UNIVERSIDAD NACIONAL DE SAN AGUSTÍN

FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS

MAESTRÍA EN INFORMÁTICA

Curso:

Minería de datos

Trabajo Final:

Contar Escenas en Video.

Nombres:

Romero Ampuero, Wilmer

AREQUIPA- PERÚ

2018

Trabajo Final

1. Resumen

El presente trabajo tiene como objetivo contar el número de escenas que hay en un video. El video de prueba fue de la película Hachiko, en este trabajo se realizará el particionamiento del video en frames, posteriormente se extraerá las características de cada una de las imágenes. Se probarán diferentes técnicas para capturar características, y realizaremos una comparación con PySceneDetect el cual es una librería en python para detectar cambios de escenas en videos. El procedimiento que se utilizará será el siguiente: extraer las características de cada una de las imágenes y utilizar un algoritmo de clustering para agruparlos por sus características similares.

2. Introducción

En la literatura existen diferentes maneras de detectar cambios en escena en un video, y muchas técnicas para utilizar. PySceneDetect ofrece un conjunto de algoritmos para detectar cambios de escena, los algoritmos que posee son los siguientes:

Detector consciente de Contenido

El detector de escena sensible al contenido funciona de la misma manera que la mayoría de las personas piensa en "cortes" entre escenas en una película, dado dos fotogramas, ¿pertenecen a la misma escena o diferentes escenas? El detector de escenas con contenido consciente encuentra áreas donde la diferencia entre dos cuadros posteriores excede el valor de umbral que se establece (un buen valor para comenzar es - umbral 30).

Esto le permite detectar cortes entre escenas que contienen contenido, en lugar de como funcionan la mayoría de los métodos tradicionales de detección de escenas. Con un umbral establecido correctamente, este método incluso puede detectar cambios menores y abruptos, como los cortes de salto en la película.

Detector de umbral

El detector de escena basado en umbral (umbral -d) es cómo funcionan la mayoría de los métodos tradicionales de detección de escena (por ejemplo, el filtro de marco negro), comparando la intensidad / brillo del fotograma actual con un umbral establecido y activando un corte / pausa de escena cuando este valor cruza el umbral. En PySceneDetect, este valor se calcula promediando los valores R, G y B para cada píxel en el cuadro, produciendo un único número de coma flotante que representa el valor promedio de píxel (de 0.0 a 255.0).

3. Estado del Arte

Según Tiago Henrique Trojan [1] la segmentación de escenas en un video es un campo muy interesante. Pero se requiere utilizar técnicas que no tengan un alto costo computacional. Una técnica para la segmentación de video, menos reciente es la basada en la coherencia visual. Otras técnicas utilizan la operación del histograma, para comparar toma adyacente en busca de similaridades que puedan indicar la presencia de una escena. Para mejoras los resultados los autores usan otro conjunto de características, capaces de medir la cantidad de movimiento en las escenas. La extracción de metadatos puede ser realizado de manera automática o manual, aplicada a diversos tipos de media. En el caso del video, el primer paso es segmentar el video en unidades más pequeñas de información.

Según Yize Cui [2], propone un enfoque más efectivo para la detección de escenas en videos de noticias, para lo cual utiliza una red neuronal convolucional y las características de cada una de las escenas, previamente realiza la detección de shots, los cuales se combinan para la detección de un canal de noticias . El autor realiza pruebas con varios videos de noticias.

4. Metodología Utilizada

En esta sección mostramos la metodología utilizada:

