Mục lục

[I. Kỹ thuật xử lý video 2](#_Toc8896363)

[1. Đánh chỉ số và truy vấn video 2](#_Toc8896364)

[2. Phân đoạn video(shot detection/segmentation) 3](#_Toc8896365)

[3. Đánh chỉ số và truy vấn video dựa trên phân đoạn. 4](#_Toc8896366)

[II. Cây phân đoạn khung hình (frame segment tree) 5](#_Toc8896367)

[1. Mô tả đầy đủ về cơ sở dữ liệu video 5](#_Toc8896368)

[2. Cấu trúc dữ liệu 6](#_Toc8896369)

[2.1 Cây phân đoạn khung hình 7](#_Toc8896370)

[2.2 Các mảng khác 8](#_Toc8896371)

[3. Truy vấn đối tượng 9](#_Toc8896372)

[4. Truy vấn sự kiện 10](#_Toc8896373)

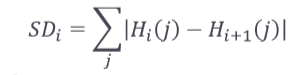
[III. Cài đặt chương trình 10](#_Toc8896374)

1. **Kỹ thuật xử lý video**
2. **Đánh chỉ số và truy vấn video**

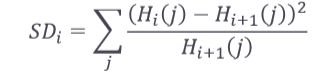
* Dựa trên các thuộc tính chung (tiêu đề, tên tác giả, ngày sản xuất, thể loại,…): giống như các hệ CSDL thông thường (mô hình dữ liệu quan hệ).
* Dựa trên văn bản (phụ đề, thông tin miêu tả,…) kèm theo các nhãn thời gian. Truy vấn bằng hệ thống IR.
* Dựa trên âm thanh đi kèm với video, có thêm thông tin về nhãn thời gian. Truy vấn giống với tín hiệu âm thanh.
* Dựa trên nội dung:
  + Coi video là một chuỗi các ảnh tĩnh. Quá trình truy vấn dựa trên truy vấn ảnh. Nhược điểm: mất liên kết thời gian giữa các khung hình, có thể phải xử lý quá nhiều ảnh.
  + Chia tệp video thành các đoạn nhỏ chứa các khung hình giống nhau. Truy vấn dựa trên các ảnh đại diện cho mỗi đoạn video.
* Phương pháp kết hợp.
* Video có thể chia nhỏ thành các mảnh (segments, shots) bao gồm chuỗi nhiều ảnh tĩnh liên tiếp có liên quan với nhau:
  + Mô tả về cùng một cảnh (scene).
  + Chứa thông tin về một thao tác của máy quay
  + Chứa thông tin về một hành động, sự kiện cụ thể của một thực thể nào đó
  + Các khung hình được lựa chọn như một công cụ duy nhất để đánh chỉ số và tra cứu.
* Các bước chính:
  + Phân đoạn video (segmentation, shot detection, partition)
  + Đánh chỉ số từng đoạn video: xác định khung hình đại diện của từng đoạn, áp dụng các phương pháp đánh chỉ số và truy vấn ảnh cho các khung hình này.
  + Truy vấn dựa trên so sánh tương tự giữa câu truy vấn (hình ảnh hoặc thông tin về khung hình nào đó) với các khung hình đại diện của các đoạn video.

1. **Phân đoạn video(shot detection/segmentation)**

* Các khung hình (frame) trong cùng một đoạn (lần bấm máy) thường giống nhau về một mặt nào đó
* Nếu có một sự khác nhau giữa hai khung hình liên tiếp (quá một ngưỡng cho trước) thì hai khung hình đó thuộc hai đoạn (lần bấm máy) khác nhau. VD:
  + Chuyển cảnh giữa các máy quay (chuyển bất ngờ).
  + Các thao tác trên máy quay: vào hình, ra hình… (chuyển cảnh từ từ).
* Các kỹ thuật cơ bản:
  + Tìm tổng tất cả các điểm ảnh khác nhau (tương ứng vị trí) giữa hai khung hình liên tiếp. So sánh với ngưỡng.
  + Tìm sự khác nhau giữa các biểu đồ tần suất mầu của hai khung hình. So sánh với ngưỡng.



