Aseguramiento de la Calidad del Software: Proyecto Semestral I

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

28 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 objetivo es desarrollar un sistema que segmente y clasifique por equipo los jugadores de futbol en un video digital.

Fecha de entrega: 31 de octubre.

1. Introducción y motivación

El análisis automatizado de videos se ha visto estimulado con la masificación de internet, las plataformas de computación de alto rendimiento a bajo costo, y el diseño de nuevos algoritmos para su procesamiento eficiente. Una aplicación del análisis automatizado de videos es la obtención de datos de alto nivel a partir de videos digitales de futbol, estos datos corresponden al uso de estrategias y tácticas por cada uno de los equipos de futbol. El sistema ACE, actualmente desarrollado en el PRIS-Lab www.pris.eie.ucr.ac.cr, de la Universidad de Costa Rica tiene por objetivo analizar de manera automática videos de futbol. Sus etapas van desde la identificación de escenas con información útil en el video, la segmentación y rastreo de jugadores, hasta el análisis automático de los recorridos y comportamiento de los jugadores.

En el presente proyecto, se propone el desarrollo de la etapa de segmentación de jugadores y la clasificación de las regiones en la imagen correspondientes a los jugadores o «blobs» por color, de manera no supervisada, para determinar la pertenencia de cada jugador a alguno de los equipos. El algoritmo de clasificación no supervisada a implementar es el conocido *K-medias*, muy utilizado en diversas aplicaciones donde se requiere clasificar o amontonar un conjunto de datos en K clases, sin que sea necesaria una etapa de

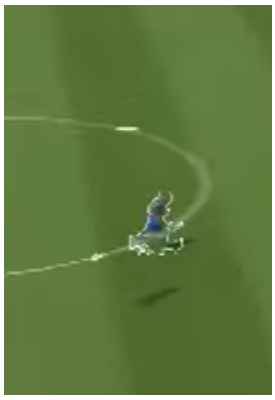

Figura 1: Segmentación de jugadores.

entrenamiento previa [1].

2. Algoritmo a implementar

El algoritmo a implementar está compuesto por dos etapas. La primer etapa detectará los pixeles correspondientes a regiones representando a los jugadores o «blobs». Esta etapa se refiere a la «segmentación de jugadores» y se ilustra en la Figura 1. La región o «blob» de cada jugador en la imagen se define como $R_{i,t}$, correspondiente al jugador i en el cuadro del vídeo U_t (donde t define el número de cuadro) . La segmentación de un video $\mathcal{V} = \langle U_1, U_2, \ldots, U_T \rangle$, compuesto por T cuadros resulta entonces en un conjunto de regiones $\mathcal{R} = \langle \mathbb{R}_1, \mathbb{R}_2, \ldots, \mathbb{R}_T \rangle$, donde $\mathbb{R}_t$ se refiere al subconjunto de regiones de los L jugadores presentes en el cuadro t: $\mathbb{R}_t = \langle R_{1,t}, R_{2,t}, \ldots, R_{L,t} \rangle$.

Para cada una de las regiones $R_{i,t}$ en el conjunto de regiones del vídeo $\mathcal{R}$, se calculan los histogramas de color o cromaticidad (usando la capa H o Hue del modelo de color HSV), denotados por $h_{i,t}$ para la región $R_{i,t}$, definiendo así el conjunto de histogramas para todos los jugadores en todo el vídeo $\mathcal{H}$. Tal conjunto de histogramas será la entrada del algoritmo de clasificación no supervisada *K-medias* el cual asignará de manera autónoma una etiqueta a cada histograma, lo cual se repesenta en la función de membresía $\mathcal{M}(h_{i,t}) = k$, donde $k = 1,\ 2$ (etiqueta de pertenencia al equipo 1 o 2).

2.1. Algoritmo de segmentación

El algoritmo de segmentación propuesto usa la información conocida de antenamo relacionada con el hecho de que los jugadores se encuentran siempre rodeados de pixeles verdes (campo de juego). Es por esto que el algoritmo se subdivide en dos etapas no necesariamente secuenciales:

- Detección del campo de juego o construcción de la «máscara del campo de juego» F_t para cada cuadro del vídeo U_t.
- Detección de las regiones candidatas de jugadores (tomando en cuenta al público) $C_{j,t}$ para cada cuadro del vídeo U_t, contenidas en la «máscara de candidatos a jugadores» P_t.

