Un estudio basado en la Técnica de Mean Shift para Agrupamiento y Seguimiento en video

Alumno: Nakama Martín Ignacio

Directora: Lic. Maria Elena Buemi

Co-Director: Dr. Julio Jacobo

Departamento de Computación, FCEyN,Universidad de Buenos Aires.

3 de agosto de 2011

Resumen

En el campo del procesamiento de imágenes y visión por computador los algoritmos de agrupamiento, segmentación y seguimiento son frecuentemente usados. En este trabajo se realiza un estudio de la toeria de *Mean Shift* y su aplicación para implementar los algoritmos antes mencionados. *Mean Shift* es una técnica no paramétrica para el análisis de un conjunto de puntos d-dimensionales en un espacio de características que obtiene un máximo local de la función de densidad estimada para la muestra. Cada uno de los algoritmos enunciados se basan en encontrar los máximos de una funcion estimada la cual tiene un significado diferente para cada caso en particular.

En lo que respecta al seguimiento, se ha implementado un predictor basado en la teoría del *Filtro de Kalman* para hacerlo robusto a oclusiones parciales y totales en un periodo corto de tiempo. Además se realiza un ajuste en el algoritmo de seguimiento para detectar cambios de escala del objetivo o *target* respecto a la posición fija de la cámara de video.

Abstract

In the area of image processing and computer vision; clustering, segmentation and tracking are frequently used. In this work, we conduct a study on *Mean Shift* theory and its applications to those techniques. The *Mean Shift* procedure is based on a non-parametric technique for the analysis of a set of d-dimensional points in the feature space to obtain the local maximum of the estimated density function asociated with the sample. This will be used for each of the techniques mentioned above.

In regards to tracking, a predictor based in *Kalman's Filter* theory was implemented in order to be robust to partial and total oclussions in a short period of time. Furthermore, we make an improvement to the tracking algorithm to detect changes in scale of the target with respect to the position of the video camera.

Agradecimientos

Quiero aprovechar este espacio, para agradecer en primer lugar a mis padres, quienes me bancaron y supieron transmitirme los valores necesarios que hacen que hoy, pueda alcanzar esta instancia en mi vida académica. En segundo lugar, quiero agrader a mi directora Maria Elena Buemi y especialmente a mi co-director Julio Jacobo por su dedicación, paciencia y generosidad al guiarme con todo su conocimiento para llevar a cabo esta tesis.

También quiero agradecer a todos a aquellos que de alguna manera me han dado una mano con este trabajo, ya sea por alguna consulta con latex, matlab, un grafico o una deducción algebraica. Agradezco a todos con los que compartí en estos largos años, tardes de estudio y trabajos prácticos interminables entre risas y teoremas. Con ellos pasé, sin darme cuenta, una etapa importante de mi vida; momentos amargos y felices, angustias y alegrias por los resultados obtenidos.

Gracias Pachi, Leo, Cynthia, Martín, Fede y a todos los que han estado apoyándome y alentándome en todo momento. Mi agradecimiento infinito a la educación pública, es un placer y un lujo para mí haber llegado al final del camino de una de las carreras de esta prestigiosa Facultad.

Índice general

Capítulo 1

Introducción

En el campo de procesamiento de imágenes y visión por computador, se destacan tres técnicas: el agrupamiento, la segmentación y el seguimiento de objetos en video. El agrupamiento (o en inglés *clustering*) es un procedimiento de agrupación de datos de una muestra de acuerdo a un criterio de cercanía y tiene varias aplicaciones relacionadas con la minería de datos, teoría de las comunicaciones, medicina, imágenes de satélite, etcétera. La segmentación es el proceso de dividir una imagen en regiones (conjunto de píxeles) a fin de simplificar su representación y posterior análisis. El seguimiento (o *tracking*) de objetos en tiempo real es actualmente de gran relevancia en muchas aplicaciones en visión como cámaras de vigilancia e interfaces de percepción de usuarios, entre otros.

Mean Shift es una técnica no paramétrica para el análisis de una muestra en un espacio de características presentada originalmente en 1975 por Fukunaga y Hosteler[FH75], que más tarde fue estudiado por Cheng[Che95] para el análisis de imágenes y extendido más recientemente por Comaniciu, Meer y Ramesh[CM99] para problemas de visión de bajo nivel como son la clusterización, segmentación y seguimiento de objetos en video.

En el Capítulo 2 se presentará a las *Ventanas de Parzen* como una técnica en donde a partir de una conjunto de puntos d-dimensionales de una muestra en un espacio de características, se obtiene una función de densidad estimada (*pdf*) para dicha muestra. Esta *pdf* presentará máximos locales o modas en regiones densas y bajos valores en regiones poco pobladas. En el Capítulo 3 se presentará el procedimiento de *Mean Shift* para obtener los máximos locales de una *pdf*, puesto que el método estima en cada iteración el gradiente de la función de densidad estimada. Se explicarán las bases del método y se demostrará el criterio de convergencia. En el Capítulo 4 se presentará un algoritmo de *clustering*[CM98] de orden de complejidad lineal a fin de poder clasificar los pixeles de una imagen y proponer de esta manera una segmentación. Además en el Capítulo 5, se mostrará otro algoritmo en el dominio empalmado o *joint-domain*[CMM02] con el cual se obtienen mejores

resultados puesto que tiene en cuenta la disposición espacial de los pixeles en la imagen aunque cuenta con un orden de complejidad mayor.

En lo que respecta al *tracking*[CRM+03], en el Capítulo 6 se implementa un algoritmo basado en la *Rerepsentación y Localización del objetivo o target* para el seguimiento de objetos no rígidos. La representación del *target* se realiza con un histograma ponderado por un *kernel* isotrópico formulado de tal manera que la localización se logra maximizando el coeficiente de *Bhattacharyya*, siendo *Mean Shift* el procedimiento para efectuarlo. Para obtener robustez a oclusiones parciales y/o totales, en el Capítulo 7 se integra un predictor basado en la teoria del *Filtro de Kalman*. Asimismo se realiza un agregado para que también puedan ser detectados cambios en la escala del objetivo en caso de existir acercamiento y/o alejamientos respecto a la cámara de video.

Los resultados de los experimentos se encuentran al final de cada sección. Una conclusión final del trabajo en su totalidad para esta tesis se encuentra en el Capítulo 8.

Capítulo 2

Técnicas no-paramétricas

Dada una muestra de datos, existe una gran variedad de métodos para el procesamiento y análisis estadístico a fin de convertir esta muestra en información significativa para un estudio en particular. Las técnicas para la extracción de características importantes de una muestra pueden clasificarse dentro de dos grandes grupos, éstas son las *paramétricas* y las *no paramétricas*.

Las técnicas no paramétricas, a diferencia de las paramétricas, se enfocan en el reconocimiento de patrones sin suponer nada acerca de la distribución de los valores en el espacio de características. Puesto que en los enfoques no paramétricos no se cuenta con ningún tipo de información a priori, los resultados son menos confiables si se comparan con el otro enfoque, aunque pueden ser utilizados en una más amplia diversidad de casos.

Los métodos no paramétricos pueden ser utilizados para clasificar datos de una muestra bajo algún criterio pero no para ser interpretados de forma clara y exacta. En este trabajo no se necesita de una estimación precisa puesto que el objetivo es diseñar un clasificador para evaluar su desempeño sin contar con información a priori de la muestra. Dado un conjunto de valores $\mathbf{x}_1, ..., \mathbf{x}_n$ dispuestos en el espacio de características, se infiere una función de densidad de probabilidad (*pdf*) estimada.

Existen muchas técnicas para una estimación no paramétrica aunque dos de ellas se destacan en la bibliografia. Una es *Ventanas de Parzen*[DH73] (en la cual centraremos nuestro estudio) y la otra *K-vecinos mas cercanos*. Ambas son muy similares, aunque presentan diferencias en algunas propiedades estadísticas[Fuk90].

2.1. Estimador de densidad

Se busca obtener una estimación de la densidad p en un punto $\mathbf{x}$. Dado un pequeño entorno Φ se busca la probabilidad de que un vector en $\mathbb{R}^n$ caiga

dentro de éste.

$$P = \int_{\Phi} p(\mathbf{u})\, d\mathbf{u} \tag{2.1}$$

Suponiendo $p(\mathbf{u})$ continua y que no sufre grandes alteraciones dentro de Φ se puede aproximar

$$\int_{\Phi} p(\mathbf{u})\, d\mathbf{u} \approx \widehat{p}(\mathbf{x}) \int_{\Phi} du = \widehat{p}(\mathbf{x})V \tag{2.2}$$

donde V representa el volumen de Φ y $\widehat{p}$ es una constante que aproxima p en $\mathbf{x}$.

Para obtener una estimación de P se utiliza una muestra con $\mathbf{x}_1$,...,$\mathbf{x}_n$ valores independientes idénticamente distribuidos (iid) y se considera cuántos de ellos caen dentro de Φ. Esta cantidad sigue una distribución binomial $Bi(n, p)$ cuya función de densidad esta dada por

$$f(i) = \binom{n}{i} P^i (1-P)^{n-i}. \tag{2.3}$$

Dado que la máxima probabilidad se alcanza en su valor medio $\mu = nP$, aproximamos P como $\frac{k}{n}$ donde k representa la cantidad de valores que caen en Φ.