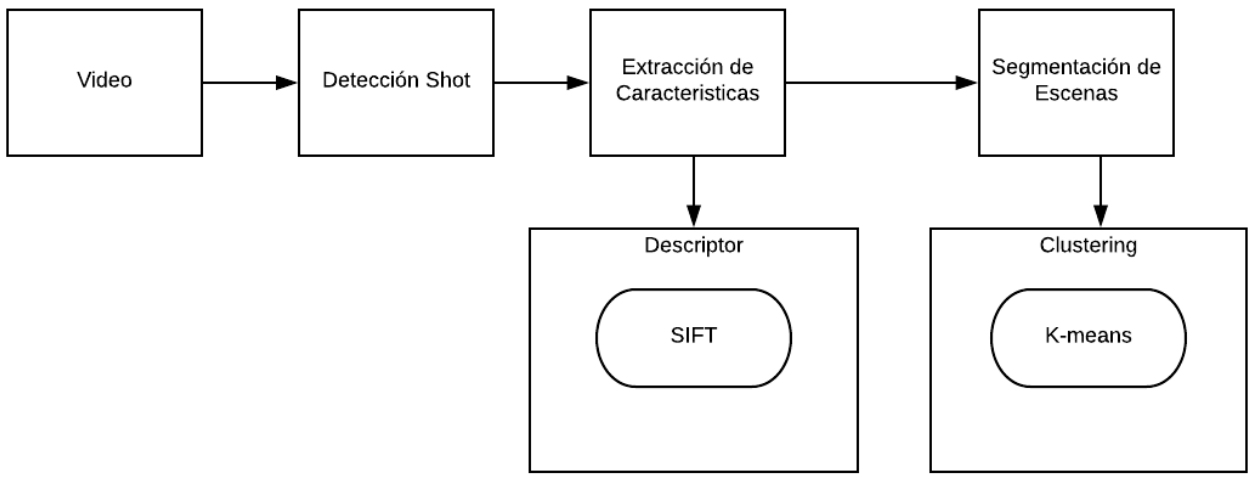

Figura N°1: Metodología utilizada

La detección de frames se realizó con el código proporcionado por el docente. Posteriormente a esto realizamos la extracción de características.

Para la extracción de características usamos el algoritmo SIFT. El extractor de características usado es referenciado por Yangqing Jia [3]. El algoritmo nos devuelve las siguientes características: **Altura, longitud, el espaciado para el**

muestreo de descriptores densos, el tamaño de cada parche tamizado, umbral de normalización de bajo contraste, a desviación estándar para el suavizado gaussiano antes de calcular el gradiente

```
D:\WILMER\UNSA\MAESTRIA\Datamining\ExtracionDeCaracteristicas>python dsift.py
Image: w 276, h 183, gs 8, ps 16, nFea 693
```

W	**Ancho de la imagen**
H	**Altura de la imagen**
Gs	**Espaciado para el muestreo de descriptores densos**
Ps	**El tamaño de cada parche tamizado**
nFea	**Número de características**

Tabla N°1: Características del descriptor denso SIFT

Otros autores indican que también sería posible determinar si dos imágenes son diferentes por la sustracción de sus distancias, para lo se requiere conocer los siguientes parámetros.

- El ancho (o alto) en alguna medida de distancia, como pulgadas o metros, del objeto que estamos utilizando como marcador.
- La distancia (en pulgadas o metros) de la cámara al marcador en el paso 1.
- Los algoritmos de visión artificial y procesamiento de imágenes se pueden usar para determinar automáticamente el ancho / alto percibido del objeto en píxeles y completar la similitud del triángulo y darnos nuestra distancia focal.
- Luego, en las imágenes siguientes, simplemente necesitamos encontrar nuestro marcador / objeto y utilizar la distancia focal calculada para determinar la distancia al objeto desde la cámara.

Podríamos decir que se detecta un cambio de escena si cambia el escenario. Otro descriptor que podríamos utilizar sería detectar el número de objetos en una imagen. En este caso podríamos utilizar el descriptor HOG.
A continuación mostramos las características que HOG pudo extraer.

```
D:\WILMER\UNSA\MAESTRIA\Datamining\ExtracionDeCaracteristicas>python dhog.py
Saving cat.txt ........

[8.17831451e-02 6.26310600e-02 5.27155189e-02 0.00000000e+00
 1.21863493e-01 3.12449555e-01 2.10140527e-01 2.50646643e-01
 2.57455235e-01 2.20661607e-01 3.08236042e-01 5.26329710e-02
 1.47439187e-02 1.48201160e-01 1.18584581e-01 2.94786967e-02
 6.99515846e-02 2.60750931e-02 4.94260496e-03 1.98581909e-02
 3.61375493e-03 8.41646127e-03 5.62277303e-01 2.97289591e-01
 1.12021874e-01 9.56731825e-02 8.86266931e-02 1.07965365e-01
 1.66341067e-01 8.99270683e-02 1.29832762e-01 1.13340699e-01
 2.26961892e-01 1.94526150e-01 2.71728152e-01 4.63990513e-02
 1.29976292e-02 1.30648016e-01 1.04539264e-01 2.59872003e-02
 3.28936765e-01 0.00000000e+00 5.81905002e-03 3.49046290e-03
 8.73595690e-03 8.66056603e-03 4.50789901e-03 5.51827811e-01
 9.87538526e-02 8.43415224e-02 7.81296287e-02 9.51778023e-02
 1.46639407e-01 7.92759857e-02 1.14455196e-01 9.99164747e-02
 3.91326702e-01 4.60758341e-02 2.46982428e-02 2.92762872e-02
 4.38862319e-02 1.92709023e-02 6.16724649e-03 3.88489544e-01
 3.86594105e-01 0.00000000e+00 6.83903617e-03 4.10228507e-03
 1.02672300e-02 1.01786244e-02 5.29806141e-03 6.48554377e-01
 2.28478968e-03 0.00000000e+00 0.00000000e+00 1.54086802e-03
 6.27267016e-03 2.08568612e-02 2.25256941e-02 1.55650725e-03
 4.59919997e-01 5.41521889e-02 2.90274487e-02 3.44079508e-02
 5.15787846e-02 2.26487824e-02 7.24826590e-03 4.56585531e-01
 7.54796614e-04 4.60186620e-04 6.79252072e-04 9.28201747e-04
 4.21947886e-03 9.05921071e-03 1.82936140e-02 3.39523947e-03
 6.48209831e-02 2.41626144e-02 4.58008942e-03 1.84016911e-02
 3.34870394e-03 7.79915562e-03 5.21037054e-01 2.75484876e-01
 1.03805625e-01 8.86560295e-02 8.21263652e-02 1.00046641e-01
 1.54140775e-01 8.33313643e-02 1.20310174e-01 1.05027722e-01
 4.31374711e-02 6.08355787e-03 2.00857324e-03 3.06387928e-03
 3.11887660e-03 1.59646705e-03 1.08463561e-01 4.18956696e-01
```

