VIETNAM NATIONAL UNIVERSITY
UNIVERSITY OF ENGINEERING AND TECHNOLOGY

MID-TERM REPORT

Image Identification Using Scale-Invariant Feature Transform (SIFT) Algorithm

Môn học	Xử lý ảnh INT3304 20
Ngày	22.10.2020
Sinh viên	Lê Minh Tâm
Mã số sinh viên	17021332

Bài báo cáo sẽ chia làm các phần sau

1. Template Matching

1.1 Định nghĩa:

Template Matching là một phương pháp sử dụng để tìm kiếm một vật cho sẵn. Phương pháp này sử dụng bằng cách so sánh sự giống nhau theo vùng giữa ảnh mẫu và ảnh cần tìm.

1.2 Ý tưởng:

Đầu vào:

- Source Image – một bức ảnh chứa đối tượng được quan tâm;
- Template Image – một "object patch" được dùng để tìm kiếm trong Source Image.

Phương pháp:

Theo kỹ thuật căn bản của phương pháp, ảnh mẫu sẽ được so sánh với từng vùng trong ảnh cần tìm, đi qua mỗi vùng sẽ tính toán số liệu độ phù hợp với giữa vùng ảnh mẫu và vùng đó

Đầu ra:

Vùng trên ảnh chứa mẫu giống template image nhất

1.3 Cài đặt

Để sử dụng Template Matching trở nên dễ dàng, OpenCV cung cấp function **cv2.matchTemplate()**

result = cv2.matchTemplate(source, template, method)

Trong đó:

- *source* – ảnh chứa đối tượng cần tìm;
- *template* – ảnh mẫu;
- *method* – phương pháp so sánh (cài đặt sẵn trong cv2), gồm *CCOEFF, SQDIFF, CCORR*,...;
- *result* – ma trận chứa kết quả tính toán tại mỗi vị trí để thể hiện mức độ phù hợp của ảnh mẫu cho trước.

Trích xuất được tọa độ của "Best Match" chúng ta sử dụng function *cv2.minMaxLoc()*,

Sử dụng bằng câu lệnh:

minVal, maxVal, minLoc, maxLoc = cv2.minMaxLoc(matrix)

Trong đó:

minVal, maxVal – giá trị nhỏ nhất và lớn nhất của matrix;

minLoc, maxLoc – tọa độ của giá trị *minVal* và *maxVal* trong matrix, ở dạng *(x, y)*.

1.3 Vấn đề

Nhân tố

Những yếu tố liên quan có thể ảnh hưởng đến kết quả của phương pháp Template Matching:

- *Độ lớn*: mức độ thu phóng của ảnh mẫu trong ảnh. (1)
- *Vị trí*: ảnh mẫu có thể đổi hướng trong hệ trục tọa độ. (2)
- *Điểm nhìn*: ảnh mẫu từ các góc nhìn khác nhau. (3)
- *Màu sắc, ánh sáng*: bức ảnh có ảnh hưởng của các buổi trong ngày, hay tùy vào màu ảnh, chất lượng ánh sáng. (4)
- *Vật nhiễu*: ảnh có thể bị tổn hại vì nhiều lý do: ảnh mờ, vật thể méo mó biến dạng. (5)

Lưu ý:

- Những yếu tố kể trên có thể ảnh hưởng đơn lẻ hoặc kết hợp với đối tượng phải tìm trong ảnh hoặc chính ảnh hoặc cả hai
- Danh sách liệt kê trên chỉ có yếu tố ví dụ, trên thực tế còn nhiều điều sai khác nữa.

Lý do:

Template Matching tìm kiếm theo vùng, nên đảm bảo vùng phải trùng khớp, chỉ một khác biệt nhỏ cũng sẽ làm thay đổi kết quả của Template Matching và khiến công cuộc tìm kiếm đi vào bế tắc.

⇒ Như vậy cần phải khắc phục độ giới hạn của phạm vi vùng tìm kiếm, kiểm soát những thay đổi rất thường thấy ở ảnh.

