**ĐẠI HỌC ĐÀ NẴNG**
**TRƯỜNG ĐẠI HỌC BÁCH KHOA**
**KHOA ĐIỆN TỬ - VIỄN THÔNG**

# BÁO CÁO BÀI TẬP LỚN SỐ 1

## ĐỀ TÀI: XÂY DỰNG HỆ THỐNG TỰ ĐỘNG PHÁT HIỆN CÂY BÚT BI

## NHÓM 10

| | | |
|---|---|---|
| **Sinh viên thực hiện:** | Đinh Văn Quang | 20DT1 |
| | Hà Phước Phúc | 20DT2 |
| | Nguyễn Văn Qúy | 21DT2 |
| **Lớp học phần:** | 20.38B | |
| **Giảng viên hướng dẫn:** | TS. Hoàng Lê Uyên Thục | |

**Đà Nẵng, 9/2024**

# PHẦN 1. GIỚI THIỆU

## 1.1 Tính cấp thiết

- Tăng cường tự động hóa trong công nghiệp và sản xuất: Hệ thống phát hiện đối tượng như cây bút bi có thể áp dụng trong các quy trình sản xuất và kiểm tra chất lượng.

Ví dụ: trong các dây chuyền sản xuất, phát hiện các vật phẩm cụ thể như bút bi có thể giúp phân loại sản phẩm hoặc kiểm tra số lượng và chất lượng sản phẩm tự động, giảm thiểu sai sót do con người.

- Tối ưu hóa nhận diện và phân loại: Hệ thống tự động phát hiện đối tượng là cây bút bi có thể là tiền đề cho các ứng dụng phức tạp hơn. Nó có thể được mở rộng để phát hiện các đối tượng khác với độ chính xác cao hơn và trong nhiều môi trường khác nhau, giúp tối ưu hóa các hệ thống nhận diện và phân loại trong các ứng dụng như thương mại điện tử (phân loại sản phẩm tự động) hay bán lẻ (kiểm kê hàng hóa).

- Hệ thống phát hiện đối tượng cơ bản có thể được ứng dụng trong các dự án giáo dục, giúp sinh viên và người học nghiên cứu về các thuật toán thị giác máy tính và học máy. Họ có thể sử dụng các dữ liệu đơn giản để xây dựng các hệ thống nhận diện đối tượng với chi phí thấp, phục vụ mục đích nghiên cứu và học tập.

Việc phát triển hệ thống tự động phát hiện đối tượng như cây bút bi, dù với yêu cầu đơn giản về ảnh nền trơn và số lượng dữ liệu huấn luyện ít, vẫn mang lại giá trị lớn trong nhiều ứng dụng thực tiễn. Nó là bước khởi đầu quan trọng trong việc xây dựng các hệ thống nhận dạng tự động với hiệu suất cao và chi phí thấp, đồng thời có thể là tiền đề cho các hệ thống phức tạp hơn trong tương lai.

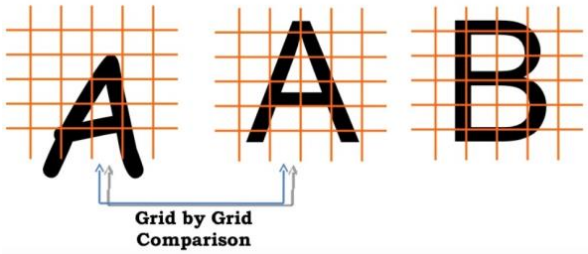
## 1.2 Giới thiệu bài toán

Bài toán xây dựng hệ thống tự động phát hiện cây bút bi trong ảnh đầu vào với các đặc điểm sau:

- Ảnh đầu vào là ảnh nền trơn chứa 1 đối tượng là cây bút bi (ảnh test), kích thước của ảnh huấn luyện bằng với kích thước của vật thể cây bút trong ảnh test.
- Ảnh huấn luyện (dữ liệu huấn luyện) gồm 1 ảnh chứa cây bút bi (dương tính) và 1 ảnh không chứa đối tượng (âm tính).

Yêu cầu: Phát hiện đối tượng là cây bút bi trong ảnh đầu vào dựa trên dữ liệu huấn luyện.