Para obtener las regiones correspondientes a los jugadores (blobs de personas dentro del campo de juego) se realiza un AND entre las máscaras ($P_t \& F_t$), por lo que entonces $R_{i,t} = C_{j,t} \& F_t$. A continuación se detalla el procedimiento, ilustrado con las funcionalidades en MATLAB correspondientes.

2.1.1. Detección del campo de juego

Para detectar el campo de juego, se siguen los siguientes pasos.

1. Convertir la imagen de entrada U_t a una imagen de cromaticidad H_t tomando la capa de cromaticidad H después de usar la función *rgb2hsv.*
2. Encontrar los valores α_{min} y $\alpha_{\text{máx}}$ que definan un rango de «verdosidad», para umbralizar la imagen en ese rango y obtener una máscara binaria de pixeles «verdosos» G_t a partir de H_t .
3. Rellenar los hoyos en la máscara G_t, posiblemente correspondientes los jugadores, usando la función *imfill*, resultando en G'_t.
4. Eliminar «regiones espurias» pequeñas (menos del 10 % de la imagen), usando la función *bwareaopen*, resultando en la máscara G''_t.
   a) Posiblemente queden hoyos no rellenados desde la máscara G_t (usualmente asociados a pequeñas regiones cerca de los bordes), por lo que se puede calcular el complemento de la máscara G''_t con *imcomplement* y eliminar de nuevo regiones pequeñas con *bwareaopen.* Calcular de nuevo el complemento, almacenando el resultado en G'''_t
5. Eliminar el logo del «marcador del partido», resultando en la máscara F_t.

2.1.2. Detección de regiones candidatas a jugadores

Para obtener las M regiones candidatas a corresponder a jugadores $C_{j,t}$ con $j = 1, \ldots, M$ del cuadro U_t el algortimo propuesto consiste en:

1. Convertir la imagen de entrada U_t a una imagen de cromaticidad H_t tomando la capa de cromaticidad H después de usar la función *rgb2hsv.* Se aconseja normalizar de 0 a 255 la imagen.
2. Calcular la imagen de *varianza local* S_t usando como entrada la imagen H_t. En MATLAB ello se puede realizar con la función *stdfilt* (a cuyo resultado se le debe calcular el cuadrado a cada elemento).

Figura 2: Imagen original y máscara del campo de juego F_t.

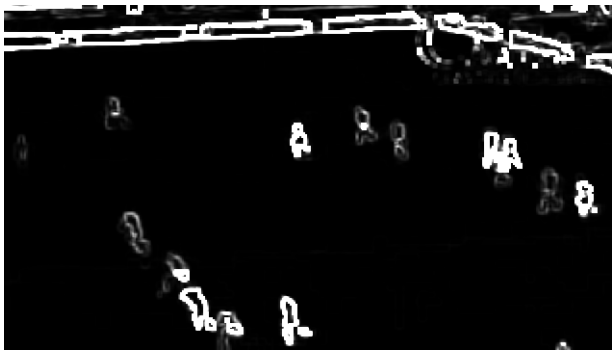

Figura 3: Resultado el cálculo de la varianza local y su truncamiento.

a) La varianza local, calcula, para una imagen U de $A \times B$ pixeles la varianza para toda ventana de $C \times D$ pixeles, de la siguiente forma:

$$S(x,y) = \frac{1}{C\,D} \sum_{i=x-\Delta_1}^{x+\Delta_1} \sum_{j=y-\Delta_2}^{y+\Delta_2} \left(U(i,j) - \mu(x,y)\right)^2$$

donde $\mu(x,y)$ se define como el valor medio en la ventana de $C \times D$ pixeles centrada en (x,y), con $\Delta_1 = \frac{C-1}{2}$ y $\Delta_2 = \frac{D-1}{2}$:

$$\mu(x,y) = \frac{1}{C\,D} \sum_{i=x-\Delta_1}^{x+\Delta_1} \sum_{j=y-\Delta_2}^{y+\Delta_2} U(i,j)$$

b) Si la imagen H_t fue normalizada de 0 a 255, truncar la imagen de varianza local S_t a 255, de modo que $S_t(x,y) \leq 255$. El resultado se muestra en la Figura 3.

3. Calcular el umbral óptimo τ para la imagen S_t, usando el algoritmo de Otsu o Kittler de umbralización por máxima verosimilitud, implementado en la función *graythresh* de MATLAB.

   a) Aplicar el umbral con la función *im2bw* y rellenar los hoyos con *imfill* obteniendo la máscara P_t con todas las regiones candidatas $C_{j,t}$, mostrada en la Figura 4.