* + Dựa trên kiểm tra 𝜒2 (𝜒2 test)



Vấn đề đặt ra là chọn các giá trị ngưỡng phù hợp.

* Phân đoạn với khung hình biến đổi từ từ:
  + Sử dụng hai ngưỡng, một ngưỡng để bắt cảnh thay đổi đột ngột, một ngưỡng (thấp hơn) để bắt cảnh thay đổi từ từ.
    - Ngưỡng cao T1 được dùng để xác định cảnh thay đổi đột ngột.
    - Ngưỡng thấp T2 dùng để đánh dấu khả năng cảnh thay đổi từ từ. Nếu tổng sự khác nhau giữa các khung hình liên tiếp vượt ngưỡng T1 thì xác định sự đổi cảnh.
* Giảm thiểu sai sót khi phân đoạn:
  + Phân biệt sự chuyển cảnh với các thao tác phóng to, thu nhỏ trên máy quay, hoặc thao thác lia hình bằng máy quay: sử dụng thuộc tính “dòng quang” (optical flow).
  + Phân biệt chuyển cảnh với sự thay đổi cường độ sáng của phim (do các hiệu ứng ánh sáng, hoặc do thay đổi nguồn sáng trong phim): chuẩn hóa các giá trị màu.
    - Chuẩn hóa các giá trị mầu thành phần:



* + - Chuyển đổi sang độ mầu:



* + - Biểu đồ tần suất mầu tổng hợp của r và g được xây dựng.

1. **Đánh chỉ số và truy vấn video dựa trên phân đoạn.**

* Mỗi đoạn video được đại diện bởi một hoặc một vài khung hình.
* Các khung hình được coi là các ảnh tĩnh, việc đánh chỉ số và truy vấn giống như đối với ảnh.
* Trong quá trình truy vấn, so sánh tương tự được thực hiện giữa nội dung truy vấn và các hình đại diện.
* Có nhiều cách để chọn khung hình đại diện cho từng đoạn video.

1. Các cách chọn số khung hình đại diện cho từng phân đoạn video

* Chọn một khung hình duy nhất đại diện cho từng phân đoạn video. Nhược điểm: mất thông tin về độ dài thời gian cũng như nội dung thay đổi trong đoạn video đó.
* Chọn nhiều khung hình đại diện cho đoạn video, số khung hình dựa vào độ dài đoại video đó. Thông thường cứ mỗi giây video chọn một khung hình.
* Mỗi phân đoạn video lại được chia nhỏ thành các mảnh chứa các nội dung/cảnh khác nhau dựa trên vector chuyển động, dòng quang, hoặc sự khác biệt giữa các khung hình liên tiếp nhau. Một khung hình đại diện được chọn cho mỗi mảnh video.

1. Các cách chọn khung hình đại diện cho từng đoạn/mảnh video

* Chọn khung hình đầu tiên của mỗi đoạn/mảnh video.
* Dựa trên khung hình trung bình của mỗi đoạn/mảnh video, trong đó mỗi điểm ảnh chứa giá trị trung bình giá trị mầu của tất cả các điểm cùng vị trí trong các khung hình thuộc đoạn đó. Khung hình nào giống nhất với khung hình trung bình sẽ được chọn làm đại diện cho đoạn/mảnh video đó.
* Dựa trên biểu đồ tần suất mầu trung bình của tất cả các khung hình trong đoạn đó. Khung hình nào có biểu đồ tần suất mầu gần với biểu đồ trung bình thì sẽ được chọn.

1. Đánh chỉ số và truy vấn dựa trên thông tin về chuyển động: thông tin chuyển động được lấy từ dòng quang, hoặc các vector chuyển động.

* Nội dung của chuyển động. VD: đoạn video hội thoại có ít nội dung chuyển động hơn đoạn phim hành động.
* Tính đồng nhất của chuyển động: nói lên tính liên tục của các chuyển động.
* Chuyển động lia hình (theo chiều ngang).
* Chuyển động nghiêng (theo chiều dọc).
* Các khung hình được đánh chỉ số và truy vấn dựa trên cả thông tin về nội dung lẫn thông tin về chuyển động.