En la Figura 2.1 (a) y (b), se muestran varias distribuciones binomiales unidimensionales para diferentes muestras de tamaño n. Se observa que a medida que aumenta el tamaño de la muestra, la dispersión disminuye y el máximo alcanzado en μ aumenta, acercando la estimación a su verdadero valor.

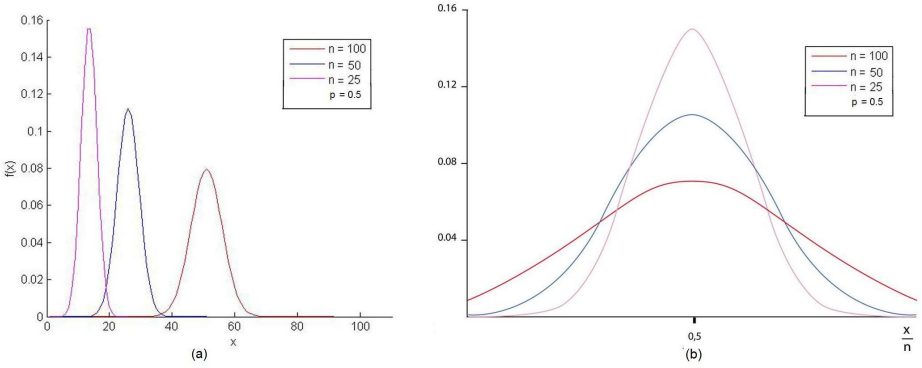

Figura 2.1: En (a) se muestran las distribuciones binomiales para diferentes valores de n y $p = 0{,}5$. El máximo valor alcanzado se produce en $\mu = nP$ para cada una de las densidades. Observar que a medida que n aumenta la varianza disminuye y la máxima probabilidad aumenta. En (b) el eje de abcisas está en función de $\frac{x}{n}$ para que esto pueda verse más claro.

Por lo tanto una buena estimación para $p(\mathbf{x})$ es:

$$\widehat{p}(\mathbf{x}) \approx \frac{k/n}{V} \tag{2.4}$$

2.2. Elección de parámetros

Algunas consideraciones en la elección de los parámetros que se utilizarán para estimar $p(\mathbf{x})$ deberán ser tenidas en cuenta. Por ejemplo, si se fija V y se hace crecer n infinitamente, k/n convergerá obteniendo el área promediada de $p(\mathbf{x})$ en la región Φ y lo que se busca es la estimación de la densidad en un punto. Por otro lado si n se mantiene constante y V se aproxima a cero, eventualmente no habrán valores en la región y $\widehat{p}(\mathbf{x})$ tenderá a cero lo cual tampoco sirve.

Se define un conjunto de regiones Φ_1,...,Φ_n que contienen a $\mathbf{x}$ donde la i-ésima región será usada con i valores, V_n es el volumen de Φ_n y k_n la cantidad de valores dentro de la región en cuestión. Además la n-esima estimación de $\widehat{p}(\mathbf{x})$ se define como:

$$\widehat{p}_n(\mathbf{x}) = \frac{k_n/n}{V_n} \tag{2.5}$$

Para garantizar que $\widehat{p}_n(\mathbf{x})$ converge a $p(\mathbf{x})$, se deben cumplir tres condiciones:

1. $\lim_{n\to\infty} V_n = 0$, en la medida en que el tamaño de la muestra aumente, V_n disminuirá
2. $\lim_{n\to\infty} k_n = \infty$, garantiza convergencia (en probabilidad) a P
3. $\lim_{n\to\infty} k_n/n = 0$, a pesar de contar con un gran número de puntos dentro del entorno V_n, k_n será despreciable en relación al tamaño de la muestra

2.3. Parzen Windows

Sea Φ_n un hipercubo d-dimensional, definimos h_n como la longitud de uno de los lados del hipercubo de manera tal que:

$$V_n = h_n^d \tag{2.6}$$

Por otro lado se define la *función ventana* d-dimensional como:

$$\alpha(\mathbf{u}) = \begin{cases} 1 & \text{si } |u_j| \leq 1/2 \text{ para j=1,...,d} \\ 0 & \text{caso contrario} \end{cases}$$

que determina aquellos valores dentro del hipercubo centrado en el origen de longitud uno. Definido esto, se puede deducir que la función $\alpha(\frac{\mathbf{x}-\mathbf{x}_i}{h_n})$

retornará uno en el caso de que $\mathbf{x}_i$ se encuentre en el hipercubo centrado en $\mathbf{x}$ con longitud de ventana de h_n y cero sino. El número de valores de la muestra $\mathbf{x}_1$,...,$\mathbf{x}_n$ que caen dentro del hipercubo de lado h_n centrado en $\mathbf{x}$ con tamaño de muestra igual a n está dado por:

$$k_n = \sum_{i=1}^{n} \alpha\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_n}\right) \tag{2.7}$$

La fórmula del estimador propuesta queda entonces definida como:

$$\widehat{p}_n(\mathbf{x}) = \frac{1}{n}\frac{1}{h_n^d}\sum_{i=1}^{n} \alpha\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_n}\right) \tag{2.8}$$

Notar que la función ventana cumple el rol de interpolador (ver Figura 2.2) ya que el estimador $p_n(\mathbf{x})$ resulta de la suma promediada de las funciones ventanas centradas en cada uno de los valores $\mathbf{x}_i$ de la muestra. La constante $V_n = h_n^d$ es el normalizador del interpolador ya que es igual a su integral; de este modo $\frac{1}{n}$ multiplica el resultado de la sumatoria haciendo que ésta sea igual a 1.

Se muestra esto en el caso unidimensional:

$$\int_{-\infty}^{+\infty} \widehat{p}_n(x)\,dx = \frac{1}{n}\frac{1}{h_n}\sum_{i=1}^{n}\int_{-\infty}^{+\infty} \alpha\left(\frac{x-x_i}{h_n}\right)\,dx = \frac{1}{n}\frac{1}{h_n}\sum_{i=1}^{n} h_n = \frac{1}{n}\sum_{i=1}^{n} 1 = \frac{1}{n}n = 1 \tag{2.9}$$

Existen varios interpoladores además del propuesto anteriormente, un ejemplo es la función ventana *Gaussiana* centrada en los valores $\mathbf{x}_i$ y con ancho σ. En el caso unidimesional el estimador usando éste interpolador sería:

$$\widehat{p}_n(x) = \frac{1}{n}\frac{1}{\sqrt{2\pi}\sigma}\sum_{i=1}^{n} \mathrm{e}^{-\frac{(x-x_i)^2}{2\sigma^2}} \tag{2.10}$$

donde la función ventana estaría dada por $\mathrm{e}^{-\frac{(x-x_i)^2}{2\sigma^2}}$ y $V_n = \sqrt{2\pi}\sigma$ dado que $\int_{-\infty}^{+\infty} \mathrm{e}^{-\frac{(x-x_i)^2}{2\sigma^2}}\,dx = \sqrt{2\pi}\sigma$.

2.4. Rol del ancho de ventana en la estimación

El valor de h_n afecta en la magnitud y ancho de la función ventana. Cuando h_n es grande, la ventana será ancha y baja. En este caso la superposición de funciones será grande resultando en una estimación suave (baja en resolución) sobre la muestra. Por otro lado, cuando h_n es pequeño, la influencia de cada una de las funciones ventana será de mayor peso al mismo

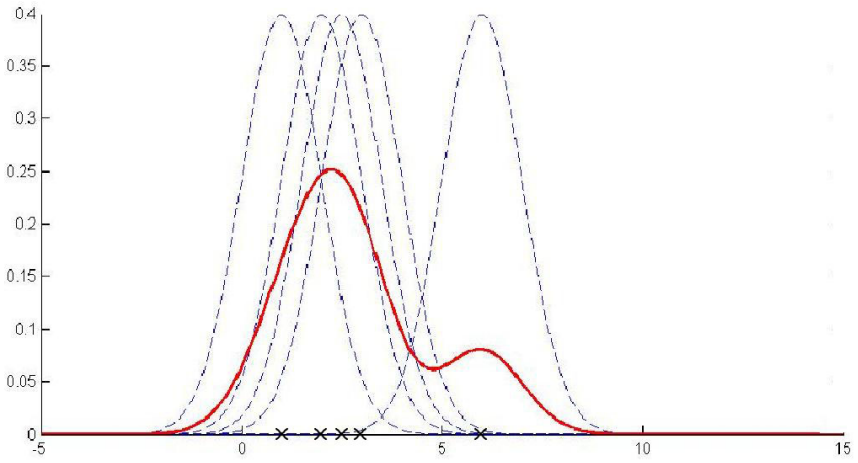

Figura 2.2: En este ejemplo unidimensional se muestra el $\widehat{p}_n(x)$ resultante de 5 funciones ventanas *gaussianas* centradas en 1, 2, 2.5, 3 y 6

tiempo que se verán afectadas por menos funciones dentro de cada ventana. El resultado será más sensible a pequeños cambios entre los valores de la muestra haciéndolo más influenciable a ruidos no deseados sobre el conjunto de valores.

En teoría si se cuenta con una cantidad infinita de valores n, V_n tendería a cero y $p_n(x)$ a la densidad desconocida. En la Figura 2.3 se muestran las densidades obtenidas cuando varian el ancho de ventana y el tamaño de la muestra.

2.5. Convergencia de la media y la varianza

Para garantizar convergencia se deben cumplir condiciones sobre la media y la varianza del estimador $\widehat{p}_n(\mathbf{x})$.