Figura N°2: Descriptor de HOG

El vector resultante es una matriz de 288 elementos. Según [4] el descriptor SIFT proporciona un número de características insuficientes por lo que usar el descriptor HOG es mucho mejor, dado que hay imágenes que están sometidas a la iluminación.

5. Resultados Alcanzados

Se realizó la de cambios de escena en el video Hachico. A continuación se muestra el estudio comparativo de las validaciones proporcionadas por el docente, así como las escenas detectadas por PyDetectScenes de python:

	Sin Nombre	**PyDetect Scene**	**Rafael Prandi**	**Guillerme Barranco**	**Otro1**	**Otro2**
Escenas Detectadas	57	40	68	86	65	51

El framework detectó un total de 40 escenas, podemos decir que esta cantidad es muy distante, por lo que podemos pensar que el framework no detecto todas las escenas y esto se debe a que probablemente que el umbral utilizado no fue el correcto. El archivo resultante tiene la siguiente estructura:

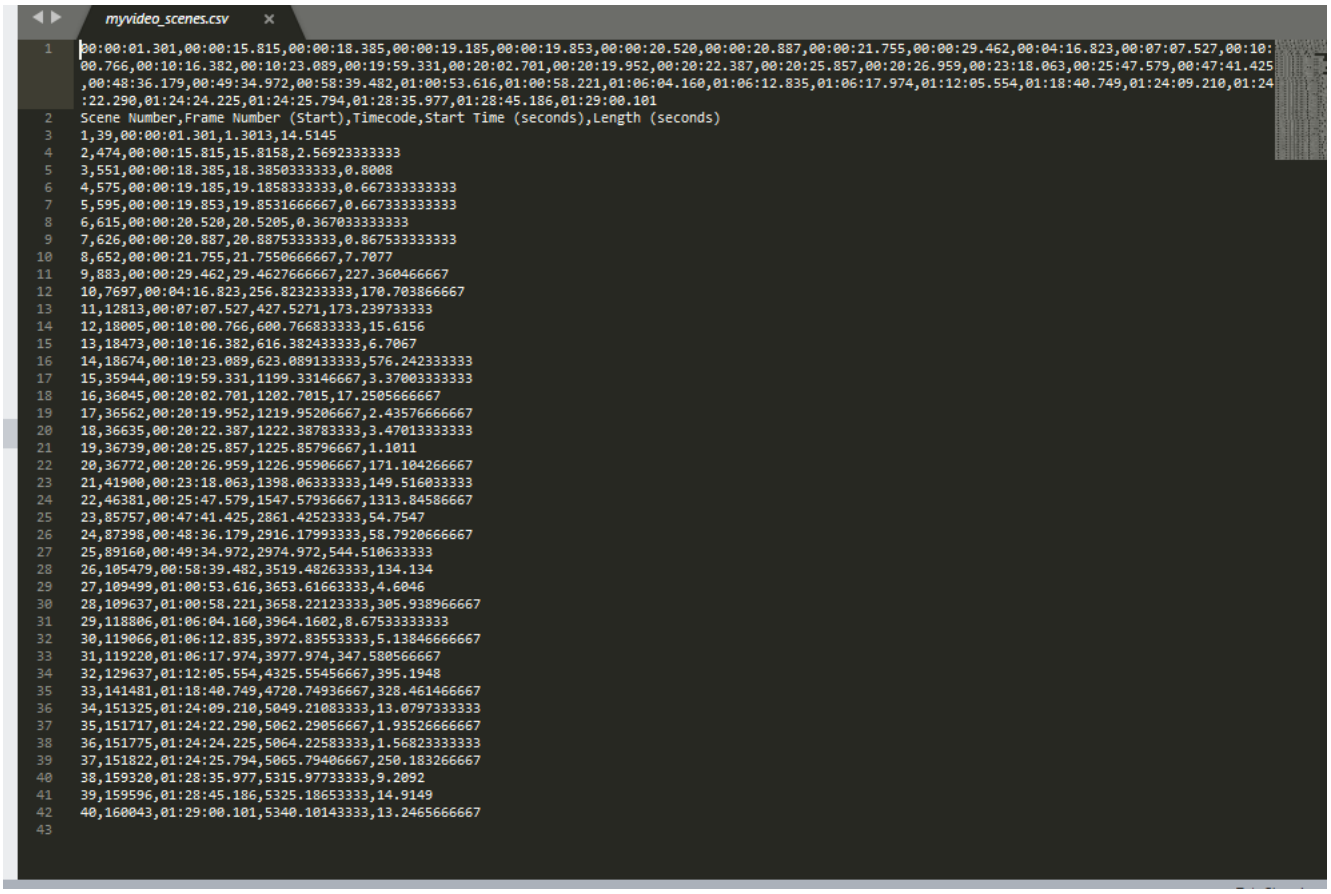

Figura N°3: Número de escenas detectadas por PyDetectScene

Podemos ver que la primera escena detectada se da en los siguientes casos:

	Sin Nombre	**Rafael Prandi**	**Guillerme Barranco**	**Otro1**	**Otro2**	**PyDetectScene**
Primera escena detectada	11	11	0	0	0	1

Podemos ver que en cuanto a la primero escena coinciden el framework, Guillerme Barranco, Otro 1 y Otro 2, pero si coinciden en un inicio porque la diferencia en el número de escenas detectadas, quizás la respuesta se da porque el framework al realizar el proceso de fragmentación, hay unos que omite ya que no los cuenta.

6. Discusión o reflexión crítica

- PyDetectScene permite detectar escenas en video, pero un tiempo de procesamiento muy alto. Depende del nivel del threshold que se agregue ya que a partir de este realizará la detección de la escena.
- PyDetectScene demoró aproximadamente unas dos hora en detectar todas las escenas del video.
- El vector de características es muy importante, los descriptores probados son el SIFT y el HOG.

7. Referencias bibliográficas