1. Đánh chỉ số và truy vấn dựa trên các thực thể.

* Các thực thể trong video có thể được xác định dựa trên các điểm ảnh di chuyển giữa các khung hình liên tiếp.
* Các thực thể có thể được dùng để phân đoạn video và truy vấn video.

1. **Cây phân đoạn khung hình (frame segment tree)**
2. **Mô tả đầy đủ về cơ sở dữ liệu video**

Đầu tiên, chúng ta giả định rằng OBJ là một tập hợp các đối tượng video và EVT là một tập hợp các sự kiện được tìm thấy trong cơ sở dữ liệu video. Do đó, ENT = OBJ∪EVT (ENT là tập hợp của tất cả các thực thể). Bản đồ liên kết λ ánh xạ các phần tử của ENT vào các tập hợp các chuỗi khung. Đối với mỗi sự kiện, ∈ EVT, ánh xạ nhiều-một ℵ ánh xạ mỗi sự kiện thành một loại hoạt động.

Ví dụ: trong ví dụ phim của chúng ta, (e1) = giết người. Tiếp theo, chúng ta giả định rằng, với một loại hoạt động A, một ROLE (A) được thiết lập được liên kết với nó. Các thành viên của bộ ROLE (A) là các chuỗi biểu thị tên của các vai trò liên quan đến hoạt động A. Ví dụ: ROLE (giết người) = {nạn nhân, kẻ giết người, kẻ giết người} và ROLE ({đưa ra một bữa tiệc} = { máy chủ, khách}. Với bất kỳ sự kiện E nào thuộc loại hoạt động A, chúng ta giả sử rằng có một PLAYERS ánh xạ ánh xạ ROLE (A) tới tập OBJ ∪ EVT ∪ STR; STR là tập hợp tất cả các chuỗi có thể. bản đồ PLAYERS chỉ định nhóm tham gia vào sự kiện này.

Định nghĩa 1. Cơ sở dữ liệu video là 9-tuple  
(RVD, OBJ, EVT,λ, ACT, R,,℘,ROLE,PLAYERS)  
trong đó - RVD là một bộ số nguyên {1, ..., n} cho sone n.  
- OBJ là một tập hợp các đối tượng. - EVT là một tập hợp các sự kiện.  
- là bản đồ liên kết gán một tập hợp các chuỗi khung vững chắc cho mỗi thực thể ent (OBJ∪EVT).  
- ACT là một tập hợp các loại hoạt động.  
- R là tập hợp các vai trò.  
- ℘ là bản đồ chỉ định loại hoạt động cho từng sự kiện.  
- ROLE là bản đồ lấy chuỗi khung [i, j) ∈ EVT∪So∈OBJ λ (o), một hoạt động A trong ℘([i, j)) và vai trò trong R (A) làm đầu vào và chỉ định một thành viên của OBJ làm đầu ra.  
- PLAYERS là một bản đồ lấy một sự kiện và loại hoạt động của nó làm đầu vào và trả về ánh xạ từ các vai trò của hoạt động đến các thực thể trong cơ sở dữ liệu và thành chuỗi làm đầu ra.

1. **Cấu trúc dữ liệu**

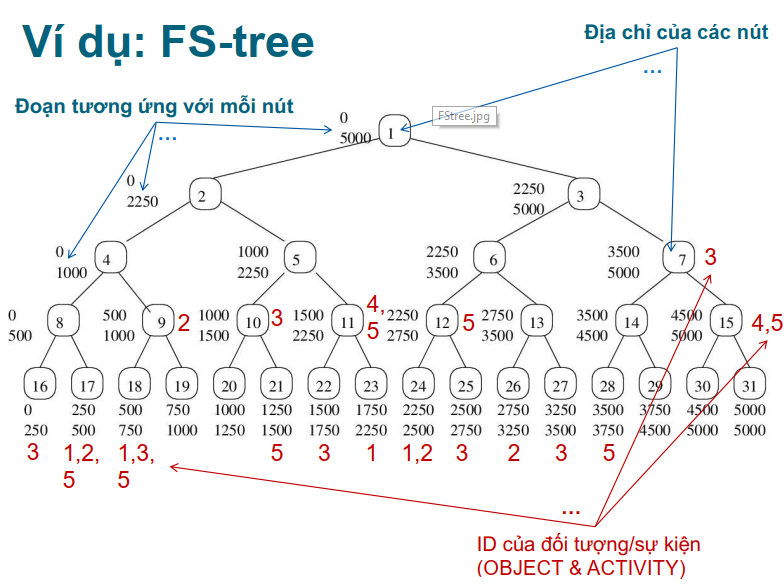
Kỹ thuật FS-tree (Frame-segment tree)