## 1.3 Giới thiệu phương pháp thực hiện




- Phương pháp so sánh từng ô lưới (grid-by-grid comparison) là một kỹ thuật đơn giản để so sánh hai hình ảnh bằng cách chuyển ảnh về dạng nhị phân và chia chúng thành các ô lưới nhỏ và sau đó so sánh từng ô lưới này với nhau.

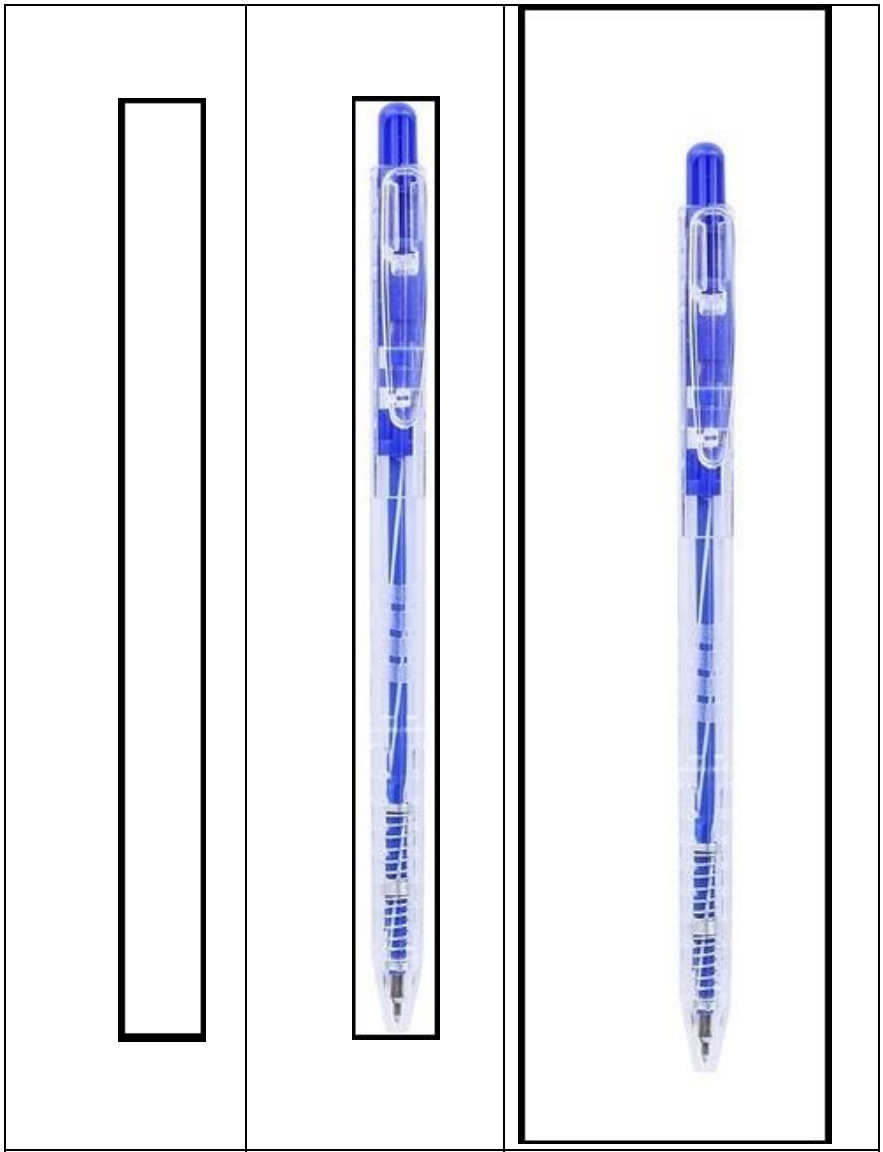
- Khi tất cả các ô lưới đã được xác định, quá trình so sánh bắt đầu. Mỗi ô lưới trong hình ảnh test được so sánh với mỗi ô lưới tương ứng trong hình ảnh train. Sau khi so sánh xong, sẽ đưa ra số lượng cặp các ô lưới (điểm ảnh) khác nhau, gọi là số mismatch.
- Số mismatch càng nhỏ thì số điểm ảnh không khớp giữa 2 hình ảnh càng ít, từ đó kết luận được mẫu ảnh test giống với ảnh train.
- Phương pháp so sánh từng ô lưới (grid-by-grid comparison) có thể được áp dụng trong nhiều bài toán khác nhau, từ nhận dạng vật thể đến so sánh hình ảnh, … Tuy nhiên, điều quan trọng là kích thước của ảnh train cần phải được chọn một cách cẩn thận để đảm bảo tính linh hoạt và hiệu suất của phương pháp.

