Aseguramiento de la Calidad del Software: Proyecto Semestral II

M. Sc. Saúl Calderón Ramírez
Instituto Tecnológico de Costa Rica,
Escuela de Ingeniería en Computación,
PAttern Recognition and MAchine Learning Group (PARMA-Group)

29 de julio de 2016

El presente proyecto pretende ser desarrollado en grupos de tres personas, a lo largo del curso de aseguramiento de la calidad del software. El proyecto será insumo para aplicar distintos estándares de calidad en sus varias etapas, por lo que a lo largo del curso se realizarán tareas y agregados específicos al proyecto. El proyecto trata sobre el desarrollo de un sistema automático de segmentación temporal, lo que se refiere a la detección de cortes en videos digitales.

Fecha de entrega: 31 de octubre.

1. Introducción y motivación

La masificación del internet, consecuencia del desarrollo de tecnologías de transimisión y compresión han llevado al uso generalizado y la disponibilidad de video digital. Aplicaciones tales como bibliotecas digitales, aprendizaje a distancia, vídeo bajo demanda, vídeo digital difusión, la televisión interactiva y la información multimedia sistemas generan utilizan grandes colecciones de videos digitales. Esto ha creado la necesidad de herramientas que pueden e indizar , buscar, explorar, y analizar este tipo de datos.

Ejemplo de estos sistemas de análisis automático de vídeo, es el sistema ACE, desarrollado actualmente en el PRIS-Lab, en la Universidad de Costa Rica. Este sistema tiene por objetivo analizar de manera automatica videos de futbol extraídos de transmisiones televisivas. La segmentación temporal en videos digitales se refiere a la distinción de cuadros de corte o transición entre dos secuencias de video distintas. La primer etapa del sistema ACE para analizar el video, vendría dada entonces por la segmentación temporal del video, para así definir las distintas secuencias en el video, y posteriormente clasificar las escenas según su importancia (descartando acercamientos al público y anuncios por ejemplo) y realizar el análisis del movimiento, tácticas y estrategias de los

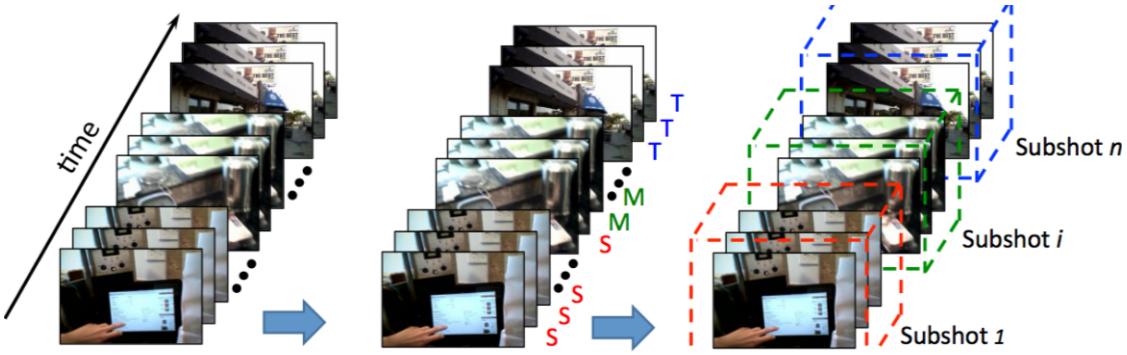

Figura 1: Diagrama general del proceso de segmentación temporal.

jugadores. En el presente proyecto se propone desarrollar el algoritmo de segmentación temporal, basado en el algoritmo propuesto en [2, 1, 3]. La Figura 1 muestra el proceso general de un algoritmo de segmentacion temporal.

2. Algoritmo de segmentación temporal

Un cambio o corte en un video digital, se puede definir como un cambio abrupto o gradual entre dos cuadros consecutivos de un video: U_{t-1} y U_t. El algoritmo a implementar utiliza como descriptor el histograma $h_t(z)$ para cada cuadro o imagen U_t. Un histograma se define como un arreglo que contabiliza la cantidad de apariciones de la escala de gris z (para una imagen en escala de grises $0 \leq z \leq M$, con $M = 255$), normalizado a la cantidad de pixels N en la imagen.

Básicamente, el algoritmo por cada par de cuadros U_{t-1} y U_t calcula los histogramas h_{t-1} y h_t y cuantifica la disimilitud entre ambos histogramas, con el funcional $d\left(h_{t-1}, h_t\right)$. La medida de disimilitud o distancia puede venir dada por la distancia euclidiana, de Mahalanobis o de Bathacharyya (esta última es la que se usará en el presente trabajo). La disimilitud entre cuadros consecutivos se representa entonces en el arreglo g, y tiene un aspecto como el mostrado en la Figura 2, donde se puede observar la presencia de «picos» altos para cada par de cuadros, lo que denota la posible ocurrencia de un corte. A partir entonces del arreglo g se define un umbral óptimo, usando estadística básica, para clasificar un cuadro como de corte o no corte.

A continuación se presenta

1. Abra el vídeo de prueba provisto, e itere por cada uno de los cuadros, leyendo la imagen RGB correspondiente, y transformándola al espacio de color HSV, con la función $rgb2hsv$ en MATLAB, o similar en la librería *OpenCV*. Normalice de 0 a 255 la capa H.