Khái niệm:

– Chuỗi khung hình: [i, j) = {k | i≤ k <j}  
– Thứ tự chuỗi khung hình: [i1, j1) ⊑ [i2, j2) ⇔ i1 < j1 ≤ i2 < j2  
fs1 = [8, 10), fs2 = [10, 15), fs3 = [11, 13)  
 fs1 ⊑ fs2, fs1 ⊑ fs3, fs2 ⋢ fs3  
– Tập có thứ tự chuỗi khung hình:  
+ X= {[i1, j1), [i2, j2), …, [ir, jr)}

+ [i1, j1) ⊑ [i2, j2) ⊑ … ⊑ [ir, jr)  
+ VD: X= {[4, 9), [10, 11), …, [27, 41)}

– Tập chặt (*solid set*) chuỗi khung hình  
+ Là tập có thứ tự chuỗi khung hình  
+ Không tồn tại cặp chuỗi: [i1,i2) , [i2, i3)  
+ X = { [3, 5), [5, 7), [9, 11) }: not solid  
+ X = { [3, 5), [6, 7), [9, 11) }: solid  
– Biểu đồ kết hợp đoạn (*segment association map*) σv  
+ V = *(OBJ, ACT,* λ*)*+ σv(x) = A, x ∈ OBJ ∪ ACT, A: solid set  
 [s, e) ∈ A ➔ ∀f: s ≤ f < e, x ∈ λ*(f)* ∀f , ∀x ∈ OBJ ∪ ACT, nếu x ∈ λ*(f)* ➔ ∃ *[s, e)* ∈ A *: f* ∈[s, e)



Chúng ta định nghĩa các cấu trúc dữ liệu cần thiết để lưu trữ dữ liệu video mà chúng ta quan tâm để xử lý. Có bốn loại cấu trúc cần được định nghĩa: cây phân đoạn khung cho phép chúng ta chỉ định các sự kiện nào (và các trình phát, đội, v.v. có liên quan của chúng) và các đối tượng video xảy ra trong các phân đoạn của video; OBJECTARRAY cho phép truy cập các phân đoạn video bằng cách sử dụng các đối tượng làm khóa,

ACTIVITYARRAY cho phép truy cập các phân đoạn video bằng cách sử dụng các loại hoạt động làm khóa

EVENTARRAY tạo điều kiện truy cập bằng các sự kiện làm khóa. Trong số này, thành phần chính là một loại cây phân đoạn đặc biệt.

* 1. **Cây phân đoạn khung hình**

Chúng ta định nghĩa một loại cây phân đoạn đặc biệt gọi là cây phân đoạn khung hinh được sử dụng cho dữ liệu đa phương tiện. Cấu trúc của các nút trong cây phân đoạn khung hình có thể được định nghĩa như sau. Lưu ý rằng trong cấu trúc dữ liệu này, trình phát bị giới hạn ở các đối tượng video. Nó là đơn giản để tăng cường các cấu trúc dữ liệu để cho phép người chơi thuộc bất kỳ loại nào.

type treenode = record of

start : integer; /\* vị trí frame bắt đầu \*/

finish : integer; /\* vị trí frame kết thúc \*/

objlist : ^objnode ; /\* danh sách các đối tượng liên kết với node \*/

evtlist : ^evtnode; /\*danh sách các sự kiện liên kết với node \*/

lchild : ^treenode ; /\* con bên trái\*/

rchild : ^treenode; /\* con bên phải \*/

end;

type objnode = record of

objid : integer ; /\* số của đối tượng \*/

next : ^objnode; /\* con trỏ đến node tiếp theo trong cây \*/

end;

type evtnode = record of

evtid : integer; /\* id của sự kiện \*/

next : ^evtnode; /\*con trỏ đến node sự kiện tiếp theo \*/

end;

