ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

MÔN: HỆ CƠ SỞ DỮ LIỆU ĐA PHƯƠNG TIỆN

ĐỀ TÀI: PHÁT HIỆN CÁC SHOTS SỬ DỤNG ĐẶC TRƯNG CỤC BỘ SIFT

GVHD: TS. Nguyễn Thị Oanh

NHÓM SV:

1. Nguyễn Lan Anh - 20140131
2. Nguyễn Mạnh Hùng - 20142088
3. Nguyễn Anh Tuấn - 20144898

Hà Nội, tháng 5, năm 2018

Mục lục:

Lời nói đầu

Với sự phát triển của công nghệ hiện nay, hẳn con người đang dần quen với nhiều kiểu dữ liệu đa dạng. Các kiểu dữ liệu không chỉ đơn giản là dạng văn bản, dạng text, hay dạng bảng số liệu,...Mà còn là các kiểu dữ liệu phức tạp hơn như ảnh, âm thanh hay video (một sự kết hợp giữa ảnh và video). và nhu cầu làm việc với những kiểu dữ liệu như này càng tăng lên đặc biệt là với dạng video. Xuất phát từ việc truy vấn đối tượng trong video, bài toán phát hiện các shots trong video, nhằm đáp ứng tốt cho việc đánh chỉ mục cho video, đã ra đời. Vậy nên với đề tài này, nhóm đã quyết định tìm hiểu phát hiện các shots dựa trênđặc trưng cục bộ SIFT.

1. Xây dựng hệ thống phân đoạn video dựa trên đặc trưng cục bộ SIFT

- Input: video

- Output: Các video đã được phân ra từ video Input/ Các frames trong các shots riêng biệt lưu vào một folder riêng(tức mỗi shot tương ứng với một folder, folder này chứa các frames)

- Engine:

Cấu trúc chương trình sẽ gồm các files với chức năng như dưới đây

1. "FilterFrames.py": run file với

python3 FilterFrames.py

Để từ video ban đầu tách ra được các frames của video đó, lưu vào folder frames

2. "ColorExtract.py":

File này chưa function đối sánh màu sắc giữa hai frames liền kề, trả về mảng các cặp frames ứng cử viên mà tại đó có thể xảy ra chuyển cảnh (transition)

3. "ShotsExtract.py"

File này sẽ chưa các function đối sánh các cặp frames trong mảng ứng cử viên bằng cách sử dụng trích chọn đặc trưng SIFT của các frames và sử Match trong OpenCV để đối sánh các đặc trưng này. Đồng thời, nhóm cũng đề xuất một phương pháp để phân biệt chuyển cảnh(transition) là cut transition hay fade transition (trường hợp wipe transition hay dissolve transition sẽ không được đề cập đến trong đề tài này)

2. Nguyên lý xây dựng hệ thống

- Trích chọn các ứng cử viên , tại đó có thể xảy ra chuyển cảnh:

Một video, là sự kết hợp của số lượng không nhỏ các frames, với việc phát hiện các shot boundaries mà phải duyệt qua tất cả các frames để trích chọn đặc trưng cục bộ sẽ gây ra không ít nhưng công việc thừa thãi mà hiệu suất lại không cao. (Bởi theo thông kê, số lượng các shots boundaries thường ít hơn 1% tổng số lượng các frames trong video). Vậy nên nhóm đề xuất cách so sánh sự sai khác về mặt màu sắc giữa các frames trước khi trích chọn đặc trưng cục bộ (sử dụng sai khác về color histogram).

Sử dụng thư viện OpenCV, cụ thể sử dụng các hàm cho việc trích đặc trưng màu sắc và hàm đối sánh màu sắc.

```
histi = cv2.calcHist([hsvi], [0, 1], None, [180, 256], [0, 180, 0, 256])
histi_1 = cv2.calcHist([hsvi_1], [0, 1], None, [180, 256], [0, 180, 0, 256])
a = cv2.compareHist(histi, histi_1, cv2.HISTCMP_BHATTACHARYYA)
```

Trong đó:

histi: kết quả sau khi trích chọn đặc trưng về màu sắc của frame thứ I

histi_1: Kết quả sau khi trích chọn đặc trưng về màu sắc của frame ngay sau frame thứ

a: Là kết quả đối sánh màu sắc (tức sự tương đồng về màu sắc). a = 0, tức 2 frames này có màu sắc hoàn toàn giống nhau. a càng lớn, tức màu sắc giữa 2 frames này càng khác nhau.

Phải chọn a như một ngưỡng để xác định tại đó màu sắc giữa 2 frames là đủ khác để tại đó có thể xảy ra chuyển cảnh. Việc chọn a này, đang hoàn toàn dựa trên xét thử các giá trị của a. Và giá trị a đang nhận trong hệ thống là a = 0.45, tức với 2 frames kề nhau có a > 0.45 sẽ được xác định là tại đó có thể xảy ra chuyển cảnh, và được đưa vào mảng cặp đề cử.

Việc chọn ngưỡng a tại đây rất quan trong cho các phần xác định phía sau.