2. Scale Invariant Feature Transform (SIFT)

2.1 Giới thiệu

Scale Invariant Feature Transform (SIFT) là một phương pháp tìm kiếm ảnh mẫu được giới thiệu trong bài “Distinctive image features from scale-invariant keypoints” trên tạp chí International Journal of Computer Vision, 60, 2 vào năm 2004. Thuật toán này dùng để xác định những điểm

đặc trưng của ảnh, so sánh những điểm đặc trưng giữa ảnh mẫu và ảnh cần tìm. Phương pháp đã được đăng ký bản quyền, nếu muốn sử dụng vì mục đích thương mại cần phải có sự đồng ý của tác giả.

2.2 Thuật toán

Ý tưởng:

Ảnh được xử lý và tìm ra các điểm đặc trưng, những điểm này sẽ không bị ảnh hưởng nếu xoay, đảo,...

Thuật toán có thể giới thiệu trong 4 bước sau đây:

- 1. Scale-space Extrema Detection: Tìm kiếm keypoints (kp) lần lượt theo không gian và tỷ lệ thu phóng.
- 2. Key-point Localization: Xác định vị trí và tỷ lệ thu phóng của keypoint. Lọc và chọn lựa lại keypoints ở 1. dựa theo một số tiêu chuẩn đảm bảo chất lượng.
- 3. Orientation Assignment: tính toán tọa độ hướng, chiều thích hợp nhất của các keypoints
- 4. Key-point Description: Dựa vào image gradients ở một mức độ thu phóng xác định và hướng xác định, xác định vùng đặc trưng cho mỗi keypoint

Chi tiết:

- 1. Scale-space Extrema Detection

Mục tiêu: tìm ra vị trí và tỷ lệ thu phóng của ảnh (theo góc nhìn, góc quay,..)

Thành phẩm: danh sách các điểm hơi đặc biệt trong ảnh

Cách thức:

Bước 1: Thiết lập không gian tỷ lệ scale space

Ảnh gốc → Làm mờ → Giảm kích cỡ xuống 1 nửa → Làm mờ

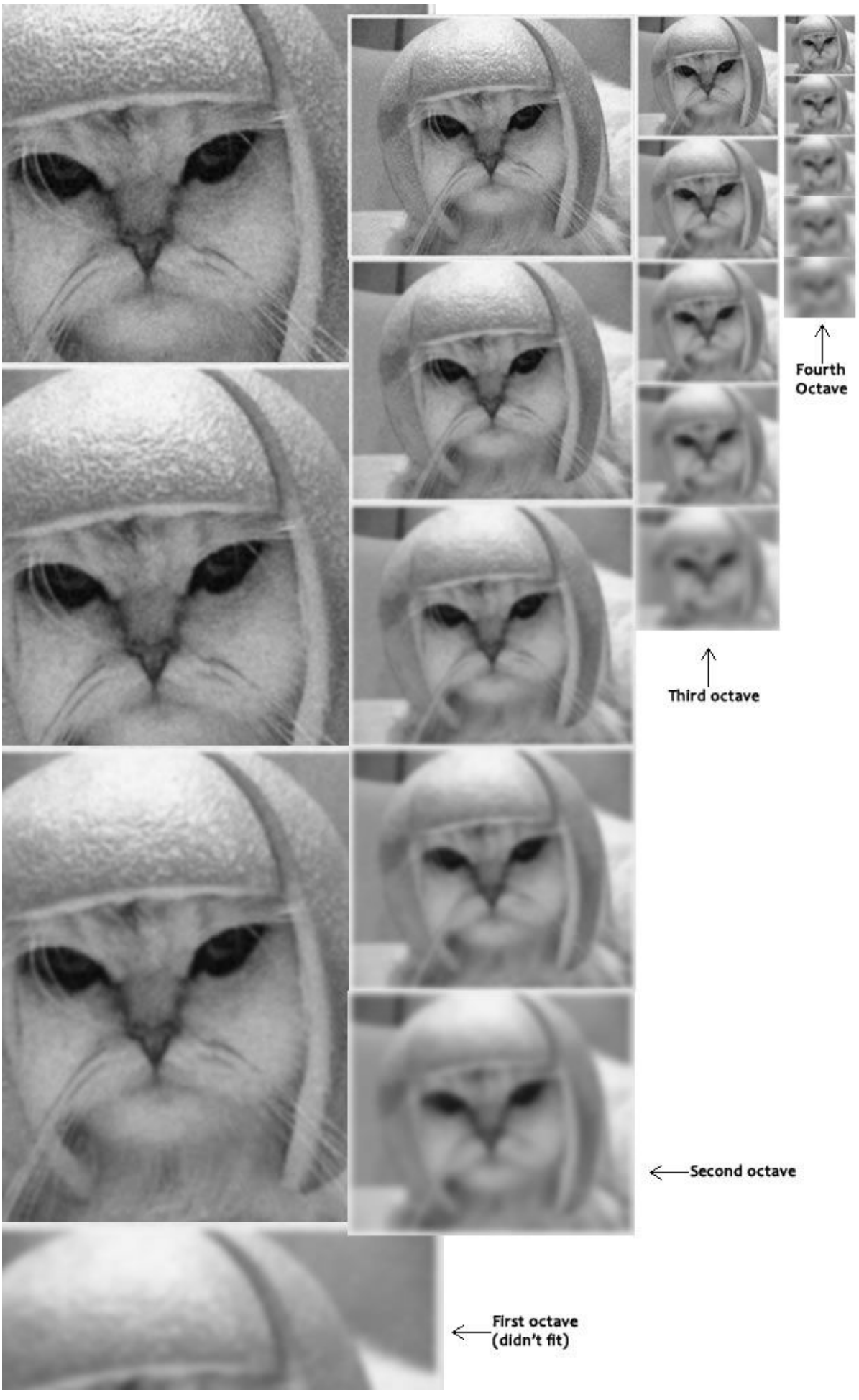
Số vòng lặp lại của quá trình dựa vào kích cỡ ảnh gốc.