* 1. **Các mảng khác**

OBARARRAY, ACTIVITYARRAY và EVENTARRAY có thể được định nghĩa là các mảng với các phần tử có cấu trúc bản ghi tương ứng là objectrec, activityrec và eventrec. Chúng ta giả sử rằng Val1, Val2, Val3, Val4 là các hằng số.

type objectrec = record name : string; /\* tên của đối tượng\*/ frames : ^frameseqlist;

/\* con trỏ đến danh sách các node \*/

end;

type activityrec = record

name : string;

events : ^evtnode;

end;

type eventrec = record

acttype: integer;

teamlist: ^playernode;

frames : ^frameseqlist;

end;

type playernode = record of

player: integer; /\* số của đối tượng \*/

role : integer; /\* vai trò của đối tượng \*/

next : ^playernode; /\* con trỏ đến player tiếp theo \*/

end;

type frameseqlist = record

segments : ^treenode; /\* con trỏ đến 1 node của cây \*/

next : ^frameseqlist; /\* con trỏ đến node tiếp theo \*/

end;

eventarray : array[1..Val1] of eventrec;

activityarray : array[1..Val2] of activityrec;

objectarray : array[1..Val3] of objectrec;

rolearray: array[1..Val4] of string;

Với cấu trúc dữ liệu cây phân đoạn khung hình này, chúng có thể tiếp tục với:

1. Danh sách các truy vấn cơ bản có thể được đặt ra cho hệ thống
2. Phát triển các thuật toán để thực hiện các truy vấn này.
3. Phân tích độ phức tạp của các thuật toán cho mỗi truy vấn.
4. Kết xuất chính thức các thuật toán để cập nhật cấu trúc dữ liệu một cách hiệu quả khi các thực thể mới trong các video hiện có trở nên quan trọng.
5. **Truy vấn đối tượng**

Đây là một truy vấn có dạng: tất cả các đối tượng xảy ra trong một tập hợp các chuỗi khung nhất định.  
Phương pháp: Truy vấn có thể được giải quyết bằng cách tìm kiếm cây phân đoạn khung bắt đầu từ gốc cho tập hợp các chuỗi khung đã cho. Nếu nút được truy cập giao với bất kỳ chuỗi khung nào trong tập hợp, tất cả các đối tượng được lưu trữ trong nút này sẽ được thêm vào tập hợp các đối tượng đầu ra, và sau đó cả hai bên trái và bên phải của nút được truy cập. Có thể phân chia tập hợp các chuỗi khung con để chỉ những chuỗi đó có thể giao nhau với khung con tương ứng mới được đưa vào lời gọi. Mỗi khi một tập hợp các đối tượng mới được thêm vào danh sách đầu ra, các bản sao cần phải được loại bỏ. Đây là một hoạt động tốn kém. Sử dụng một túi làm cụm các đối tượng là một bước để giảm độ phức tạp của thao tác này.

Phức tạp: Vì thông tin này có thể được tính thông qua cây phân đoạn, chi phí liên quan tỷ lệ thuận với số lượng các nút trong cây phân đoạn có các khoảng của nút cha chồng lên một chuỗi khung đã cho trong nhật ký thời gian truy vấn (số chuỗi khung đầu vào) cộng với số của các đối tượng trong mỗi nút đó. Nó không thể được tính toán nhanh hơn bởi vì mỗi nút như vậy phải được kiểm tra. (hệ số nhật ký là chi phí kiểm tra xem một nút đã cho có một khoảng thời gian chồng chéo với một chuỗi khung trong truy vấn không.

1. **Truy vấn sự kiện**

Đây là một truy vấn có dạng: tất cả các hoạt động xảy ra trong một tập hợp các chuỗi khung nhất định.  
Phương pháp / Độ phức tạp. Điều này rất giống với truy vấn xảy ra đối tượng. Cây được tìm kiếm bởi cùng một thuật toán. Lần này các sự kiện được thu thập và khi kết thúc tìm kiếm, các sự kiện được thu thập được nhóm thành các loại hoạt động tương ứng của chúng. Danh sách này được trả lại.