# PHẦN 2. THỰC HIỆN

## 2.1 Các bước thực hiện bài toán

### Bước 1. Thu thập dữ liệu

- Ảnh đầu vào là ảnh nền trơn chứa 1 đối tượng là cây bút bi (ảnh test), kích thước của ảnh huấn luyện bằng với kích thước của vật thể cây bút trong ảnh test.
- Ảnh huấn luyện (dữ liệu huấn luyện) gồm 1 ảnh chứa cây bút bi (dương tính) và 1 ảnh không có cây bút bi (âm tính).



| *Train không bút* | *Train có bút* | *Test* |
|---|---|---|

***Hình 1.*** *Ảnh gốc*

**Bước 2. Tiền xử lý dữ liệu**

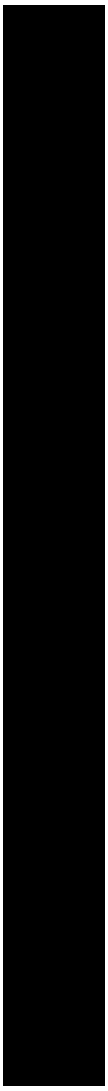
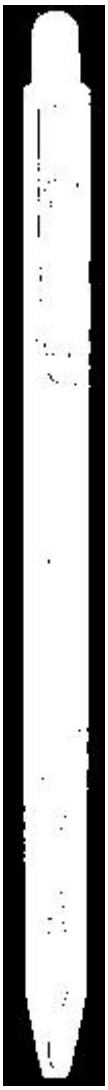
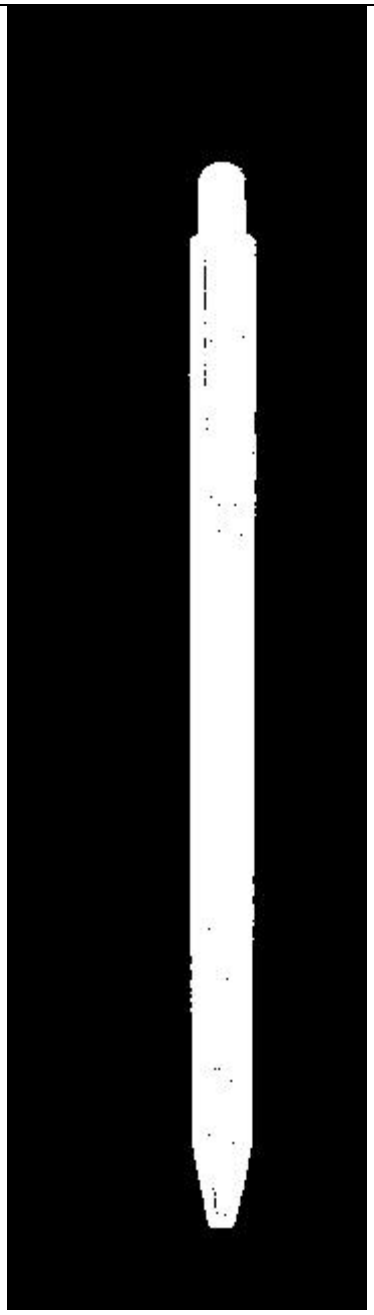
Các dữ liệu ảnh trong tập test và train ban đầu đều ở dạng ảnh RGB

