ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA CÔNG NGHỆ THÔNG TIN**

**---------- 🙐⯎🙖 ----------**

****

**LAB 3: OBJECT DETECTION USING SIFT**

MÔN HỌC: THỊ GIÁC MÁY TÍNH

|  |  |
| --- | --- |
| ***SINH VIÊN THỰC HIỆN:*** | |
| Nguyễn Bùi Mẫn Nhi | 21127662 |

|  |
| --- |
| ***GIÁO VIÊN MÔN HỌC:*** |
| Thầy Võ Hoài Việt |
| Thầy Phạm Minh Hoàng |
| Thầy Nguyễn Trọng Việt |

**TPHCM – 03/2024**

**MỤC LỤC**

[I. CÁCH SỬ DỤNG: 3](#_Toc161515948)

[1. Cây thư mục của đồ án: 3](#_Toc161515949)

[2. Cách thực thi chạy lệnh: 3](#_Toc161515950)

[II. BÁO CÁO KẾT QUẢ: 4](#_Toc161515951)

[1. Bảng đánh giá kết quả công việc: 4](#_Toc161515952)

[2. Các hàm trong chương trình: 4](#_Toc161515953)

[3. Cách thực hiện: 11](#_Toc161515954)

[4. Nhận xét: 12](#_Toc161515955)

[III. TÀI LIỆU THAM KHẢO: 13](#_Toc161515956)

# **I. CÁCH SỬ DỤNG:**

## 1. Cây thư mục của đồ án:

A screenshot of a computer

Description automatically generated

Trong cây thư mục đồ án, có các folder và file quan trọng:

* Folder data: Folder lưu 2 folder: input (lưu trữ hình ảnh đầu vào) và output (lưu trữ kết quả khi thực thi xử lý hình ảnh).
* Folder sources: Folder chứa các file code và file thực thi.
* File 21127662: File thực thi, được nằm trong folder build.
* File main.cpp: File code.

## 2. Cách thực thi chạy lệnh:

Các bước để chạy code:

* Bước 1: Mở terminal.
* Bước 2: Di chuyển đến nơi chứa thư mục build trong thư mục source.
* Bước 3: Nhập argument command line như sau:

./21127662 <tasks> <input\_path> <output\_path> (<addition\_argument>)

# **II. BÁO CÁO KẾT QUẢ:**

## 1. Bảng đánh giá kết quả công việc:

|  |  |
| --- | --- |
| **Công việc** | **Đánh giá** |
| **Thực hiện triển khai ứng dụng phát hiện đối tượng trong OpenCV C/C++ bằng tính năng SIFT.** | **100%** |
| + Tải hai hình ảnh: mẫu và hình ảnh cảnh từ đường dẫn của chúng. | 100% |
| + Phát hiện các điểm chính trong cả hai hình ảnh và tính toán các bộ mô tả cho từng điểm chính. | 100% |
| + Tìm sự trùng khớp gần nhất giữa các mô tả từ hình ảnh đầu tiên đến hình ảnh thứ hai. | 100% |
| + Lọc để giữ kết quả phù hợp. | 100% |
| + Tìm phép biến đổi đồng nhất giữa hai tập hợp điểm. | 100% |
| **Chương trình được gọi bằng dòng lệnh theo yêu cầu.** | 100% |

## 2. Các hàm trong chương trình:

Các hàm được code từ mã ban đầu:

* void **construct\_scale\_space**(vector<vector<Mat>> &scale\_space\_octaves, Mat &image, int num\_octaves=4, int num\_scales=5, double sigma=1.6, double k=sqrt(2), int kernel\_size=5)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Scale 1 | Scale 2 | Scale 3 | Scale 4 | Scale 5 |
| Octave 1 | A group of books on a table  Description automatically generated | A group of books on a table  Description automatically generatedA group of books on a table  Description automatically generated | A group of books on a table  Description automatically generated | A group of books on a table  Description automatically generated | A group of books on a table  Description automatically generated |
| Octave 2 | A group of books on a table  Description automatically generated | A group of books on a table  Description automatically generated | A group of books on a table  Description automatically generated |  | A group of books on a table  Description automatically generated |
| Octave 3 | A group of books on a table  Description automatically generated |  | A group of books on a table  Description automatically generated | A group of books on a table  Description automatically generated | A group of books on a table  Description automatically generated |
| Octave 4 | A blurry image of a group of boxes  Description automatically generated |  | A blurry image of a group of boxes  Description automatically generated | A blurry image of a group of boxes  Description automatically generated | A blurry image of a group of boxes  Description automatically generated |