1. **Cài đặt chương trình**

Do các khung hình liên tiếp có độ tương đồng khá giống nhau, nên ta cứ 10 frame thì lấy 1 frame và lưu lại

**public** **class** ExtractFrameVideo {

**public** **static** **void** main(String args[]) {

System.*loadLibrary*(Core.***NATIVE\_LIBRARY\_NAME***);

VideoCapture cap = **new** VideoCapture();

//Đường dẫn vidao

String input = "G:\\eclipse-workspace\\testOpenCV\\videoplayback.mp4";

//Đường dẫn thư mục lưu các ảnh được tách

String output = "G:\\eclipse-workspace\\testOpenCV\\extractedImage";

//Đọc video

cap.open(input);

**int** video\_length = (**int**) cap.get(Videoio.***CAP\_PROP\_FRAME\_COUNT***);

**int** frames\_per\_second = (**int**) cap.get(Videoio.***CAP\_PROP\_FPS***);

**int** frame\_number = 0;

Mat frame = **new** Mat();

**if** (cap.isOpened()) {

**if** (cap.isOpened()) {

System.***out***.println("Video đã đọc");

System.***out***.println("Tổng số frame: " + video\_length);

System.***out***.println("Tốc độ đọc frame/s: "+frames\_per\_second );

System.***out***.println("Đang tách ảnh từ video");

//Đọc các frame

**while** (cap.read(frame))

{

//cứ 10 frame thì tách lấy 1 ảnh

**if** (frame\_number % 10 == 0)

Imgcodecs.*imwrite*(output + "/" + frame\_number + ".jpg", frame);

frame\_number++;

}

System.***out***.println("Xuất thành công");

cap.release();

}

}

**else** {

System.***out***.println("Fail");

}

}

}

1. **Hàm so sánh phần trăm độ tương đồng giữa 2 bức ảnh**

**class** SimilerImage

{

//Kiểm tra độ tương đồng giữa 2 bức ảnh

**public** **static** **double** differencePercentageTwoImage(BufferedImage imgA,BufferedImage imgB)

{

**int** width1 = imgA.getWidth();

**int** width2 = imgB.getWidth();

**int** height1 = imgA.getHeight();

**int** height2 = imgB.getHeight();

**if** ((width1 != width2) || (height1 != height2)) {

System.***out***.println("Error: Images dimensions"+

" mismatch");

**return** 100;

}

**else**

{

**long** difference = 0;

**for** (**int** y = 0; y < height1; y++)

{

**for** (**int** x = 0; x < width1; x++)

{

**int** rgbA = imgA.getRGB(x, y);

**int** rgbB = imgB.getRGB(x, y);

**int** redA = (rgbA >> 16) & 0xff;

**int** greenA = (rgbA >> 8) & 0xff;

**int** blueA = (rgbA) & 0xff;

**int** redB = (rgbB >> 16) & 0xff;

**int** greenB = (rgbB >> 8) & 0xff;

**int** blueB = (rgbB) & 0xff;

difference += Math.*abs*(redA - redB);

difference += Math.*abs*(greenA - greenB);

difference += Math.*abs*(blueA - blueB);

}

}

//tổng các pixels của đỏ xanh lá và danh dương điểm ảnh

**double** total\_pixels = width1 \* height1 \* 3;

//Trung bình giá trị khác nhau của pixel

**double** avg\_different\_pixels = difference /

total\_pixels;

// There are 255 values of pixels in total

**double** percentage = (avg\_different\_pixels /

255) \* 100;

**return** percentage;

}

}

}

1. **Tách các frame thành các segment. Mỗi segment có 1 frame đại diện**

Frame đại diện là frame đầu tiên của đoạn segment. Tách frame đó dựa trên độ khác nhau giữa 2 frame liên tiếp > 30% dựa theo thuật toán phía trên là phù hợp.