- $\lim_{n\to\infty} \overline{p}_n(\mathbf{x}) = p(\mathbf{x})$
- $\lim_{n\to\infty} \sigma_n^2(\mathbf{x}) = 0$

Calculamos primero la media de la estimación:

$$\overline{p}_n(\mathbf{x}) = E(\widehat{p}_n(\mathbf{x})) \tag{2.11}$$

$$= \frac{1}{n}\sum_{i=1}^{n} E\left(\frac{1}{V_n}\alpha\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_n}\right)\right)$$

$$= \frac{1}{n}\sum_{i=1}^{n} \frac{1}{V_n} E\left(\alpha\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_n}\right)\right) \tag{2.12}$$

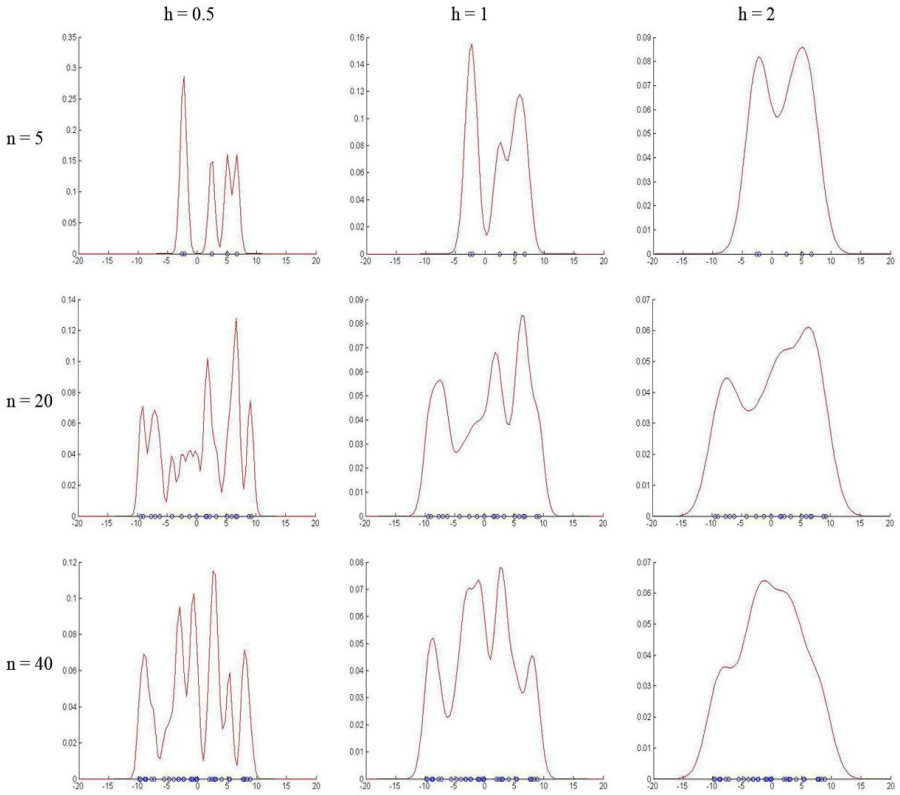

Figura 2.3: Ejemplo de la densidad estimada por el método de *Ventanas de Parzen* variando el tamaño de la muestra (n) y el ancho de ventana (h) para un conjunto de datos distribuidos de forma uniforme.

al ser $\mathbf{x}_i$ valores iid, puedo reemplazarlas por un $\mathbf{v}$ constante entonces:

$$= \frac{1}{n}\sum_{i=1}^{n}\frac{1}{V_n}E\left(\alpha\left(\frac{\mathbf{x}-\mathbf{v}}{h_n}\right)\right) \tag{2.13}$$

$$= \frac{1}{n}n\frac{1}{V_n}E\left(\alpha\left(\frac{\mathbf{x}-\mathbf{v}}{h_n}\right)\right) \tag{2.14}$$

$$= \int_{-\infty}^{+\infty}\frac{1}{V_n}\alpha\left(\frac{\mathbf{x}-\mathbf{v}}{h_n}\right)p(\mathbf{v})\,d\mathbf{v} \tag{2.15}$$

$$= \int_{-\infty}^{+\infty}\gamma_n(\mathbf{x}-\mathbf{v})p(\mathbf{v})\,d\mathbf{v} \tag{2.16}$$

donde γ es la función de ventana normalizada.

Como puede observarse la media resulta de la convolución entre la densidad desconocida y la función ventana. Puesto que cuando n tiende a infinito

$V_n \to 0$, la función ventana se aproximará a una función de *Delta de Dirac*. Debido a que $\int_{-\infty}^{+\infty} \delta(x-a)f(x)\,dx = f(a)$ se puede decir entonces que $\overline{p}_n(\mathbf{x})$ tiende a $p(\mathbf{x})$ desconocido.

En cuanto a la convergencia de la varianza se tiene que:

$$\sigma_n^2(\mathbf{x}) = \sum_{i=1}^{n} E\left[\left(\frac{1}{nV_n}\alpha\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_n}\right) - \frac{1}{n}\overline{p}_n(\mathbf{x})\right)^2\right] \tag{2.17}$$

$$= \sum_{i=1}^{n} E\left(\frac{1}{n^2V_n^2}\alpha^2\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_n}\right) - \frac{2}{n^2V_n}\alpha\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_n}\right)\overline{p}_n(\mathbf{x}) + \frac{1}{n^2}\overline{p}_n^2(\mathbf{x})\right) \tag{2.18}$$

$$= \sum_{i=1}^{n} E\left(\frac{1}{n^2V_n^2}\alpha^2\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_n}\right)\right) - \frac{2}{n^2V_n}\overline{p}_n(\mathbf{x})E\left(\alpha\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_n}\right)\right) + \frac{1}{n^2}\overline{p}_n^2(\mathbf{x}) \tag{2.19}$$

$$= \sum_{i=1}^{n} \frac{1}{n^2V_n^2}E\left(\alpha^2(\frac{\mathbf{x}-\mathbf{x}_i}{h_n})\right) - \frac{2}{n^2V_n}\overline{p}_n(\mathbf{x})\overline{p}_n(\mathbf{x})V_n + \frac{1}{n^2}\overline{p}_n^2(\mathbf{x}) \tag{2.20}$$

$$= \sum_{i=1}^{n} \frac{1}{n^2V_n^2}E\left(\alpha^2\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_n}\right)\right) - \frac{2}{n^2V_n}\overline{p}_n^2(\mathbf{x}) + \frac{1}{n^2}\overline{p}_n^2(\mathbf{x}) \tag{2.21}$$

$$= \sum_{i=1}^{n} \frac{1}{n^2V_n^2}E\left(\alpha^2\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_n}\right)\right) - \frac{1}{n^2}\overline{p}_n^2(\mathbf{x}) \tag{2.22}$$

al ser $\mathbf{x}_i$ valores iid, se pueden reemplazar por un $\mathbf{v}$ constante entonces:

$$= \sum_{i=1}^{n} \frac{1}{n^2V_n^2}E\left(\alpha^2\left(\frac{\mathbf{x}-\mathbf{v}}{h_n}\right)\right) - \frac{1}{n^2}\overline{p}_n^2(\mathbf{x}) \tag{2.23}$$

$$= \frac{1}{nV_n^2}E\left(\alpha^2\left(\frac{\mathbf{x}-\mathbf{v}}{h_n}\right)\right) - \frac{1}{n}\overline{p}_n^2(\mathbf{x}) \tag{2.24}$$

se puede acotar superiormente la varianza despreciando el sustraendo:

$$\sigma_n^2(\mathbf{x}) \leq \frac{1}{nV_n^2} E\left(\alpha^2\left(\frac{\mathbf{x}-\mathbf{v}}{h_n}\right)\right) \tag{2.25}$$

$$= \frac{1}{nV_n}\int \alpha^2\left(\frac{\mathbf{x}-\mathbf{v}}{h_n}\right) p(\mathbf{v})\, d\mathbf{v} \tag{2.26}$$

$$= \frac{1}{nV_n}\int \alpha\left(\frac{\mathbf{x}-\mathbf{v}}{h_n}\right)\alpha\left(\frac{\mathbf{x}-\mathbf{v}}{h_n}\right) p(\mathbf{v})\, d\mathbf{v} \tag{2.27}$$

$$\leq \frac{1}{nV_n}\int \alpha\left(\frac{\mathbf{x}-\mathbf{v}}{h_n}\right) sup(\alpha(.))p(\mathbf{v})\, d\mathbf{v} \tag{2.28}$$

$$= \frac{1}{nV_n} sup(\alpha(.))\int \alpha\left(\frac{\mathbf{x}-\mathbf{v}}{h_n}\right) p(\mathbf{v})\, d\mathbf{v} \tag{2.29}$$

$$= \frac{1}{nV_n} sup(\alpha(.))\overline{p}_n(\mathbf{x}) \tag{2.30}$$

Si se cumple que V_n se aproxima a cero a una velocidad menor que $\frac{1}{n}$, es decir que $nV_n \to \infty$, como el supremo de α esté acotado puedo asegurar que la varianza converge a cero.