- Điều chỉnh kích thước 2 ảnh train bằng với kích thước của vật thể cây bút bi về kích thước (50x540) pixels.
- Từ ảnh RGB chuyển tất cả dữ liệu ảnh về dạng ảnh xám.
- Tiếp tục chuyển ảnh xám về dạng ảnh nhị phân, với vật thể cây bút là màu trắng và nền màu đen.

```python
import os
from PIL import Image

def process_images(input_folder, output_folder, threshold=225):
    files = os.listdir(input_folder)

    image_files = [f for f in files if f.lower()]

    for image_file in image_files:
        input_path = os.path.join(input_folder, image_file)

        with Image.open(input_path) as img:
            # Chuyển sang ảnh xám
            gray_img = img.convert('L')

            # Tạo ảnh nhị phân
            binary_img = gray_img.point(lambda x: 255 if x < threshold else 0, 'L')

            # Tạo tên file đầu ra cho ảnh nhị phân
            binary_filename = f"binary_{image_file}"
            binary_path = os.path.join(output_folder, binary_filename)
            binary_img.save(binary_path)
        print(f"Đã xử lý: {image_file}")

input_folder = "data/raw"
output_folder = "data/processed"
process_images(input_folder, output_folder)
```

***Hình 2.*** *Đoạn code trong bước tiền xử lý dữ liệu*

**Kết quả:** Các ảnh dạng RGB đã được chuyển sang ảnh dạng nhị phân

| | | |
|---|---|---|
|  |  |  |
| Ảnh train không bút | Ảnh train có bút | Ảnh test |

***Hình 3.*** *Ảnh nhị phân*

**Bước 3. Rút trích đặc trưng**

Rút trích đặc trưng là bước quan trọng để thu thập thông tin cần thiết từ ảnh, giúp mô hình có thể phát hiện là cây bút bi. Sau khi đã xử lý các bước như chuyển đổi ảnh thành ảnh xám và nhị phân, ta cần rút trích các đặc trưng nổi bật để dễ dàng phân biệt giữa bút bi và nền.

**Bước 4. Áp dụng cửa sổ trượt**

Sau khi rút trích đặc trưng từ ảnh huấn luyện (50x540 pixels), chúng ta có thể sử dụng khung trượt (sliding window) để phát hiện cây bút trong ảnh test. Khung trượt sẽ di chuyển qua từng vị trí trong ảnh lớn và so sánh đặc trưng của mỗi phần với đặc trưng đã huấn luyện.

```
17
18  def sliding_window_and_save_arrays(image_path, window_width, window_height, step_size):
19      # Load ảnh nhị phân
20      binary_image = load_binary_image(image_path)
21      img_height, img_width = binary_image.shape
22
23      # Danh sách lưu các mảng con của khung trượt
24      subarrays = []
25
26      # Trượt khung hình chữ nhật 50x540 trên ảnh test
27      for y in range(0, img_height - window_height + 1, step_size):
28          for x in range(0, img_width - window_width + 1, step_size):
29              # Lấy mảng con tương ứng với khung trượt
30              subarray = save_subarray(binary_image, x, y, window_width, window_height)
31              subarrays.append((subarray, x, y))  # Lưu cả vị trí của khung trượt
32
33      return subarrays
34
35
36  def load_train_image(train_image_path):
37      # Đọc ảnh train.jpg và chuyển đổi thành mảng nhị phân
38      arr_train = load_binary_image(train_image_path)
39      return arr_train
40
41
42  def compare_arrays(arr1, arr2):
43      difference = np.sum(arr1 != arr2)
44      return difference
45
```

***Hình 4.*** *Đoạn code trong bước áp dụng cửa sổ trượt*

**Bước 5. Phát hiện đối tượng**

Cho khung này trượt trên toàn bộ ảnh cần phát hiện mẫu (ảnh đã chuyển thành nhị phân) rồi đem từng khung đó so sánh với ảnh dương tính. Số điểm mismatch ít nhất sẽ xác định được vị trí của vật. Sau khi xác định vị trí của vật in ra kết quả kèm theo khung bao