Các ảnh đại diện sau khi tách được lưu lại trong thư mục nodeTrees

**public** **static** ArrayList<Integer> tachIframeDaiDien() {

//đọc tất cả các frame đã tách từ video dùng để nhận diện độ tương đồng

File[] files = ***dir***.listFiles(***IMAGE\_FILTER***);

//sắp xếp mảng files theo thứ tự frame đã tách

Arrays.*sort*(files, **new** Comparator<File>() {

**public** **int** compare(File o1, File o2) {

**int** n1 = extractNumber(o1.getName());

**int** n2 = extractNumber(o2.getName());

**return** n1 - n2;

}

**private** **int** extractNumber(String name) {

**int** i = 0;

**try** {

**int** s = name.indexOf('\_') + 1;

**int** e = name.lastIndexOf('.');

String number = name.substring(s, e);

i = Integer.*parseInt*(number);

} **catch** (Exception e) {

i = 0;

}

**return** i;

}

});

ArrayList<BufferedImage> listImage = **new** ArrayList<BufferedImage>();

ArrayList<String> imageNames = **new** ArrayList<String>();

ArrayList<Integer> tree = **new** ArrayList<Integer>();

tree.add(0);

//nếu dir là thư mục

**if** (***dir***.isDirectory()) {

**for** (**final** File f : files) {

BufferedImage img = **null**;

**try** {

img = ImageIO.*read*(f);

listImage.add(img);

imageNames.add(f.getName());

} **catch** (**final** IOException e) {

}

}

File outputfile1 = **new** File("nodeTrees\\0.jpg");

**try** {

ImageIO.*write*(listImage.get(0), "jpg", outputfile1);

} **catch** (IOException e1) {

// **TODO** Auto-generated catch block

e1.printStackTrace();

}

**for** (**int** k = 0; k < listImage.size() - 1; k++) {

//nếu 2 ảnh liên tiếp mà độ khác nhau >30% thì lấy ảnh đó làm fame đại diện

**if**(SimilerImage.*differencePercentageTwoImage*(listImage.get(k),listImage.get(k+1))>30) {

//Ghi frame đại diện đó vào thư mục nodeTrees

File outputfile = **new** File("nodeTrees\\"+imageNames.get(k+1));

**try** {

ImageIO.*write*(listImage.get(k+1), "jpg", outputfile);

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

String str=imageNames.get(k+1);

str=str.replaceAll("\\D+","");

tree.add(Integer.*valueOf*(str));

System.***out***.println("frame đại diện "+str+" độ khác nhau với ảnh kế tiếp "+SimilerImage.*differencePercentageTwoImage*(listImage.get(k), listImage.get(k + 1)));

}

}

}

System.***out***.println(tree);

System.***out***.println("Tách frame đại diện thành công");

**return** tree;

}

1. **Dùng python để nhận diện lần lượt mỗi frame đại diện có những đối tượng nào và in ra file**

from imageai.Detection import ObjectDetection

import os

import glob

def laydoituong(thislist):

print("hihi" )

file = open("DetectedObject.txt", "r+")

execution\_path = os.getcwd()

detector = ObjectDetection()

detector.setModelTypeAsRetinaNet()

detector.setModelPath( os.path.join(execution\_path , "resnet50\_coco\_best\_v2.0.1.h5"))

detector.loadModel()

doituong=[]

for anh in thislist:

detections = detector.detectObjectsFromImage(input\_image=os.path.join(execution\_path , anh), output\_image\_path=os.path.join(execution\_path , "imagenew.jpg"))

file.write(anh+" ")

print(anh+" ")

for eachObject in detections:

file.write(eachObject["name"] + " ");

print(eachObject["name"] + " ")

doituong.append(eachObject["name"])

#print(eachObject["name"] + " : " + eachObject["percentage\_probability"] )

file.write("\n")

print("\n")

return doituong

chim=laydoituong(glob.glob("nodeTrees/\*.jpg"));

1. **Class Node của đoạn segment**

**public** **class** Node {

**public** String anhDaiDien;

**public** String objects;

**public** **int** start;

**public** **int** end;

@Override

**public** String toString() {

**return** "Node [anhDaiDien=" + anhDaiDien + ", objects=" + objects + ", start=" + start + ", end=" + end + "]";

}

}

1. **Tìm object có xuất hiện trong đoạn [s1,s2]**

**public** **static** Set<String> timObjectXuatHien(**int** s1,**int** s2,ArrayList<Node> nodes) {

String objects="";