* void **compute\_dog\_space(**vector<vector<Mat>> &dog\_space\_octaves, vector<vector<Mat>> &scale\_space\_octaves)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | DoG 1 | DoG 2 | DoG 3 | DoG 4 |
| Octave 1 |  | A group of books on a table  Description automatically generated |  |  |
| Octave 2 |  |  |  |  |
| Octave 3 |  |  |  |  |
| Octave 4 |  |  |  |  |

* bool **filter\_keypoints\_along\_edges**(KeyPoint &kp, Mat &image)

A screenshot of a computer screen

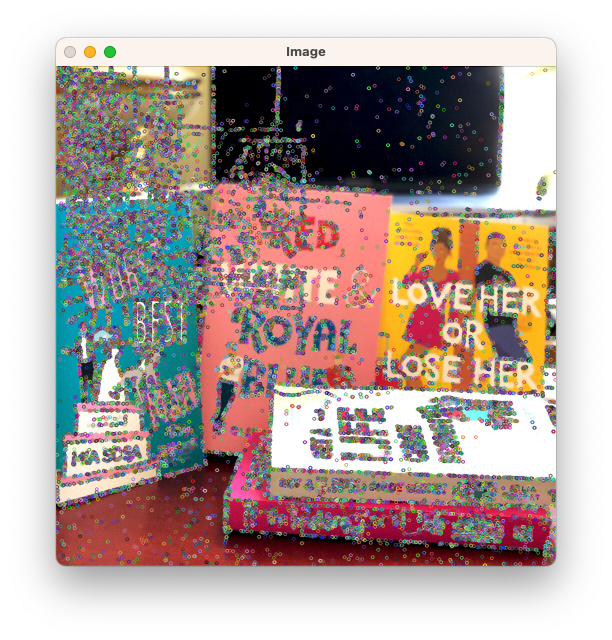
Description automatically generated

* bool **filter\_low\_contrast\_keypoints**(KeyPoint &kp, Mat &image)

A screenshot of a computer screen

Description automatically generated

* void **localize\_keypoints**(vector<KeyPoint> &keypoints, vector<vector<Mat>> &dog\_spaces\_octaves)



* void **assign\_orientation**(vector<KeyPoint> &keypoints, vector<vector<Mat>> &scale\_space\_octaves)

A screenshot of a computer screen

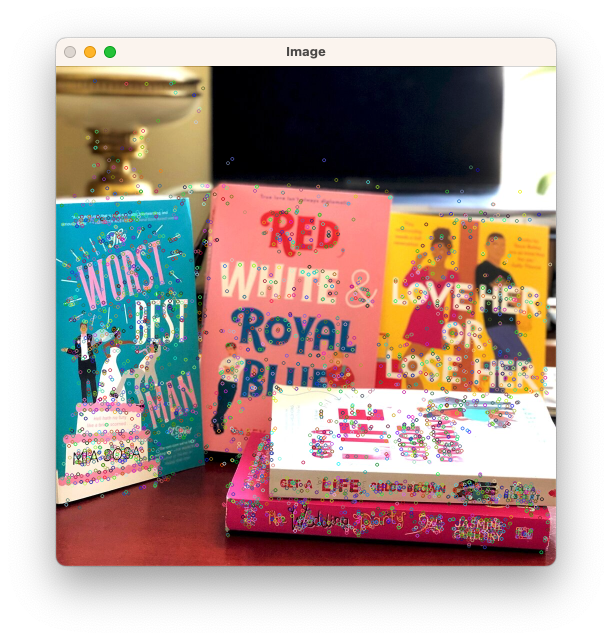
Description automatically generated

* Mat **compute\_descriptor**(vector<KeyPoint> &keypoints, vector<vector<Mat>> &scale\_space\_octaves)
* void **detect\_keypoints\_from\_scratch**(Mat &img\_scene, Mat &img\_template, vector<KeyPoint> &keypoint\_scene, vector<KeyPoint> &keypoint\_template, Mat &descriptor\_scene, Mat &descriptor\_template)

|  |  |
| --- | --- |
| A screenshot of a computer screen  Description automatically generated | A screenshot of a phone  Description automatically generated |

* void **detect\_keypoints\_by\_surf** (Mat &img\_scene, Mat &img\_template, vector<KeyPoint> &keypoint\_scene, vector<KeyPoint> &keypoint\_template, Mat &descriptor\_scene, Mat &descriptor\_template)

|  |  |
| --- | --- |
|  |  |

A blue cover with a couple of people on top of a cake

Description automatically generated

* vector<vector<DMatch>> **match\_descriptors** (Mat &descriptor\_scene, Mat &descriptor\_template)
* vector<DMatch> **filter\_matches** (vector<vector<DMatch>> &knn\_matches)
* void **show\_results**(Mat &image\_scene, Mat &image\_template, vector<KeyPoint> &keypoint\_scene, vector<KeyPoint> &keypoint\_template, vector<DMatch> &good\_matches)