2.6. Conclusiones

Desafortunadamente los resultados teóricos se alejan bastante de los prácticos puesto que para una adecuada elección de $\alpha(.)$ y ancho de V_n es necesario tener algún tipo de información acerca de $p(\mathbf{x})$ desconocido. En general las ventanas de tipo *Gaussianas* son más frecuentemente usadas debido a su robustez al ruido y a la ausencia de información en la distribución de los datos. Las técnicas no paramétricas tienen la ventaja de ser generales, fáciles de implementar y no es necesario suponer nada de la distribución. Por otro lado se necesita un tamaño grande de muestra (crecen exponencialmente en función de la dimensión del espacio de características) para asegurar la convergencia del método lo que provoca altos costos de computación. Este problema es conocido como la "*maldición de la dimensionalidad*".

Capítulo 3

Mean Shift

En el Capítulo anterior se presentó una técnica no paramétrica en donde se infiere una función de densidad estimada a partir de una muestra o conjunto de datos. En este Capítulo, a partir de esta función de densidad se aplicará el método de *Mean Shift* (o corrimiento hacia la media) para encontrar las modas. Éste, consta de un proceso iterativo de punto fijo el cual converge a un máximo local de la densidad obteniendo en cada paso una estimación del gradiente de dicha función de densidad a partir de la muestra.

Se dará además, una definición de *kernel* multivariado y sus características. Se probará que el proceso de *Mean Shift* converge con cierto tipos de *kernels* y la forma en que ésto se produce. Por último se enunciará diferentes criterios para seleccionar el parámetro *bandwidth* h de un *kernel*.

3.1. Introducción

Un *espacio de características* (*ec*) es un espacio d-dimensional en el que cada elemento de la muestra es representada como un punto. En la Figura 3.1 se muestra un conjunto de puntos en un *ec* tridimensional, dispuesto cada uno según las coordenadas correspondientes a los valores L, a y b asociados a cada uno de los píxeles de la imagen que se encuentra a la izquierda. De esta forma, la dimensión del espacio está determinado por el número de características utilizadas para representar cada punto. Una vez elegido el *ec* apropiado para un problema en particular, el análisis de los datos es una tarea independiente a la elección realizada.

El objetivo del análisis de una muestra en un *ec* es delinear los grupos representados como regiones densas de puntos. Los métodos que necesitan información sobre el número de clusters o la forma de éstos, en general no son aplicables en situaciones reales. Ésta es la razón por la que los métodos no paramétricos son estudiados para su uso.

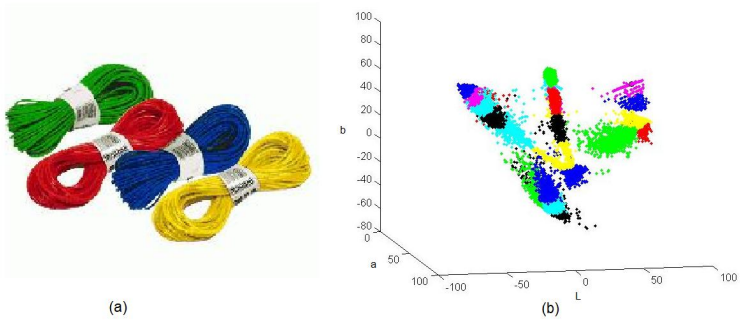

Figura 3.1: Representación de una imagen en $L^*a^*b^*$ en un espacio de características. En (a) mostramos la imagen a color original de 250x250 pixels. En (b) su correpondiente espacio de color $L^*a^*b^*$ de 62.500 data points.

Los datos se representarán (a partir de los pixeles de una imagen), estimando una función de densidad de probabilidad (ver Figura 3.2) parametrizada solamente por un *kernel* y su *bandwidth* los cuales serán definidos más adelante. Las regiones con gran cantidad de datos en el espacio de características corresponderán a los máximos locales de la función de densidad. Para detectar estas modas utilizamos la técnica de *Mean Shift*.

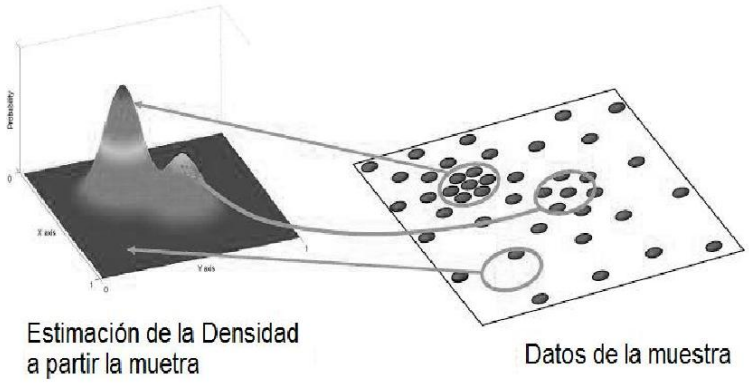

Figura 3.2: La muestra es representada por una función de densidad estimada donde los picos corresponden a regiones donde la conglomeración de datos es alta y las densidades bajas a regiones desérticas.

3.2. Análisis general de *Mean Shift*

Mean Shift es un proceso iterativo de punto fijo que converge a un máximo local, donde en cada iteración se estima el gradiente normalizado de la función de densidad en el punto correspondiente a cada paso. Aplicado varias veces desde diferentes puntos de partida, se pueden encontrar las modas de la

densidad; es decir las zonas en las que hay una mayor densidad de datos. El primer trabajo realizado referente a este tema fue desarrollado por Fukunaga y Hostetler en [FH75].

3.2.1. Estimación de la función de densidad

Para hallar la función de densidad se utiliza un método de estimación basado en *kernels* o mejor conocido en la literatura como *Ventanas de Parzen*. Sean $\mathbf{x}_1$,...,$\mathbf{x}_n$ valores en un espacio d-dimensional, el estimador de densidad multivariado queda definido como:

$$\widehat{f}(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} K_H(\mathbf{x}-\mathbf{x}_i) \text{ donde } K_H(\mathbf{x}) = |H|^{-1/2}K(H^{-1/2}\mathbf{x}) \tag{3.1}$$

donde $K(\mathbf{x})$ es un *kernel* y $|H| = h^2\mathrm{I}$ con *bandwidth* $h > 0$, la determinante de una matriz definida positiva y bandada de dimensión dxd. Se obtiene entonces, la expresión vista en la fórmula 2.8:

$$\widehat{f}(\mathbf{x}) = \frac{1}{nh^d}\sum_{i=1}^{n} K\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right) \tag{3.2}$$

El *kernel* K y el *bandwidth* h cumplen el rol de la función ventana y longitud de uno de sus lados respectivamente definidos en el Capítulo 2.

3.2.2. Kernel

Sea X un espacio euclídio d-dimensional y $\|x\|$ la norma 2 de $\mathbf{x}$. Una función $K : X \to \mathbb{R}$ se denomina *kernel* si existe un perfil $k : [0,\infty) \to \mathbb{R}$ tal que: $K(\mathbf{x}) = c_{k,d}k(\|\mathbf{x}\|^2)$ donde $c_{k,d}$ es una constante normalizadora y el perfil cumple que:

- k es no negativa
- k es no creciente
- k es continua de a trozos y $\int_0^\infty k(r)dr < \infty$

Los *kernels* multivariados que se van a utilizar son radialmente simétricos y caracterizables por su perfil. Notar que al rotar el perfil 360° respecto al origen no se determina la forma del *kernel*, la definición de perfil dada restringe los *kernels* a utilizar a una clase de *kernels* isotrópicos. Asimismo cumplen con las siguientes propiedades:

- $\int_{\mathbb{R}^d} K(\mathbf{x})dx = 1$ (normalizado)
- $\int_{\mathbb{R}^d} \mathbf{x}K(\mathbf{x})dx = 0$ (simétrico)
- $\int_{\mathbb{R}^d} \mathbf{x}\mathbf{x}^t K(\mathbf{x})dx = c_k I$ (no correlacionado)

- $\lim_{\|\mathbf{x}\|^d \to \infty} \|\mathbf{x}\|^d K(\mathbf{x}) = 0$ (decae exponencialmente)

Algunos de los *Kernels* más usados son:

Kernel de Epanechnikov

$$K_E(\mathbf{x}) = \begin{cases} \frac{1}{2} c_d^{-1}(d+2)(1-\|\mathbf{x}\|^2) & \text{si } \|\mathbf{x}\| \leq 1 \\ 0 & \text{caso contrario} \end{cases}$$

donde c_d es el volumen de la esfera definida de radio uno cuyo perfil es

$$k_E(x) = \begin{cases} 1-x & \text{si } 0 \leq x \leq 1 \\ 0 & \text{si } x > 1 \end{cases}$$

Kernel de Gaussiano

$$K_N(\mathbf{x}) = (2\pi)^{-d/2} exp(-\frac{1}{2}\|\mathbf{x}\|^2)$$

cuyo perfil es

$$k_N(x) = exp(-\frac{1}{2}x)$$

Kernel Uniforme

$$K_U(\mathbf{x}) = \begin{cases} 1 & \text{si a } 0 \leq \|\mathbf{x}\| \leq 1 \\ 0 & \text{caso contrario} \end{cases}$$

cuyo perfil es

$$k_U(x) = \begin{cases} 1 & \text{si } 0 \leq x \leq 1 \\ 0 & \text{si } x > 1 \end{cases}$$