**for** (**int** i=0;i<nodes.size();i++) {

**if**(nodes.get(i).start<s1&&nodes.get(i).end>s1) {

objects+=nodes.get(i).objects;

**for** (**int** j=i+1;j<nodes.size();j++) {

**if**(nodes.get(j).start<=s2) {

objects+=nodes.get(j).objects;

}

**else** {

**break**;

}

}

**break**;

}

}

String[] splited = objects.trim().split("\\s+");

Set<String> set = **new** HashSet();

**for** (String string : splited) {

set.add(string);

}

**return** set;

}

1. **Tìm object xuất hiện liên tiếp trong đoạn [s1,s2]**

**public** **static** List<String> timObjectHienLienTuc(**int** s1,**int** s2,ArrayList<Node> nodes){

ArrayList<String> strings=**new** ArrayList<String>();

**for** (**int** i=0;i<nodes.size();i++) {

**if**(nodes.get(i).start<s1&&nodes.get(i).end>s1) {

strings.add(nodes.get(i).objects);

**for** (**int** j=i+1;j<nodes.size();j++) {

**if**(nodes.get(j).start<=s2) {

strings.add(nodes.get(j).objects);

}

**else** {

**break**;

}

}

**break**;

}

}

String string=strings.get(0);

strings.remove(0);

List<String> parts = Arrays.*asList*(string.trim().split("\\s+"));

String[] splited = string.trim().split("\\s+");

**for** (String string2 : strings) {

String[] temp = string2.trim().split("\\s+");

List<String> mang=**new** ArrayList<String>();

**for** (String string3 : parts) {

//System.out.println("string3 "+string3);

**for**(**int** i=0;i<temp.length;i++) {

//System.out.println(string3+" "+temp[i]);

**if**(string3.equals(temp[i])) {

//System.out.println("hihi");

mang.add(string3);

}

}

}

Set<String> set = **new** HashSet();

**for** (String a : mang) {

set.add(a);

}

List<String> mang2=**new** ArrayList<String>();

**for** (String a : set) {

mang2.add(a);

}

parts=mang2;

}

**return** parts;

}

1. **Hàm main lấy cho dữ liệu vào các node và tìm kiếm**

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

ArrayList<Integer> myList=ExtractFrameOfSegment.*layMang*();

ArrayList<Node> nodes=**new** ArrayList<Node>();

ArrayList<String> ds=**new** ArrayList<String>();

**try** (FileReader reader = **new** FileReader("DetectedObject.txt");

BufferedReader br = **new** BufferedReader(reader)) {

// read line by line

String line;

**while** ((line = br.readLine()) != **null**) {

ds.add(line);

}

} **catch** (IOException e) {

System.***err***.format("IOException: %s%n", e);

}

**for** (**int** i=0;i<myList.size()-1;i++) {

Node node=**new** Node();

node.start=myList.get(i);

node.anhDaiDien=String.*valueOf*(myList.get(i))+".jpg";

String s = ds.get(i);

**int** vt=s.indexOf(" ");

s=s.substring(vt+1);

node.objects=s;

node.end=myList.get(i+1)-1;

nodes.add(node);

}

**for** (Node node : nodes) {

System.***out***.println(node.toString());

}

Scanner keyboard = **new** Scanner(System.***in***);

**int** s1;

**int** s2;

String key="";

**do** {

System.***out***.println("Nhập s1,s2:");

s1=keyboard.nextInt();

s2=keyboard.nextInt();

keyboard.nextLine();

System.***out***.println("Nhập từ khóa:");

key=keyboard.nextLine();

**if**(key.equals("forall"))

System.***out***.println("thực thể xuất hiện liên tục:"+*timObjectHienLienTuc*(s1, s2, nodes));

**if**(key.equals("exists"))

System.***out***.println("thực thể xuất hiện:"+*timObjectXuatHien*(s1, s2, nodes));

}**while**(!key.equals("0"));

}