2.2. Introducción al aprendizaje no supervisado

El aprendizaje no supervisado consiste en particionar un conjunto de N datos de entrada $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ (donde cada muestra es de dimensión D, $\mathbf{x} = \{x_1, x_2, \ldots, x_D\}$) en K grupos (montones o clústers), máximamente homogeneos. Para el caso del presente proyecto, cada muestra $\mathbf{x}_i = h_i$ corresponde

Figura 4: Máscara P_t.

Figura 5: «Blobs» encontrados en la imagen.

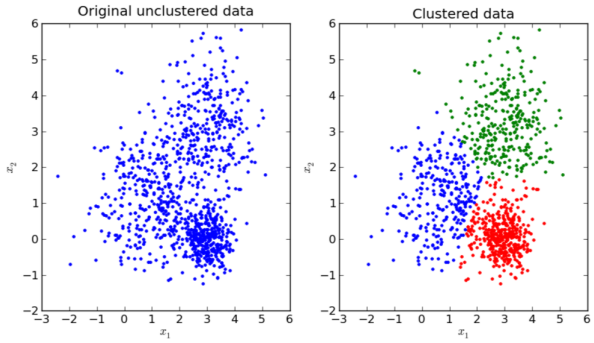

Figura 6: Resultado de un algoritmo de amontonamiento típico.

un histograma lo que hace que $D = 255$. El conjunto de los histogramas para todos los jugadores en T cuadros del video es $\mathbf{X}$.

El algoritmo *K-medias* asigna de manera automática una etiqueta $k_i \in 1, \ldots, K$ para cada muestra, de modo que si $\mathcal{M}$ corresponde a la función de membresía $\mathcal{M}(\mathbf{x}_i) = k_i$. Ello que significa que el algoritmo *K-medias* define la función de mapeo a las clases o membresía $\mathcal{M}$. Si una etiqueta correcta o real para la muestra i se define como $r_i \in 1, \ldots, K$, ello implica que la membresía correcta se representa en la función $\mathcal{R}(\mathbf{x}_i) = r_i$ y el algoritmo *K-medias* logra un 100 % de correctitud si $\mathcal{M} = \mathcal{R}$. La Figura 6 muestra el resultado final típico de un algoritmo de amontonamiento, e ilustra el proceso de asignación de etiquetas automático, prescindiendo del conjunto de etiquetas correctas necesario en el problema de clasificación supervisada.

2.2.1. El algoritmo de amontonamiento K-medias

A partir de la notación introducida en la sección anterior, se resumen los aspectos importantes del algoritmo K-medias.

- **Objetivo general:** Partir el conjunto de datos de entrada $\mathbf{X}$ en K nuevos conjuntos $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_K$ cuya **varianza intra-clase sea la menor posible**. Para lograrlo, el algoritmo de K-medias propone en P iteraciones ajustar las K medias de cada cluster $k = 1, \ldots, K$, minimizando la varianza intra-clase. El número de clases K y de iteraciones a ejecutar P son conocidas. En el caso del presente proyecto, se supondrán $K = 2$ (dos equipos presentes en los videos de futbol). Las siguientes son definiciones utilizadas en la construcción del algoritmo.

  - **Cluster:** Conjunto de datos $\mathbf{X}_k$, los cuales tienen una distancia *pequeña* entre ellos (distancia intra-cluster), comparado con los datos de otros clusters $\mathbf{X}_j$ (distancia inter-cluster). A cada cluster $\mathbf{X}_k$ le

corresponde una *muestra media* μ_k, definida como:

$$\mu_k = \frac{\sum_{n=1}^{N} w_{n,k}\mathbf{x}_n}{\sum_{n=1}^{N} w_{n,k}},$$

donde:

- la matriz de pesos o matriz de membresía $\mathbf{W} = \begin{bmatrix} w_{1,1} & \dots & w_{1,K} \\ \vdots & \ddots & \vdots \\ w_{N,1} & \dots & w_{N,K} \end{bmatrix}$ de dimensiones $N \times K$, determina la pertenencia de la muestra n a la clase k (definiendo así la función de membresía $\mathcal{M}(\mathbf{x}_n) = k$), de modo que:

$$w_{n,k} = \begin{cases} 1 & k = \text{mín}_j\, d(\mathbf{x}_n - \mu_j) \\ 0 & \text{de lo contrario} \end{cases}.$$

  En otras palabras, el peso $w_{n,k}$ se asigna como $w_{n,k} = 1$ para una muestra $\mathbf{x}_n$ cuyo cluster con media μ_k es el más cercano y es $w_{n,k} = 0$ en caso contrario. Lo anterior, en palabras más simples, representa el hecho que la *muestra media* μ_k es el promedio de todas las muestras pertenecientes al cluster k.