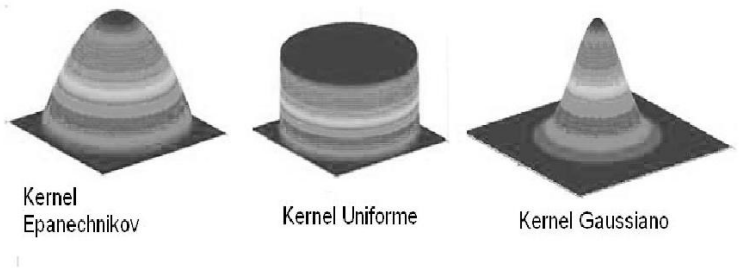

Figura 3.3: *Kernel* de *Epanechnicov*, *Gaussiano* y *Uniforme*

3.2.3. Estimación del gradiente de densidad

Se denomina *Mean Shift* o corrimiento hacia la media, al desplazamiento desde un punto inicial $\mathbf{x}$ en el espacio a otro que resulta del promedio de los pesos de los datos s dentro de una vecindad determinada por la región S centrado en $\mathbf{x}$. Los pesos asociados a los datos dentro del hipercubo S están directamente relacionados con sus distancias respecto al origen y sus valores quedan determinados por la función *kernel* K.

$$m_K(\mathbf{x}) = \frac{\sum_{s \in S}^{n} K(s - \mathbf{x})s}{\sum_{s \in S}^{n} K(s - \mathbf{x})} - \mathbf{x} \tag{3.3}$$

El numerador del cociente del lado derecho de la ecuación es la suma ponderada de cada uno de los datos s dentro del entorno S centrado en $\mathbf{x}$, mientras que el denominador promedia la sumatoria. El vector desplazamiento $m_K(\mathbf{x})$ queda entonces definido como la diferencia entre el punto de mayor densidad de datos y el inicial $\mathbf{x}$. Para que esto pueda entenderse de mejor manera ver la Figura 3.4.

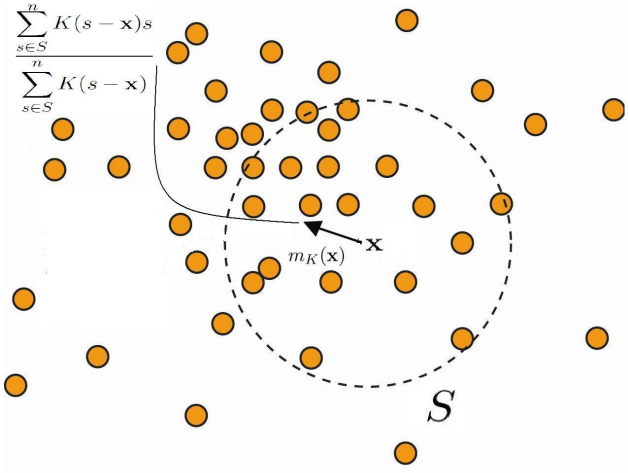

Figura 3.4: Una iteración de *Mean Shift*.

Si se estima el gradiente de ésta función de densidad, entonces:

$$\widehat{\nabla} f_{h,K}(\mathbf{x}) = \nabla \widehat{f}_{h,K}(\mathbf{x}) = \frac{c_{k,d}}{nh^d} \sum_{i=1}^{n} k'\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right) \tag{3.4}$$

$$= \frac{2c_{k,d}}{nh^{d+2}} \sum_{i=1}^{n} (\mathbf{x}-\mathbf{x}_i) k'\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right) \tag{3.5}$$

se define un nuevo perfil como $g(\mathbf{x}) = -k'(\mathbf{x})$ suponiendo que k es derivable en $[0,\infty)$ salvo en un conjunto finito de puntos.

Notar que g cumple con la condición de perfil de un *kernel* dado que,

1. En primer término, k es decreciente por ende k' es negativa y g es positiva.
2. Segundo, se supone k convexa, entonces k'' es positiva, g' es negativa y entonces g no creciente.
3. Tercero, se sabe que k es C^1 salvo finitos puntos entonces k' es continua de a tozos y entonces g es continua de a trozos.
4. Por último, $\int_0^\infty g(r)\,dr < \infty \Leftrightarrow \int_0^\infty -k(r)\,dr < \infty \Leftrightarrow k(0) - k(\infty) < \infty$. Por un lado $k(0) < \infty$ por k estar definida en cero y por otro $k(\infty) > k(0)$ por ser decreciento y $k(\infty) > 0$ por ser positiva, entonces la integral converge.

Continuando lo obtenido en 3.5:

$$= \frac{2c_{k,d}}{nh^{d+2}} \sum_{i=1}^{n} (\mathbf{x}_i - \mathbf{x}) g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right) \tag{3.6}$$

$$= \frac{2c_{k,d}}{nh^{d+2}} \sum_{i=1}^{n} \left[\mathbf{x}_i g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right) - \mathbf{x} g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)\right] \frac{\sum_{i=1}^n g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^n g(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2)}$$

$$= \frac{2c_{k,d}}{nh^{d+2}} \sum_{i=1}^{n} \left[\frac{\mathbf{x}_i g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^n g(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2)} - \frac{\mathbf{x} g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^n g(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2)}\right] \sum_{i=1}^{n} g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)$$

$$= \frac{2c_{k,d}}{nh^{d+2}} \left[\frac{\sum_{i=1}^n \mathbf{x}_i g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^n g(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2)} - \frac{\mathbf{x}\sum_{i=1}^n g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^n g(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2)}\right] \sum_{i=1}^{n} g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)$$

$$= \frac{2c_{k,d}}{nh^{d+2}} \left[\sum_{i=1}^{n} g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)\right] \left[\frac{\sum_{i=1}^n \mathbf{x}_i g(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2)}{\sum_{i=1}^n g(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2)} - \mathbf{x}\right] \tag{3.7}$$

De la ecuación 3.7 se discrimina por un lado la función de densidad estimada con el *kernel* G y por otro el vector *Mean Shift* con *kernel* G

$$\widehat{f}_{h,G}(\mathbf{x}) = \frac{c_{g,d}}{nh^d} \sum_{i=1}^{n} g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right) \tag{3.8}$$

$$m_{h,G} = \left[\frac{\sum_{i=1}^{n} \mathbf{x}_i g(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2)}{\sum_{i=1}^{n} g(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2)} - \mathbf{x}\right] \tag{3.9}$$

de 3.7, 3.8 y 3.9 reescribiendo $m_{h,G}(\mathbf{x})$ se obtiene que

$$m_{h,G} = \frac{1}{2}h^2 \frac{\widehat{\nabla} f_{h,K}(\mathbf{x})}{\widehat{f}_{h,G}(\mathbf{x})} \tag{3.10}$$

Puede observarse que el vector de *Mean Shift* queda definido como una estimación normalizada del gradiente de la función de densidad de la muestra. La normalización que está dada por la función de densidad con el *kernel* G, hace que en regiones en donde la densidad es baja se realicen pasos largos y en lugares próximos a la cumbre los pasos sean cada vez más pequeños.

El *Mean Shift* $m_{h,G}$ con *kernel* G representa el paso respecto a un punto de la densidad estimada con *kernel* K hacia la cima. En este caso se dice que el *kernel* K es la sombra (o en inglés *shadow*) de G. Se puede demostrar que el *kernel* de *Epanechnikov* es la sombra del *kernel* Uniforme, mientras que el *kernel* Gaussiano tiene a sí mismo como su propia sombra. Una definición más específica de *shadow kernel* y sus propiedades se encuentra en el artículo de Yizong Cheng [Che95].

El procedimiento de *Mean Shift* (ver Figura 3.5) funciona de la siguiente manera:

1. A partir de un punto $\mathbf{x}$ en el espacio de caracteríticas generar el vector de desplazamiento hacia la media $m_{h,G}(\mathbf{x})$ a partir de los datos contenidos en la región S
2. Una vez obtenido el nuevo punto, correr el centro de la ventana a éste
3. En caso de que el desplazamiento sea menor a un epsilon dado terminar, sino volver a repetir el procedimiento

De esta forma se puede formular la siguiente iteración de punto fijo que resume los puntos anteriores:

$$\mathbf{y}_{j+1} = \mathbf{y}_j + m_{h,G}(\mathbf{y}_j)$$

Existen situaciones en donde el método propuesto no alcanza un máximo local. Por ejemplo en el caso de encontrar una meseta o punto de ensilladura convergerá, pero no a lo esperado. *Mean Shift* alcanzará un máximo en tanto se tenga garantía de que en cada iteración exista un único punto de mayor densidad de datos dentro del entorno determinado por el ancho del *kernel* utilizado.

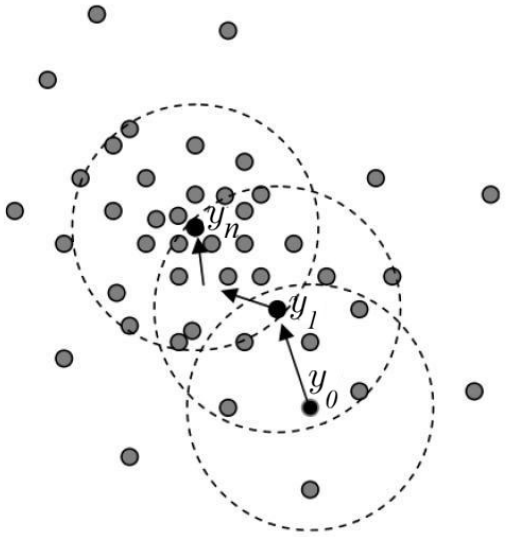

Figura 3.5: Como primer paso se computa el vector *Mean Shift* $m_{h,G}$, luego se translada el *kernel* (ventana) al nuevo centro $m_{h,G}$. Por último se repite los 2 primeros pasos hasta que el corrimiento sea menor a una cota.