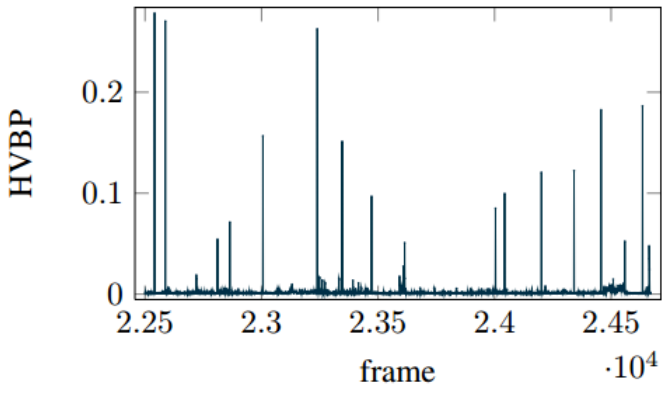

Figura 2: Disimilitud entre cuadros consecutivos usando la distancia de Bhattacharyya modificada en [2].

2. Calcule los histogramas normalizados de cada uno de los cuadros usando únicamente la capa H, implementando una función para ello.

3. Implemente la función de distancia entre dos histogramas $d_B(h_1, h_2)$, que retorne la disimilitud entre dos histogramas normalizados usando la **distancia de Bhattacharyya**, la cual se define de la siguiente manera:

$$d_B(h_1, h_2) = \left(\frac{1}{\sqrt{1 - \alpha\beta}} \right),$$

donde el coeficiente de Bhattacharyya $\beta = \sum_z \sqrt{h_1(z)\, h_2(z)}$ y el coeficiente de normalización

$$\alpha = \frac{1}{\sqrt{\overline{h}_1\, \overline{h}_2\, M^2}},$$

donde $\overline{h}_i$ se refiere al valor medio del histograma.

4. Calcule el arreglo de disimilitud entre cuadros consecutivos $g(t) = d_B\left(h_{t-1}, h_t\right)$.

5. A partir del arreglo g calcule su media g_μ y desviación estándar g_σ, y suponiendo que las distancias entre cuadros consecutivos siguen una distribución normal, clasifique cada cuadro U_t usando la propiedad de los 3 sigmas:

$$\begin{array}{c} U_t \in C \text{ si } g(t) \geq g_\mu + g_\sigma \\ U_t \in N \text{ sino} \end{array},$$

donde C corresponde al conjunto de cuadros que son cortes, y N al conjunto de cuadros que no lo son.

3. Requerimientos de la aplicación

La aplicación debe proveer una interfaz gráfica de usuario amigable, que permita (por orden de prioridad):

1. Cargar un video digital almacenado en la dirección provista por el usuario.
2. Los N videos correspondientes a las N escenas detectadas.
3. Visualizar la cantidad de cortes detectados.
4. Visualizar cada una de las escenas detectadas.
5. Visualizar la energía promedio de los cortes y no cortes según g.
6. Visualizar el tiempo en procesar el vídeo.
7. **Generar un archivo de groundtruth donde se especifque en que cuadros hay un corte.**
8. **Métrica de exactitud en los resultados:** Cargar un archivo de *ground truth* en el que el número de frame en el que sucedió un corte esté manualmente marcado, *y* que permita además cuantificar la cantidad de falsos positivos y negativos en la detección de cortes (Investigar otras métricas además).

4. Implementación

Para el diseño e implementación del proyecto, es necesario usar la metodología de desarrollo Scrum, usando alguna herramienta para gestión de proyectos en línea como *zoho projects.* Como herramientas de diseño, se recomienda usar *StarUML*, *Altova*, *Visio*, etc. Para promover el atributo de mantenibilidad del proyecto, al menos un patrón de diseño debe ser implementado.

Respecto a la codificación del proyecto, es muy recomendado el uso del lenguaje Java, usando Eclipse como ambiente de desarrollo, pues existen herramientas y *plugins* relacionados con el cálculo de métricas y gestión de la calidad del proyecto, lo cual será un rubro importante a evaluar. **Los distintos estándares y métricas se aplicarán a lo largo del curso.** Es conocido el soporte de Java con OpenCV además, librería de algoritmos de visión por computador y aprendizaje automático.

La documentación final del proyecto debe incluir todos los estándares implementados a lo largo del curso, incluyendo los realizados en tareas, de manera consistente. Es el obligatorio el uso de LaTeX o algún editor basado en esa tecnología como LyX para la documentación del proyecto. Para facilitar la edición colaborativa, se recomienda el uso de la herramienta *Overleaf.*

Referencias

[1] F. S. Canales, "Ace-football, analysing football from tv broadcasting," *24th German Soccer Conference DVS – Fusball in Lehre und Forshung*, 2013.

[2] F. Siles, "Temporal segmentation of association football from tv broadcasting," in *Intelligent Engineering Systems (INES), 2013 IEEE 17th International Conference on*. IEEE, 2013, pp. 39–43.

[3] F. Siles Canales, "Temporal segmentation of association football from tv broadcasting," in *IEEE 17th International Conference on Intelligent Engineering Systems – INES 2013*. San Jose, Costa Rica: IEEE, 2013.