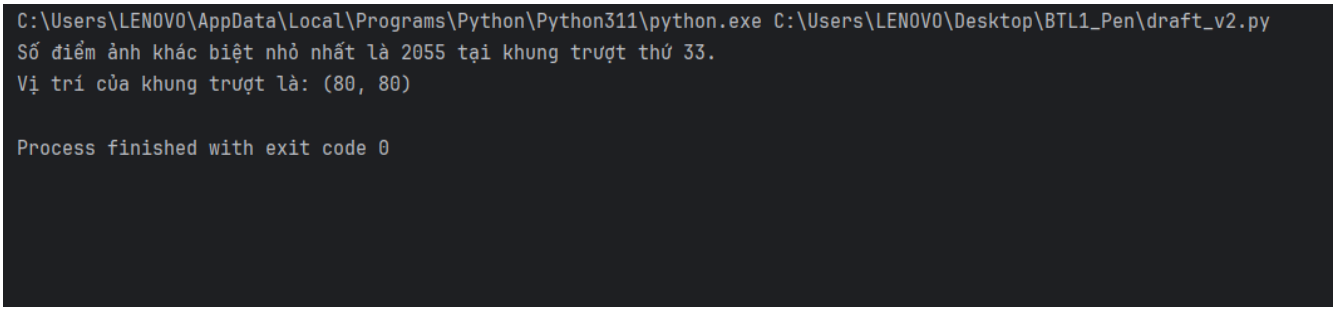
quanh vật đã xác định.

```
72
73  # Tìm khung trượt có sự khác biệt nhỏ nhất
74  min_difference = float('inf')  # Khởi tạo giá trị khác biệt nhỏ nhất là vô cùng lớn
75  min_index = -1  # Chỉ số của mảng với khác biệt nhỏ nhất
76  min_x, min_y = 0, 0  # Vị trí của khung trượt có sự khác biệt nhỏ nhất
77
78  for i, (subarray, x, y) in enumerate(subarrays, start=1):
79      difference = compare_arrays(subarray, arr_train)
80      if difference is not None:
81          if difference < min_difference:
82              min_difference = difference
83              min_index = i
84              min_x, min_y = x, y
85
86  # In số điểm ảnh khác biệt nhỏ nhất và vẽ hình chữ nhật quanh khung trượt
87  if min_index != -1:
88      print(f"Số điểm ảnh khác biệt nhỏ nhất là {min_difference} tại khung trượt thứ {min_index}.")
89      print(f"Vị trí của khung trượt là: ({min_x}, {min_y})")
90
91      # Vẽ hình chữ nhật và hiển thị ảnh
92      draw_rectangle_and_show('data/raw/test.jpg', min_x, min_y, window_width, window_height)
93
```

***Hình 5.*** *Đoạn code trong bước phát hiện đối tượng bút bi*

## 2.2 Kết quả

***Hình 6.*** *Kết quả phát hiện đối tượng*




***Hình 7.*** *Kết quả chạy chương trình*

## 2.3 Đánh giá

- Với kích thước bước cửa sổ trượt là 20 điểm ảnh, thì có tổng cộng 42 khung ảnh trượt trên ảnh test.

- Tìm được số điểm ảnh khác biệt nhỏ nhất là 2055 tại khung ảnh thứ 33

- Hệ thống đã phát hiện được vật thể cây bút bi trong ảnh và khoanh vùng cây bút bằng khung hình chữ nhật màu xanh 50x540 pixels.