3.3. Criterio de convergencia de *Mean Shift*

Se demostrará que el método de *Mean Shift* utilizando un *kernel* de *Epanechnikov* para estimar la función de densidad siempre converge. Además se enunciarán las condiciones mínimas y suficientes que deben cumplir los *kernels* en general para garantizar convergencia.

3.3.1. Convergencia utilizando un *Kernel Epanechnicov*

Sea $\{\mathbf{y}_j\}_{j=1,2...}$ la secuencia de puntos resultados de las iteraciones de *Mean Shift*, es decir:

$$\mathbf{y}_{j+1} = \frac{1}{n_j} \sum_{\mathbf{x}_i \in S_h(\mathbf{y}_j)} \mathbf{x}_i \tag{3.11}$$

donde n_j es el número de puntos $\mathbf{x}_i$ que caen dentro del espacio delimitado por la hiperesfera $S_h(\mathbf{y}_j)$ centrada en $\mathbf{y}_j$ de ancho h.

Teorema. Sea $\widehat{f}_E = \{\widehat{f}_E(j)\}_{j=1,2,...} = \{\widehat{f}_E(\mathbf{y}_j)\}_{j=1,2,...}$ la secuencia de densidades obtenidas utilizando el *kernel* de *Epanechnikov* en los puntos $\{\mathbf{y}_j\}_{j=1,2,...}$ definido como la secuencia de puntos del procedimiento de *Mean Shift* con un *kernel* Uniforme. Entonces la secuencia de densidades es convergente.

Demostración. Dado que la cantidad de valores $\mathbf{x}_i$ de la muestra tiene una cardinalidad n, la secuencia de densidades $\widehat{f}_E$ está acotada superiormente. Si probamos entonces que $\widehat{f}_E$ es estricta monótonamente creciente, la secuencia

es convergente.

Se define n_j, n'_j y n''_j como $n_j = n'_j + n''_j$ el número de puntos que caen dentro de la ventana d-dimensional $S_h(\mathbf{y}_j), S'_h(\mathbf{y}_j) = S_h(\mathbf{y}_j) - S''_h(\mathbf{y}_j)$ y $S''_h(\mathbf{y}_j) = S_h(\mathbf{y}_j) \cap S_h(\mathbf{y}_{j+1})$ como se muestra en la Figura 3.6.

Se define como el origen, a $\mathbf{y}_j$ con lo que podemos notar $\left\|\mathbf{y}_j - \mathbf{x}_i\right\|^2 = \left\|\mathbf{x}_i\right\|^2$

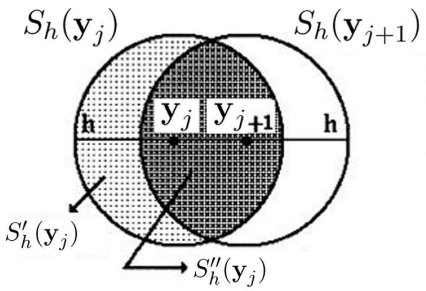

Figura 3.6: Ventanas d-dimensionales utilizadas en dos iteraciones sucesivas por *Mean Shift*.

$$\widehat{f}_E(j) = \widehat{f}_E(\mathbf{y}_j) = \frac{1}{nh^d} \sum_{\mathbf{x}_i \in S_h(\mathbf{y}_j)} K_E\left(\frac{\mathbf{y}_j - \mathbf{x}_i}{h}\right) = \frac{d+2}{2n(h^d c_d)} \sum_{\mathbf{x}_i \in S_h(\mathbf{y}_j)} \left(1 - \frac{\left\|\mathbf{x}_i\right\|^2}{h^2}\right)$$

puesto que K_E es no negativo y $S''_h(\mathbf{y}_j) \subset S_h(\mathbf{y}_{j+1})$ podemos afirmar que

$$\widehat{f}_E(j+1) = \widehat{f}_E(\mathbf{y}_{j+1}) = \frac{d+2}{2n(h^d c_d)} \sum_{\mathbf{x}_i \in S_h(\mathbf{y}_{j+1})} \left(1 - \frac{\left\|\mathbf{y}_{j+1} - \mathbf{x}_i\right\|^2}{h^2}\right) \quad (3.12)$$

$$\geq \frac{d+2}{2n(h^d c_d)} \sum_{\mathbf{x}_i \in S''_h(\mathbf{y}_j)} \left(1 - \frac{\left\|\mathbf{y}_{j+1} - \mathbf{x}_i\right\|^2}{h^2}\right) \quad (3.13)$$

Sabiendo que $n'_j = n_j - n''_j$

$$\widehat{f}_E(j+1) - \widehat{f}_E(j) \tag{3.14}$$

$$\geq \frac{d+2}{2n(h^d c_d)} \left[\sum_{\mathbf{x}_i \in S''_h(\mathbf{y}_j)} (1 - \frac{\left\|\mathbf{y}_{j+1} - \mathbf{x}_i\right\|^2}{h^2}) - \sum_{\mathbf{x}_i \in S_h(\mathbf{y}_j)} (1 - \frac{\|\mathbf{x}_i\|^2}{h^2}) \right] \tag{3.15}$$

$$= \frac{d+2}{2n(h^d c_d)} \left[- \sum_{\mathbf{x}_i \in S''_h(\mathbf{y}_j)} \frac{\left\|\mathbf{y}_{j+1} - \mathbf{x}_i\right\|^2}{h^2} + \sum_{\mathbf{x}_i \in S_h(\mathbf{y}_j)} \frac{\|\mathbf{x}_i\|^2}{h^2} + n''_j - n_j \right] \tag{3.16}$$

$$= \frac{d+2}{2n(h^{d+2} c_d)} \left[\sum_{\mathbf{x}_i \in S_h(\mathbf{y}_j)} \|\mathbf{x}_i\|^2 - \sum_{\mathbf{x}_i \in S''_h(\mathbf{y}_j)} \left\|\mathbf{y}_{j+1} - \mathbf{x}_i\right\|^2 - n'_j h^2 \right] \tag{3.17}$$

por definición sabemos que $\left\|\mathbf{y}_{j+1} - \mathbf{x}_i\right\| \geq h^2$ para todo $\mathbf{x}_i \in S'_h(\mathbf{y}_j)$ y por tanto vale que $\sum_{\mathbf{x}_i \in S'_h(\mathbf{y}_j)} \left\|\mathbf{y}_{j+1} - \mathbf{x}_i\right\|^2 \geq n'_j h^2$, entonces

$$\geq \frac{d+2}{2n(h^{d+2} c_d)} \left[\sum_{\mathbf{x}_i \in S_h(y_j)} \|\mathbf{x}_i\|^2 - \sum_{\mathbf{x}_i \in S''_h(\mathbf{y}_j)} \left\|\mathbf{y}_{j+1} - \mathbf{x}_i\right\|^2 - \sum_{\mathbf{x}_i \in S'_h(\mathbf{y}_j)} \left\|\mathbf{y}_{j+1} - \mathbf{x}_i\right\|^2 \right] \tag{3.18}$$

$$= \frac{d+2}{2n(h^{d+2} c_d)} \left[\sum_{\mathbf{x}_i \in S_h(\mathbf{y}_j)} \|\mathbf{x}_i\|^2 - \sum_{\mathbf{x}_i \in S_h(\mathbf{y}_j)} \left\|\mathbf{y}_{j+1} - \mathbf{x}_i\right\|^2 \right] \tag{3.19}$$

$$= \frac{d+2}{2n(h^{d+2} c_d)} \left[\sum_{\mathbf{x}_i \in S_h(\mathbf{y}_j)} \|\mathbf{x}_i\|^2 - \sum_{\mathbf{x}_i \in S_h(\mathbf{y}_j)} \|\mathbf{x}_i\|^2 + \left\|\mathbf{y}_{j+1}\right\|^2 - 2\mathbf{y}_{j+1}^t \mathbf{x}_i \right] \tag{3.20}$$

$$= \frac{d+2}{2n(h^{d+2} c_d)} \left[\sum_{\mathbf{x}_i \in S_h(\mathbf{y}_j)} 2\mathbf{y}_{j+1}^t \mathbf{x}_i - \sum_{\mathbf{x}_i \in S_h(\mathbf{y}_j)} \left\|\mathbf{y}_{j+1}\right\|^2 \right] \tag{3.21}$$

$$= \frac{d+2}{2n(h^{d+2} c_d)} \left[2\mathbf{y}_{j+1}^t \sum_{\mathbf{x}_i \in S_h(\mathbf{y}_j)} \mathbf{x}_i - n_j \left\|\mathbf{y}_{j+1}\right\|^2 \right] \tag{3.22}$$

puesto que $\mathbf{y}_{j+1} = \frac{1}{n_j} \sum_{\mathbf{x}_i \in S_h(\mathbf{y}_j)} \mathbf{x}_i$, vale $n_j \mathbf{y}_{j+1} = \sum_{\mathbf{x}_i \in S_h(\mathbf{y}_j)} \mathbf{x}_i$; reemplazando ésto en la expresión anterior

$$= \frac{d+2}{2n(h^{d+2}c_d)} \left[2n_j \left\| \mathbf{y}_{j+1} \right\|^2 - n_j \left\| \mathbf{y}_{j+1} \right\|^2 \right] \tag{3.23}$$

$$= \frac{d+2}{2n(h^{d+2}c_d)} n_j \left\| \mathbf{y}_{j+1} \right\|^2 \tag{3.24}$$

Esta última expresión es estrictamente positiva salvo cuando $\mathbf{y}_j = \mathbf{y}_{j+1}$ con lo cual he llegado al límite de $\widehat{f}_E$. Dado que $\widehat{f}_E$ es acotada y monótonamente estricta creciente queda probado que la secuencia de densidades en los puntos $\mathbf{y}_j$ es convergente.

