thức TM_SQDIFF_NORMED để đo sự khác biệt bình phương giữa template và ảnh
- kết hợp với mask để loại bỏ background trắng.

TM_SQDIFF_NORMED
Python: cv.TM_SQDIFF_NORMED

$$R(x, y) = \frac{\sum_{x',y'}(T(x', y') - I(x + x', y + y'))^2}{\sqrt{\sum_{x',y'} T(x', y')^2 \cdot \sum_{x',y'} I(x + x', y + y')^2}}$$

with mask:

$$R(x, y) = \frac{\sum_{x',y'}((T(x', y') - I(x + x', y + y')) \cdot M(x', y'))^2}{\sqrt{\sum_{x',y'}(T(x', y') \cdot M(x', y'))^2 \cdot \sum_{x',y'}(I(x + x', y + y') \cdot M(x', y'))^2}}$$



Tỉ lệ giữa template và object.



Template được cắt và đối tượng cần tìm trong ảnh có kích thước khác nhau. Dẫn tới kết quả tìm kiếm đối tượng diễn ra với kết quả không mong muốn.



⇒ Giải pháp là Muti scale.

Muti scale: xác định tỉ lệ giữa template và ảnh.



Mutiscale.

Phương pháp multi-scale trong mã nguồn của bạn là một cách tiếp cận thông minh để giải quyết một hạn chế quan trọng của template matching truyền thống.

- **Tính bất biến với tỷ lệ (Scale Invariance):** Template matching cơ bản rất nhạy cảm với kích thước, nhưng bằng cách thử template ở nhiều tỷ lệ khác nhau (0.5 đến 1.0), thuật toán có thể phát hiện đối tượng ở các kích thước khác nhau.
- **Độ chính xác cao hơn:** Bằng cách thử 20 tỷ lệ khác nhau, thuật toán có khả năng cao hơn trong việc tìm ra kích thước template phù hợp nhất với đối tượng trong ảnh.
- **Hiệu quả trong việc xử lý đối tượng đa kích thước:** Đặc biệt hữu ích khi không biết trước kích thước chính xác của đối tượng cần tìm trong ảnh đích.

kết quả khi tiền xử lý với Gray scale.(Template và Image)



Sau khi áp dụng **Gray scale + Template Matching + Muti scale + mask + TM_SQDIFF_NORMED**

→ ảnh 1 nhận diện được 13/15; ảnh 2: 12/12

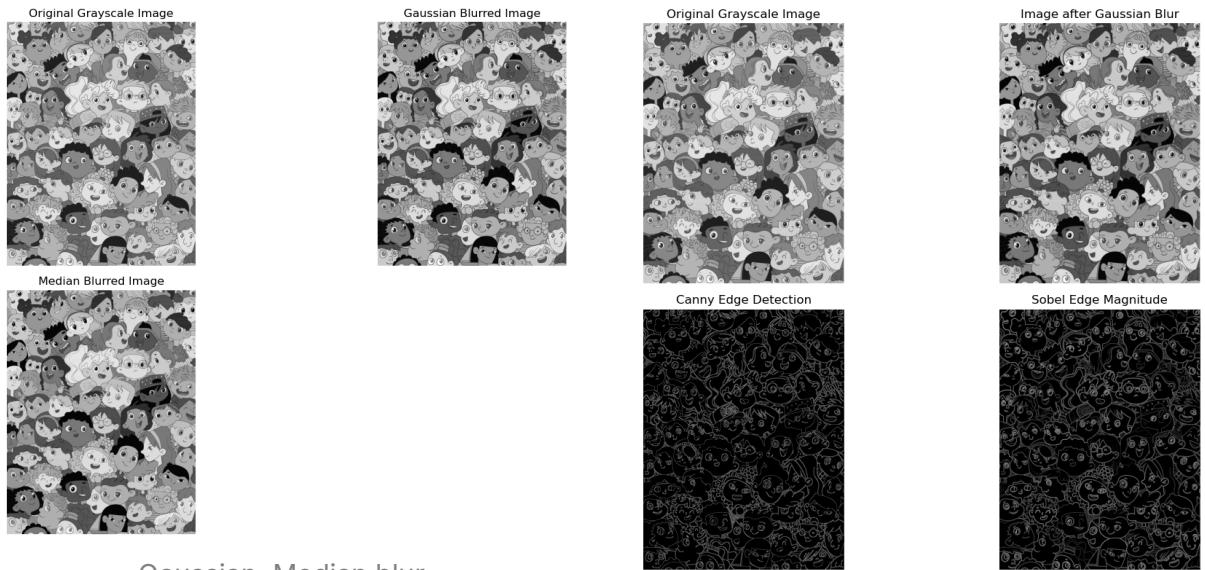


kết quả khi tiền xử lý với Gray scale + Gaussian + Sobel Operator.(Template và Image)



Qua thử nghiệm với Gaussian, Median blur ; Candy , Sobel ⇒ nhận thấy được với

Gaussian + Sobel Operator cho ra kết quả sát và phù hợp hơn.



Gaussian, Median blur

Gaussian blur + Candy or Sobel



Thực hiện xử lý ảnh 1 thêm Gaussian + Sobel Operator

- **Làm mờ Gaussian:** Sử dụng bộ lọc Gaussian với kernel $(3,3)$ để giảm nhiễu.
 - **Phát hiện biên:** Sử dụng **Sobel Operator** để tính gradient theo hai trục X và Y, sau đó kết hợp chúng.

⇒ Đã nhận diện được 15/15



Nhận xét chung.



Tùy vào ảnh đầu vào mà ta cần chọn phương pháp xử lý phù hợp với phương pháp tìm kiếm mà ta đang sử dụng.

- Với ảnh 2 chỉ cần xử lý ảnh với Gray scale cho qua Template Matching sử dụng TM_SQDIFF_NORMED + MASK \Rightarrow Tìm kiếm được 12/12
- Với ảnh 1 xử lý ảnh với Gray scale cho qua Template Matching sử dụng TM_SQDIFF_NORMED + MASK \Rightarrow Tìm kiếm được 13/15
- Sau khi xử lý thêm với **Gaussian + Sobel Operator** \Rightarrow Tìm kiếm được 15/15