# PHẦN 3 KẾT LUẬN

## 3.1 Nhận xét

3.1.1 Nhận xét về phương pháp so sánh từng ô lưới (grid-by-grid comparison):

| Ưu điểm | Nhược điểm |
|---|---|
| - **Phù hợp cho bài toán có kích thước nhỏ**: Với bài toán nhận diện một đối tượng như cây bút bi trong một không gian ảnh tương đối nhỏ (ví dụ 50x540 pixels), việc sử dụng khung trượt và so sánh từng ô lưới là một giải pháp khả thi và có thể đạt hiệu quả tốt nếu đối tượng dễ phân biệt với nền.<br>- **Phân tích chi tiết theo từng vùng**: Chia ảnh thành các ô lưới giúp phương pháp có thể phân tích chi tiết từng vùng của ảnh, cho phép phát hiện đối tượng ngay cả khi nó chỉ xuất hiện trong một phần của ảnh.<br>- **Loại bỏ nhiễu một cách hiệu quả**: Vì phương pháp tập trung vào từng ô nhỏ trong ảnh, nó có thể giúp loại bỏ các chi tiết không quan trọng hoặc nhiễu từ nền trong quá trình phát hiện đối tượng. | - **Khó phát hiện đối tượng khi có sự biến dạng**: Phương pháp so sánh từng ô lưới có thể hoạt động tốt khi cây bút bi có hình dạng ổn định. Tuy nhiên, nếu đối tượng bị che khuất, bị biến dạng, hoặc xuất hiện dưới các góc nhìn khác nhau, phương pháp này có thể gặp khó khăn trong việc phát hiện.<br>- **Không linh hoạt trước các thay đổi về tỷ lệ và góc**: Nếu cây bút bi xuất hiện ở các tỷ lệ hoặc góc độ khác với ảnh huấn luyện, phương pháp này khó có thể điều chỉnh một cách hiệu quả.<br>- **Kích thước ô lưới**: Nếu ô lưới quá lớn, phương pháp có thể bỏ qua các chi tiết quan trọng của cây bút. Nếu ô quá nhỏ, nó có thể yêu cầu quá nhiều tính toán, làm giảm hiệu suất. |

***Bảng 1.*** *Ưu điểm và nhược điểm của phương pháp*

3.1.2 Nhận xét về hệ thống tự động phát hiện đối tượng trong ảnh

- Sử dụng phương pháp so sánh từng ô lưới đơn giản nhưng hiệu quả.
- Có thể điều chỉnh ngưỡng tương đồng để điều chỉnh độ nhạy của việc phát hiện đối tượng.
- Chuyển đổi ảnh nhị phân giúp đơn giản hóa xử lý.
- So sánh trực tiếp với mẫu bằng đếm pixels khác biệt.
- Hiển thị kết quả phát hiện đối tượng trực quan bằng đóng khung hình chữ nhật.
- Hiệu quả với ảnh có nền trơn, đối tượng rõ ràng.

## 3.2 Ưu điểm và nhược điểm

3.2.1 Ưu điểm

- Nhận dạng và phân biệt được các hình ảnh đơn, rõ ràng.
- Phương pháp đơn giản, trực quan dễ thực hiện và dễ hiểu.
- Với việc sử dụng kỹ thuật so sánh từng ô lưới, phương pháp này có thể phát hiện các đối tượng giống với mẫu một cách hiệu quả.

3.2.2 Nhược điểm

- Khó phân biệt các vật thể có hình dáng tương tự nhau.
- Phụ thuộc nhiều vào việc xử lý hình ảnh đầu vào.
- Đối với các vật thể có hình dáng không cân xứng thì góc chụp ảnh hưởng nhiều tới kết quả.
- Hình nền của vật thể ảnh hưởng nhiều tới kết quả, dẫn đến khó phát hiện được vật thể trong những bức ảnh phức tạp.

**3.3 Hướng phát triển**

- So sánh nhị phân đơn giản dễ bị ảnh hưởng bởi nhiễu.
- Có thể tối ưu bằng cách sử dụng OpenCV matchTemplate để cải thiện tốc độ.
- Tối ưu hóa việc chia lưới và khung trượt, thay vì sử dụng một kích thước cố định cho khung trượt hoặc ô lưới, có thể sử dụng kích thước khung trượt thay đổi để phù hợp với nhiều đối tượng có kích thước khác nhau. Điều này giúp tăng khả năng phát hiện các đối tượng với kích thước thay đổi hoặc nằm ở các vị trí không cố định.
- Sử dụng kết hợp nhiều phương pháp rút trích đặc trưng: Thay vì chỉ sử dụng một kỹ thuật như HOG, LBP hoặc SIFT, có thể kết hợp nhiều phương pháp khác nhau để cải thiện độ chính xác.

## Tài liệu tham khảo

[1]. Object Detection [Sliding Window Approach]
[2]. What is Sliding Window in Object Detection: Methods & Tools
[3]. Sliding Windows for Object Detection with Python and OpenCV

**Link code: Github**