3.3.2. Condición suficiente de convergencia

A continuación se enuncia una condición suficiente de convergencia todo tipo de *kernels*.

Teorema. Para todo *kernel* K convexo con perfil monótonamente decreciente. Sea $\widehat{f}_K = \{\widehat{f}_K(j)\}_{j=1,2,...} = \{\widehat{f}_K(\mathbf{y}_j)\}_{j=1,2,...}$ la secuencia de densidades obtenidas utilizando el *kernel* K en los puntos $\{\mathbf{y}_j\}_{j=1,2,...}$ definido como la secuencia de puntos del procedimiento de *Mean Shift*. Entonces la secuencia de densidades es convergente.

Demostración. Sabemos que la secuencia de densidades estimadas se encuentra acotada superiormente puesto que el conjunto de muestras es finito. Sólo falta entonces mostrar que $\widehat{f}_K$ es estricta monótona creciente.

$$\widehat{f}_K(j+1) - \widehat{f}_K(j) = \frac{1}{nh^d} \sum_{i=1}^{n} \left[k\left(\left\| \frac{\mathbf{y}_{j+1} - \mathbf{x}_i}{h} \right\|^2 \right) - k\left(\left\| \frac{x_i}{h} \right\|^2 \right) \right] \tag{3.25}$$

Puesto que se trata de un *kernel* convexo se cumple que $k(x_2) \geq k(x_1) + k'(x_1)(x_2 - x_1)$ para todo $x_1, x_2 \in [0, +\infty]$, sea $-k' = g$ entonces $k(x_2) - k(x_1) \geq g(x_1)(x_1 - x_2)$. Reemplazando llegamos a que:

$$\widehat{f}_K(j+1) - \widehat{f}_K(j) \geq \frac{1}{nh^{d+2}} \sum_{i=1}^{n} g\left(\left\|\frac{x_i}{h}\right\|^2\right)\left[\|x_i\|^2 - \|y_{j+1} - x_i\|^2\right] \quad (3.26)$$

$$= \frac{1}{nh^{d+2}} \sum_{i=1}^{n} g\left(\left\|\frac{\mathbf{x}_i}{h}\right\|^2\right)\left[\|\mathbf{x}_i\|^2 - \|\mathbf{x}_i\|^2 - \left\|\mathbf{y}_{j+1}^t\right\|^2 + 2\mathbf{y}_{j+1}\mathbf{x}_i\right] \quad (3.27)$$

$$= \frac{1}{nh^{d+2}} \sum_{i=1}^{n} g\left(\left\|\frac{\mathbf{x}_i}{h}\right\|^2\right)\left[2\mathbf{y}_{j+1}^t\mathbf{x}_i - \left\|\mathbf{y}_{j+1}\right\|^2\right] \quad (3.28)$$

$$= \frac{1}{nh^{d+2}}\left[2\mathbf{y}_{j+1}^t \sum_{i=1}^{n} \mathbf{x}_i g\left(\left\|\frac{\mathbf{x}_i}{h}\right\|^2\right) - \left\|\mathbf{y}_{j+1}\right\|^2 \sum_{i=1}^{n} g\left(\left\|\frac{\mathbf{x}_i}{h}\right\|^2\right)\right] \quad (3.29)$$

$$= \frac{1}{nh^{d+2}}\left[2\mathbf{y}_{j+1}^t\mathbf{y}_{j+1} \sum_{i=1}^{n} g\left(\left\|\frac{\mathbf{x}_i}{h}\right\|^2\right) - \left\|\mathbf{y}_{j+1}\right\|^2 \sum_{i=1}^{n} g\left(\left\|\frac{\mathbf{x}_i}{h}\right\|^2\right)\right] \quad (3.30)$$

$$= \frac{1}{nh^{d+2}} \left\|\mathbf{y}_{j+1}\right\|^2 \sum_{i=1}^{n} g\left(\left\|\frac{\mathbf{x}_i}{h}\right\|^2\right) \quad (3.31)$$

Puesto que k es monótonamente decreciente $-k' = g$ será positiva y por tando $\sum_{i=1}^{n} g(\left\|\frac{\mathbf{x}_i}{h}\right\|^2)$ lo será. La expresión obtenida es entonces positiva en tanto $\mathbf{y}_j \neq \mathbf{y}_{j+1}$ probando entonces la convergencia de la secuencia de densidades a la moda.

Para probar la convergencia en la secuencia $\{\mathbf{y}_j\}_{j=1\ldots n}$ reescribimos 3.31 sin suponer $\mathbf{y}_j = 0$,

$$\widehat{f}_K(j+1) - \widehat{f}_K(j) \geq \frac{1}{nh^{d+2}} \left\|\mathbf{y}_{j+1} - \mathbf{y}_j\right\|^2 \sum_{i=1}^{n} g\left(\left\|\frac{\mathbf{y}_j - \mathbf{x}_i}{h}\right\|^2\right) \quad (3.32)$$

Sabemos que ésta desigualdad se cumple para $j, j+1$ términos consecutivos, es decir:

$$\widehat{f}_K(j+1) - \widehat{f}_K(j) \geq \left\|\mathbf{y}_{j+1} - \mathbf{y}_j\right\|^2 M$$
$$\widehat{f}_K(j+2) - \widehat{f}_K(j+1) \geq \left\|\mathbf{y}_{j+2} - \mathbf{y}_{j+1}\right\|^2 M$$
$$\vdots$$
$$\widehat{f}_K(j+m) - \widehat{f}_K(j+m-1) \geq \left\|\mathbf{y}_{j+m} - \mathbf{y}_{j+m-1}\right\|^2 M$$

donde M representa el mínimo valor de $\frac{1}{nh^{d+2}} \sum_{i=1}^{n} g\left(\left\|\frac{\mathbf{y}_j - \mathbf{x}_j}{h}\right\|^2\right)$. Si sumamos los términos de la izquierda y de la derecha respectivamente llegamos a la

expresión:

$$\widehat{f}_K(j+m) - \widehat{f}_K(j) \geq \left\|\mathbf{y}_{j+m} - \mathbf{y}_j\right\|^2 M$$

Puesto que acabamos de probar que $\{\widehat{f}_K(j)\}$ es convergente entonces $\{y_j\}_{j=1,2,...}$ es una sucesión de Cauchy y por lo tanto converge.

Las propiedades de convergencia son válidas si asociamos a cada punto $\mathbf{x}_i$ de la muestra un peso w_i y no se han incluido en la demostración para simplificar la notación. Es importante tener en cuenta ésto último para garantizar la correctitud del algoritmo de seguimiento que presentaremos en el Capítulo 6, el cual se lleva a cabo a partir del método de *Mean Shift*.

3.4. Propiedad de una trayectoria suave utilizando un *kernel Normal*

Si se usa un *kernel* normal, la trayectoria que sigue el método de *Mean Shift* es una trayectoria suave, esto significa que el ángulo entre dos vectores *Mean Shift* siempre es menor que 90 grados. La demostración de esta propiedad puede obtenerse de manera inmediata al utilizar el siguiente lema.

Lema. La suma ponderada de las proyecciones de $(\mathbf{y}_{j+1} - \mathbf{x}_i)$ en la dirección $(0, \mathbf{y}_{j+1})$ es negativa cuando los pesos están dados por un *kernel* normal centrado en $\mathbf{y}_{j+1}$, es decir que

$$\sum_{i=1}^{n}(\|\mathbf{y}_{j+1}\|^2 - \mathbf{y}_{j+1}^t\mathbf{x}_i)exp\left(-\left\|\frac{\mathbf{y}_{j+1} - \mathbf{x}_i}{h}\right\|^2\right) \leq 0 \tag{3.33}$$

La demostración del Lema se encuentra en el apéndice del trabajo realizado por Comaniciu y Meers [CMM02]. A partir de esto llegamos a la siguiente desigualdad

$$\|\mathbf{y}_{j+1}\|^2 \leq \mathbf{y}_{j+1}^T \frac{\sum_{i=1}^{n}\mathbf{x}_i exp\left(-\left\|\frac{\mathbf{y}_{j+1}-\mathbf{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} exp\left(-\left\|\frac{\mathbf{y}_{j+1}-\mathbf{x}_i}{h}\right\|^2\right)} = \mathbf{y}_{j+1}^t\mathbf{y}_{j+2} \tag{3.34}$$

Sean $\mathbf{v}$ y $\mathbf{w}$ dos vectores, sabemos que $\frac{\mathbf{vw}}{\|\mathbf{v}\|\|\mathbf{x}\|} = \cos\theta$ y por lo tanto $\theta < 90 \Leftrightarrow \cos\theta > 0$, es decir que la distancia angular entre v y w es menor a 90° si $vw > 0$.