[1] T. H. Trojahn, “Segmentação automática de vídeo em cenas baseada em coerência entre tomadas.”

[2] Y. Cui, Y. Cai, C. Qiu, X. Gao, F. Of, and I. Technology, “Scene Detection of News Video Using CNN Features,” no. 61201360, 2017.

[3] Y. Jia and T. Darrell, “Heavy-tailed Distances for Gradient Based Image Descriptors,” pp. 397–405, 2011.

[4] D. F. Llorca, R. Arroyo, and M. A. Sotelo, “Vehicle logo recognition in traffic images using HOG features and SVM,” *IEEE Conf. Intell. Transp. Syst. Proceedings, ITSC*, no. Itsc, pp. 2229–2234, 2013.

[5] http://pyscenedetect.readthedocs.io/en/latest/features/

OBSERVACIONES Y CONCLUSIONES:

- Se encontraron útiles los siguientes comandos:
  Ver los contenedores en ejecución:
  # docker ps
  Listar todos los contenedores en el host para ser borrados:
  # docker ps -a
  Detener un contenedor:
  # docker stop <CONTAINER ID>
  Matar contenedores en ejecución:
  # docker kill <CONTAINER ID>
  Borrar contenedores por id:
  # docker rm <CONTAINER ID>
  Listar todas las imágenes:
  # docker images
  Borrar imágenes por id:
  # docker rmi <IMAGE ID>
  Limpiar todo el docker:
  # rm -R /var/lib/docker
- Los contenedores creados por Docker son mínimos, y muchas veces nuestras aplicaciones requieren de paquetes adicionales, así vemos pudimos ver los requerimientos de nuestra aplicación hicieron que la imagen pesara de 100MB a 1GB.
- Docker nos permite experimentar diferentes distribuciones independientemente de la utilizada para el host.

REFERENCIAS:

[1] Docker por ArchWiki recuperado de: https://wiki.archlinux.org/index.php/Docker
[2] Arch Linux Docker Tutorial por David Morelo recuperado de: https://linuxhint.com/arch-linux-docker-tutorial/
[3] [SOLVED] Docker only runs as root, even though user is in docker group por mascip, recuperado de: https://bbs.archlinux.org/viewtopic.php?id=193845
[4] How To Install Python 3 and Set Up a Local Programming Environment on Debian 8 por Lisa Tagliaferri recuperado de: https://www.digitalocean.com/community/tutorials/how-to-install-python-3-and-set-up-a-local-programming-environment-on-debian-8