Số vòng lặp lý tưởng là 4 với 5 mức làm mờ như ví dụ trong hình

Quy trình làm mờ

Sử dụng Gaussian blur

Ảnh ví dụ sau sẽ mô tả các bước của quy trình

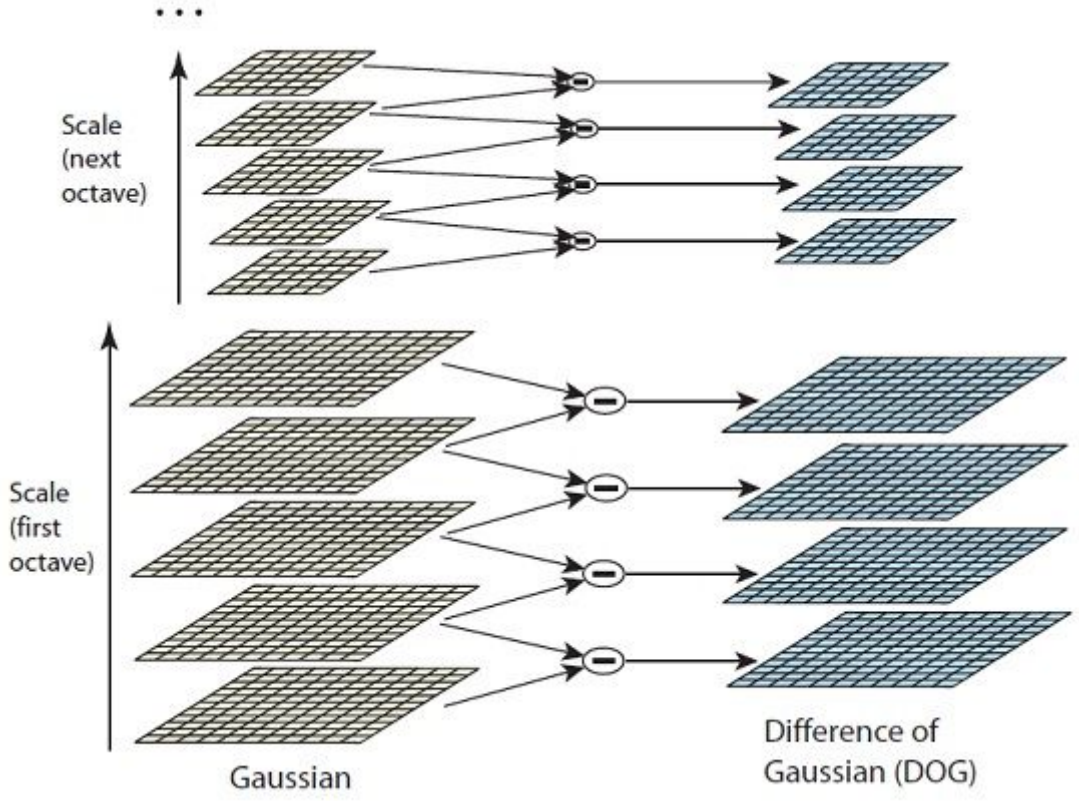

Fourth
Octave
Third octave
Second octave
First octave
(didn't fit)

Bước 2: LoG approximations

Sử dụng ảnh đã làm mờ ở bước trên, gom thành một tập các ảnh, Difference of Gaussians (DoG)

Xác định các góc và cạnh (corners) của ảnh sử dụng kỹ thuật Laplacian of Gaussian (LoG)

Tính toán độ khác biệt của 2 tỷ lệ thu phóng (difference of Gaussians) bằng scale space để cài đặt LoG



- Key-point Localization: Truy tìm keypoints

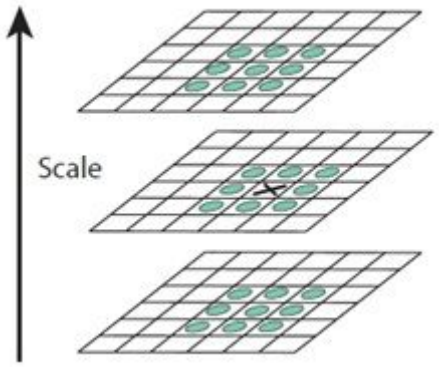