A group of books on a table

Description automatically generated

## 3. Cách thực hiện:

- Bước 1: Phát hiện các điểm chính trong cả hai hình ảnh và tính toán các bộ mô tả cho từng điểm chính:

* Bước 1.1: Chuyển hình ảnh thành dạng grayscale.
* Bước 1.2: Xây dựng kim tự tháp gaussian. Các tầng (octave) được lọc nhiễu bằng Gaussian với các sigma khác nhau. Mỗi octave giảm độ lớn một nửa. Trong bài này, thuật toán code từ mã ban đầu chọn 4 octaves, mỗi octave có 5 scales, với các sigma khác nhau.
* Bước 1.3: Xây dựng DoG. Cách xây dựng bằng cách, trong mỗi octave, lần lượt lấy 2 scale gần kề nhau để trừ.
* Bước 1.4: Xác định các keypoints. Trong mỗi octave, điểm được xem là keypoint khi nó phải là điểm cực đại cục bộ hoặc cực tiểu cục bộ, hay nói cách khác, điểm phải là cực trị 9 điểm ở scale trên, 9 điểm ở scale dưới, và 8 điểm xung quanh cùng scale. Tuy nhiên, nếu như vậy sẽ có rất nhiều điểm, vì thế, ta cần lọc những điểm nằm trên cạnh sử dụng phép toán tương tự harris corner detection để tìm tỉ lệ giữa hai giá trị riêng hoặc contrast sử dụng taylor.
* Bước 1.5: Gán hướng cho các điểm keypoints. Trong điểm keypoint, ta có attribute angle. Ta có hai bước: Tính độ lớn và hướng; Tạo biểu đồ cho độ lớn và hướng. Magnitude = √[(Gx)2+(Gy)2] và Φ = atan(Gy / Gx). Biểu đồ chọn 36 cột, mỗi cột sẽ có độ rộng là 10. Nếu hướng của nó nằm trong khoảng nào, thì giá trị cột cộng thêm độ lớn của điểm. Hướng chính của điểm là hướng có độ lớn cao nhất.
* Bước 1.6: Xây dựng bộ mô tả. Tương tự như gán hướng cho các điểm keypoints, nhưng bộ mô tả có 128 vectors. Trước tiên sẽ lấy vùng lân cận 16×16 xung quanh điểm chính. Khối 16×16 này được chia thành các khối con 4×4 và đối với mỗi khối con này, chúng tôi tạo biểu đồ bằng cách sử dụng độ lớn và hướng. Ở giai đoạn này, kích thước thùng được tăng lên và chúng tôi chỉ lấy 8 bin. Mỗi mũi tên này đại diện cho 8 bin và độ dài của mũi tên xác định độ lớn. Vì vậy, chúng ta sẽ có tổng cộng 128 giá trị bin cho mỗi điểm chính.

- Bước 2: Tìm sự trùng khớp gần nhất giữa các mô tả từ hình ảnh đầu tiên đến hình ảnh thứ hai. Trong bài này, em sử dụng knnMatch để xác định.

- Bước 3: Tìm những điểm tốt nhất để xác định.

- Bước 4: Sử dụng homography để tìm vật. Sau khi tìm sẽ vẽ khung để xác định vật thể đó.

## 4. Nhận xét:

Đối với thuật toán được code từ mã ban đầu, mặc dù vẫn nhận dạng được vật thể trong bức hình, nhưng so với thư viện opencv, thuật toán chưa lọc tốt các điểm, vì thế tốn không gian lưu trữ.

# **III. TÀI LIỆU THAM KHẢO:**

[1] OpenCV documentation on Features2D module: OpenCV. (n.d.). Features2D + Homography to find a known object. OpenCV Documentation. [Online] Available: <https://docs.opencv.org/4.x/d9/d97/tutorial_table_of_content_features2d.html>.

[2] Article on the SIFT technique for image matching by Analytics Vidhya: Analytics Vidhya. (2019, October 14). A Detailed Guide to the Powerful SIFT Technique for Image Matching (with Python code). Analytics Vidhya. [Online] Available: <https://www.analyticsvidhya.com/blog/2019/10/detailed-guide-powerful-sift-technique-image-matching-python/>.