- **Función de costo:** Sumatoria de las distancias de todas las muestras $\mathbf{x}_n$ que pertenecen a la clase k, respecto a la muestra media μ_k, para todas las K clases:

$$J(\mathbf{X}, \mu) = \sum_{n=1}^{N}\sum_{k=1}^{K} w_{n,k} d(\mathbf{x}_n - \mu_k),$$

  donde:

  - la función de distancia $d(\mathbf{x}_n - \mu_k)$ mide la disimilitud entre las muestras $\mathbf{x}_n$ y μ_k de dimensión D, de modo que si $d(\mathbf{x}_n - \mu_k) \to \infty$, las muestras $\mathbf{x}_n$ y μ_k son más disimiles. Ejemplos de distancias: distancia de euclidiana, Bhattacharyya, Kolgomorov, etc. Para el presente proyecto se utilizará la distancia de Bhattacharyya.
  - El vector de medias $\boldsymbol{\mu} = [\mu_1, \mu_2, \dots, \mu_K]$ está compuesto por las medias de cada clase k.

- **Flujo del algoritmo K-medias:** Basados en las definiciones anteriores, el algoritmo K-medias kMedias$(\mathbf{X}, K, P)$, ejecutado para K clusters y en P iteraciones, consiste en lo siguiente:

  1. Inicializar aleatoriamente las K medias $\boldsymbol{\mu} = [\mu_1, \mu_2, \dots, \mu_K]$.
  2. Por cada iteración $p = 1, \dots, P$, ejecutar:

a) **Fase de etiquetamiento de las muestras o *expectativa:*** Calcular para todas las muestras en $\mathbf{X}$ el vector de pesos w la pertenencia a cada cluster k, según las medias μ, como se especificó anteriormente:

$$w_{n,k} = \begin{cases} 1 & k = \min_j d\left(\mathbf{x}_n - \mu_j\right) \\ 0 & \text{de lo contrario} \end{cases}.$$

b) **Fase de minimización del error o maximización de la verosimilitud:** la maximización de la verosimilitud equivale a la minimización de la función de error, en este caso $J(\mathbf{X}, \mu)$, tiene su mínimo error cuando:

$$\mu_k = \frac{\sum_{n=1}^{N} w_{n,k}\mathbf{x}_n}{\sum_{n=1}^{N} w_{n,k}}.$$

Por lo que entonces en esta etapa básicamente se re-calculan las K medias μ.

3. El vector de pesos resultante w es utilizado para realizar el etiquetamiento por cada clase en el conjunto de muestras $\mathbf{X}$.

Alternativamente el algoritmo de K-medias se puede detener utilizando algún criterio de convergencia, como por ejemplo el cambio en las medias μ.

La Figura 7 muestra el funcionamiento del algoritmo K-medias.

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. Descargar el video con los blobs y las etiquetas por equipo marcados.
3. Generar un informe en formato *csv* de la cantidad de jugadores de cada equipo, por cuadro.
4. Visualizar el número de jugadores detectados en cada cuadro, conforme trascurre el video.
5. Visualizar el vídeo, con las regiones correspondientes a los jugadores o «blobs» visibles.
6. Visualizar las etiquetas por equipo (2 etiquetas) en los «blobs» correspondientes a los jugadores, a lo largo del vídeo.
7. Visualizar el tiempo en procesar el vídeo.
8. **Métrica de exactitud en los resultados:** Cargar un archivo de *ground truth* con los blobs de los jugadores manualmente marcados, que permita además cuantificar la cantidad de fallos en la clasificación no supervisada.

(Loading...)

Figura 7: Funcionamiento del algoritmo K-medias. Tomado de [**?**].

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 estándares de calidad a aplicar se agregarán 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] Francisco Siles Canales. Ace-football, analysing football from tv broadcasting. *24th German Soccer Conference DVS – Fusball in Lehre und Forshung*, 2013.