  Bước 1: xác định vị trí và các tỷ lệ keypoints: có 2 công việc:

  - Xác định vị trí cực đại / cực tiểu (maxima/minima) trong DoG

    Xác định vị trí bằng dò từng pixel ảnh và lân cận

    Ví dụ trong hình, X là pixel hiện tại; tròn xanh là lân cận, để kết

quả tốt thì tốt nhất nên có tối thiểu 26 tròn xanh



- Tìm cực đại / cực tiểu của các subpixel
  Sử dụng giá trị của pixel và subpixel, dùng Taylor expansion cho ảnh quanh điểm giá trị xấp xỉ của keypoints

Bước 2: giảm bớt keypoint bằng cách:

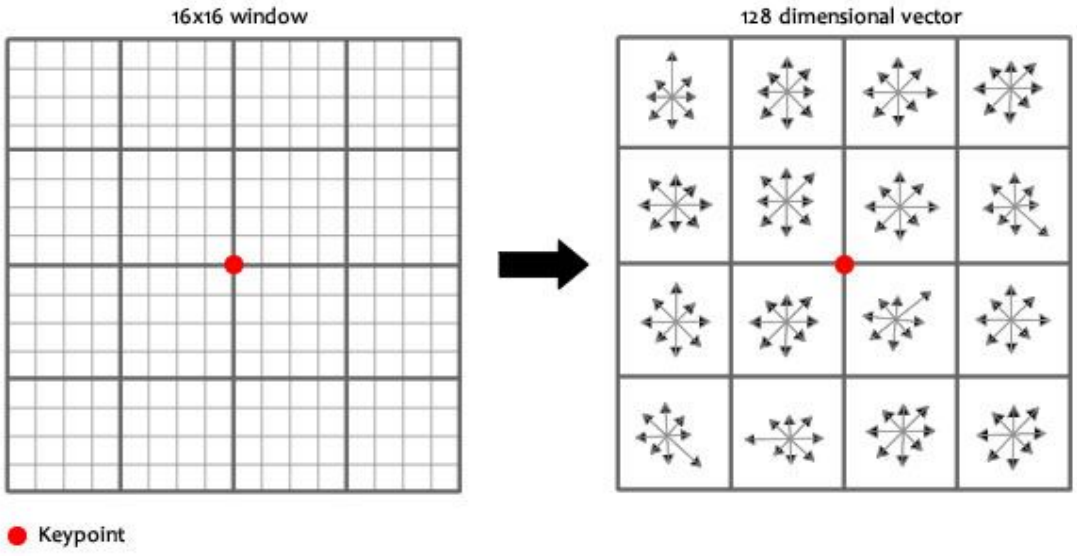
- Lọc những ảnh không đủ tiêu chuẩn về intensity value tại vị trí subpixel tại bước taylor đã tính từ trước
- Loại bỏ cạnh tương đối tương tự với Harris Corner Detector.