VD: khi chọn a = 0.3

```
./frames/frame479.jpg
./frames/frame480.jpg
_________________________
./frames/frame480.jpg
./frames/frame481.jpg
_________________________
./frames/frame481.jpg
./frames/frame482.jpg
_________________________
./frames/frame482.jpg
./frames/frame483.jpg
_________________________
./frames/frame483.jpg
./frames/frame484.jpg
_________________________
./frames/frame486.jpg
./frames/frame487.jpg
_________________________
./frames/frame498.jpg
./frames/frame499.jpg
_________________________
./frames/frame510.jpg
./frames/frame511.jpg
_________________________
```

cặp frames được in ra là ứng cử viên cho vị trí chuyển cảnh. Nhưng có thể thấy từ frame thứ 479 đến 484 đã liên tục được xếp vào ứng cử viên trong khi rõ ràng thấy khả năng chuyển cảnh lớn nhất ở vị trí giữa 2 frame (486,487)

Bên trên là 2 frames (479,480)

Trong khi 2 frames (486, 487) như hình dưới đây, rõ ràng có thể xảy ra chuyển cảnh:

sau khi chọn a = 0.45, Sự sai lệch này đã không còn.

- Phát hiện chuyển cảnh là cut transition, hay fade transition:

Sau khi nhận được một mảng danh sách các cặp frames mà tại đó có thể xảy ra hoặc cut transition, hoặc fade transition nhờ việc đối sánh color histogram, ta sẽ trích chọn đặc trưng SIFT ở cặp frames (frame_1, frame_2) này, đồng thời cũng trích đặc trưng SIFT ở 3 frames liền trước frame_1 và 3 frames liền sau frame_2. Đối sánh sự tương đồng đặc trưng giữa từng cặp frames này.

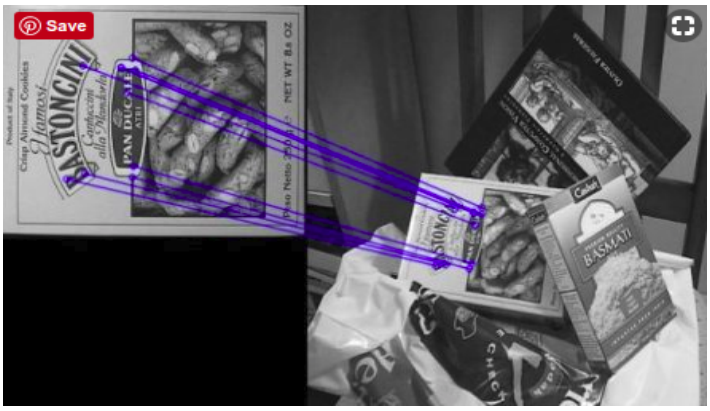
[frame_1_3; frame_1_2; frame_1_1; ***frame_1, frame_2;*** frame_2_1; frame_2_2; frame_2_3)]

Mà tại mảng frames này, frame_1, frame_2 là 2 frames đang được xét.

```
kp1, des1 = sift.detectAndCompute(img1, None)
kp2, des2 = sift.detectAndCompute(img2, None)
# BFMatcher with default params
bf = cv2.BFMatcher()
matches = bf.knnMatch(des1, des2, k=2)

# Apply ratio test
good = []
for m, n in matches:
    if m.distance < 0.75 * n.distance:
        good.append([m])
print("the similarity is", len(good))
return len(good)
```

Với đoạn mã trên , sau khi đã phát hiện và tính ra được đặc trưng SIFT của các frames, kết quả này sẽ được sử dung cho việc nối các đặc trưng, tức so khớp các đặc trưng cuả 2 frames. Mà feature matcher được sử dụng ở đây là Brute-Force Matcher. Trong đó(bf.knnMatch) với k = 2 sẽ trả về 2 lines match gần nhất với mỗi keypoint của mỗi frame. good là mảng sẽ lưu giữ lại những line nối các keypoint giữa 2 frames, mà line này là ngắn nhất.



Việc nối các keypoints có thể hình dung tương tự như hình trên.

Với hàm match trên, ta đã chọn giữ lại k = 2, tức 2 match gần nhất, và sau đó so sánh 2 match này bằng biểu thức:

```
m.distance < 0.75 * n.distance
```

Để có thể khẳng định liệu các line nối keypoints này có đủ gần (các keypoints đã đủ tương đồng) hay chưa.

Sau khi đã có được len(Good) (số lượng các keypoints gần như tương đồng giữa 2 frames) ta so sánh đại lượng này giữa 8 frames kể trên.

Nếu như đại lượng này giảm đột ngột tức nhỏ nhất ở (frame_1; frame_2) thì đây là cut transition. Còn nếu đại lượng này giảm dần cho đến (frame_1; frame_2) và lại bắt đầu tăng từ đây, thì đây là fade transition. Do thời gian tìm hiểu có hạn, nhóm sẽ không xét với trường hợp wide hay dissolve transition.

Tổng kết

Bên trên là phần tìm hiểu, đề xuất, triển khai chương trình của nhóm cho việc phát hiện shot boundary , sẽ không tránh khỏi những thiếu sót. Trong tương lai nhóm mong có thể tiếp tục phát triển hệ thống cho việc phát hiện được cả wipe transition, dissolve transition, hoặc có thể thay vì việc sử dụng trích chọn đặc trưng SIFT, có thể sử dụng SURF, để thấy được sự khác biệt, ưu nhược điểm của từng phương pháp.

Tài liệu tham khảo:

1. " A divide-and-rule scheme for shot boundary detection based on SIFT" - Jun Li, Youdong Ding, Yunyu Shi, Wei Li

2. OpenCV Docs

3. Slides môn học "Cơ sở dữ liệu đa phương tiện"