- 3. Orientation Assignment
  Tính cường độ và hướng gradient của từng điểm keypoint
  Xem xét để tìm ra đâu là hướng chính của rồi gán cho keypoint tương ứng
  Công thức tính toán lần lượt với cường độ và hướng gradient

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$

$$\theta(x,y) = \tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y)))$$

- 4. Key-point Description
  Tạo ra các thông số đặc biệt cho các keypoints. Không một keypoint được có thông số giống nhau.
  Cách làm
  Vẽ một ô 16*16 cho mỗi keypoint. Sau đó chia ra thành 16 ô 4*4



Ở mỗi ô này, tính toán cường độ và hướng gradient(gradient magnitudes and orientations) sau đó sinh một histogram 8 bins, giá trị thả vào đây.

Mỗi một keypoint ta được 4*4*8 = 128 số, normalize các giá trị này, ta được các feature vector. Như vậy, đặc trưng của keypoint là vector 128 chiều.

2.3 Cài đặt

Phương pháp được hỗ trợ bởi OpenCV, sử dụng, cài đặt bằng các câu lệnh:
Khởi tạo detector: **cv2.xfeatures2d.SIFT_create()**
Tìm key-point: **sift.detect()**
Hiển thị các key-point trên hình: **cv2.drawKeyPoints()**
Tìm điểm giống nhau của keypoint: **cv2.drawMatchesKnn()**

2.4 Áp dụng

Mô tả bài toán:

- Đầu vào: một ảnh mẫu, một dãy ảnh bình thường chứa ảnh mẫu ở nhiều chiều hướng
- Yêu cầu: xác định ảnh mẫu trong ảnh bình thường
- Đầu ra: chỉ vị trí ảnh mẫu.

Cách làm

- Preprocess ảnh
- Áp dụng SIFT cho 2 ảnh.

- So sánh độ giống nhau của các keypoint của từng ảnh bằng giải thuật knn
- Đánh giá điểm khớp trúng
- So sánh điểm ở bước ba với tổng điểm để đưa ra đánh giá
- Từ những điểm tìm được tìm tâm điểm dựa vào k-means clustering algorithm
- Vẽ bounding box

Cài đặt chi tiết:

https://colab.research.google.com/drive/1c_kmYpBF-6HSXaV4Qp3Wi_Jeht8NixHG?authuser=3#scrollTo=5CGdJqqzIdE-

2.5 Cải tiến

So sánh với nhiều bộ dữ liệu khác nhau

Cải thiện Preprocessing ảnh

Sử dụng phương pháp xử lý ảnh màu

3. Tham khảo

[1]: Method and apparatus for identifying scale invariant features in an image and use of same for locating an object in an image
https://patents.google.com/patent/US6711293B1/en
[2]: Lecture 31: Object Recognition - CSE486, Penn State. Robert Collins
https://apps.dtic.mil/dtic/tr/fulltext/u2/786720.pdf
[3]: Introduction to SIFT (Scale-Invariant Feature Transform):
https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html
[4]: OpenCV Template matching and SIFT:
https://github.com/zszazi/OpenCV-Template-matching-and-SIFT
[5]: Implementing SIFT in Python: A Complete Guide (Part 1):
https://medium.com/@russmislam/implementing-sift-in-python-a-complete-guide-part-1-306a99b50aa5
[6]: Implementing SIFT in Python: A Complete Guide (Part 2):
https://medium.com/@russmislam/implementing-sift-in-python-a-complete-guide-part-2-c4350274be2b