En nuestro caso v y w son los vectores $\mathbf{y}_{j+1}$ y aquel que va de $\mathbf{y}_{j+1}$ e $\mathbf{y}_{j+2}$ es decir $(\mathbf{y}_{j+2} - \mathbf{y}_{j+1})$

$$\begin{aligned} \mathbf{y}_{j+1}^t(\mathbf{y}_{j+2} - \mathbf{y}_{j+1}) &\geq 0 \\ \mathbf{y}_{j+1}^t\mathbf{y}_{j+2} - \mathbf{y}_{j+1}^t\mathbf{y}_{j+1} &\geq 0 \\ \mathbf{y}_{j+1}^t\mathbf{y}_{j+2} &\geq \mathbf{y}_{j+1}^t\mathbf{y}_{j+1} \\ \mathbf{y}_{j+1}\mathbf{y}_{j+2} &\geq \|\mathbf{y}_{j+1}\|^2 \end{aligned} \tag{3.35}$$

Esto último es a lo que llegamos como conclusión del Lema en 3.34.

En las aplicaciones se ha preferido usar un *kernel* uniforme ya que al hacerlo obtengo una convergencia en finitos pasos y más veloz que si se utiliza un *kernel* normal, aunque con éste último se obtienen mejores resultados.

3.5. Elección del *bandwidth*

Como se mencionó en el Capítulo 2 en la Sección 2.4 el tamaño del *bandwidth* influye en la manera en que cada uno de los datos de la muestra influye en la *pdf* resultante. Existen cuatro diferentes técnicas enumeradas en [CMM02] para la selección del parámetro *bandwidth* h utilizado en la función *kernel* para el procedimiento *Mean Shift* :

- Selección estadística: Es el *bandwidth* que alcanza el mejor compromiso entre el sesgo y la varianza del estimador sobre todos los $\mathbf{x} \in \mathbb{R}^d$.(aquel que minimiza la medida AMISE, diferencia cuadrática entre lo estimado y lo real)
- Por estabilidad de la descomposición: El *bandwidth* es la media del rango sobre el cual se obtuvo el mismo número de clusters para diferentes experimentos.
- *Bandwidth* máximo: Es el mejor *bandwidth* que maximiza una función objetivo que expresa la calidad de descomposición. La función objetivo estudia por lo general la variabilidad inter-intra clusters o conectividad-aislamiento entre ellos.
- Información de alto nivel: Como en muchos de los casos la descomposición depende de la tarea a realizar, se puede tomar el *bandwidth* de la información dada por el usuario o por niveles altos de la tarea.

Una inapropiada elección del *bandwidth* puede causar la unión de modas significativas en un contexto en particular o generar otras no representativas. La elección de un tamaño de ancho de *kernel* correcto es una tarea no trivial. En [CRM01] se detalla un análisis detallado para la elección del *bandwidth* de forma dinámica donde se presentan dos soluciones posibles, una paramétrica

y otra semi paramétrica. Se muestra cómo puede maximizarse la magnitud del vector de *Mean Shift* obteniendo resultados superiores a utilizar procedimientos con anchos de ventana fijos.

Capítulo 4

Agrupamiento

El agrupamiento o *clustering* es un procedimento de agrupación de datos d-dimensionales de una muestra de acuerdo a un criterio de cercanía el cual está determinado (en este trabajo) por la distancia euclidiana. No obstante, existen otras funciones de distancia o similaridad más robustas que pueden ser utilizadas. Los miembros pertenecientes a un mismo grupo o *cluster* comparten ciertas características y estos pueden ser representados cada uno por un punto característico.

Los algoritmos de agrupamiento tienen varias aplicaciones en la vida real como ser en la minería de datos, teoría de las comunicaciones, medicina, imágenes de satélite, entre otras.

Los métodos de agrupamiento se dividen en tres grupos: jerárquicos, particionales y basados en densidad.

- *Algoritmos jerárquicos*: son aquellos en los que se va particionando el conjunto de datos por niveles, de modo tal que en cada nivel, generalmente se unen o se dividen dos grupos del nivel anterior (según si es un algoritmo aglomerativo o divisivo).
- *Algoritmos particionales*: son los que realizan una división inicial de los datos en grupos y luego mueven los objetos de un grupo a otro según se optimice alguna función objetivo.
- *Algoritmos basados en densidad*: enfocan el problema teniendo en cuenta la distribución de densidad de los puntos, de modo tal que los grandes grupos de puntos se corresponden con altas densidades y las regiones entre *clusters* se corresponden con bajas densidades.

Dependiendo de las consideraciones que se hagan sobre la muestra se distinguen dos aproximaciones, la paramétrica y no paramétrica. En el primer caso, se supone un conocimiento a priori sobre la forma funcional de las distribuciones de probabilidad de cada clase en el *espacio de características* (ec),

las cuales vendrán determinadas por un conjunto finito y normalmente fijo de parámetros. La aproximación no paramétrica no supone ninguna forma de las distribuciones de los datos, de modo que el único conocimiento a priori será el correspondiente a la información inducida a partir de la muestra.

El *ec* debe ser eculídio, es decir que cada par de puntos distan en el *ec* de manera símilar respecto la percepción del ojo humano. $L^*a^*b^*$ y $L^*u^*v^*$ son dos espacios de color que cumplen con lo anterior. En este trabajo utilizaremos algoritmos no paramétricos basados en densidad en un espacio L^*a^*b.

Vamos a utilizar un algoritmo de *clustering* propuesto en [CM98] el cual está basado en la distribución de densidad utilizando el método *Mean Shift* que se complementará con un método supervisado/paramétrico por cuestiones de rendimiento en el tiempo de ejecución. Este algoritmo es el de *k-vecinos más cercanos* el cual tiene un orden logarítmico y es el que presentamos a continuación.

4.1. Algoritmo k-vecinos más cercanos

K-vecinos más cercanos, *k-nearest neighbours* o simplemente *Knn* es un método de clasificación supervisada que calcula la probabilidad a posteriori de que un nuevo vector $\mathbf{x}$ pertenezca a una clase C_j considerando los k vecinos más cercanos a $\mathbf{x}$.

Los ejemplos de entrenamiento son puntos d-dimensionales $\mathbf{x} = (x_1, ..., x_d) \in X$ que tienen asociados una clase C_j donde X es el espacio dividido en regiones etiquetadas en clases.

Un nuevo punto es asignado a la clase C_j, si ésta es la clase más frecuente a la que pertenecen sus k vecinos más cercanos. La vecindad se determina por lo general utilizando la distancia euclideana. Para clarificar el algoritmo ver la Figura 4.1. El algoritmo *Knn* tiene dos fases, una de entrenamiento y otra de clasificación. La primera fase consta de almacenar y rotular los ejemplos de entrenamiento mientras que la segunda de etiquetar una clase a un punto (sin clasificar) teniendo en cuenta la clase que predomina entre los los k ejemplos más cercanos a éste. Más formalmente, sea $\mathbf{x}_q$ el registro a clasificar y sean $\mathbf{x}_1, ..., \mathbf{x}_k$ los k vecinos más cercanos podemos etiquetar

$$E(\mathbf{x}_q) \leftarrow argmax_{v \in V} \sum_{i=1}^{k} \delta(v, E(\mathbf{x}_i)) \tag{4.1}$$

donde $\delta(a, b) = 1$ si $a = b$ ó 0 en cualquier otro caso. Siendo $E(\mathbf{x}_k)$ la etiqueta o clasificación asignada a $\mathbf{x}_k$.

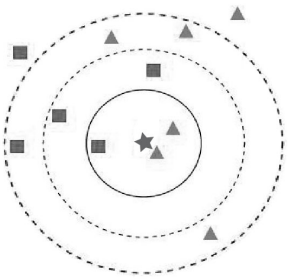

Figura 4.1: Asignación según *k vecinos más cercanos*. Cuando k=3 el algoritmo clasifica a la estrella como triángulo. Para k=5, tres vecinos están etiquetados como cuadrados y dos como triángulos por lo cual la clasificación será diferente que con k=3. En el caso de k=9 la clasificación será igual que k=3.

4.2. *Clustering* usando *Mean Shift*

Mean Shift, como se vió en el Capítulo 3 calcula el estimador del gradiente de la función de densidad para un conjunto de datos. Dado un punto de partida, el procedimiento se desplaza iterativamente hasta alcanzar una moda local. De esta forma se determinan los centros de *cluster* y clasifican cada uno de los puntos de la muestra según el centro de *cluster* más cercano.

Sin embargo, esto no es fácil de llevarlo a la práctica debido a varias circunstancias como la presencia de mesetas donde no hay un máximo local claro, regiones de baja densidad donde la convergencia es pobre y puntos de ensilladura. Por otro lado, cuando el conjunto de datos es grande (más de mil puntos o pixels) la complejidad computacional se vuelve demasiado costosa, ya que cuenta con un orden de complejidad de $O(n^2)$ para un conjunto de n datos. Además, esto se agraba según la dimensionalidad de los datos. Presentamos a continuación el algoritmo de *clustering* el cual tiene en cuenta las dificultades anteriormente mencionadas y cuyo orden de complejidad es de $O(mn)$ donde $m << n$.

4.3. Algoritmo de *Clustering*

Los pasos descriptos por Meers y Comaniciu en [CM98] del algoritmos son los descriptos debajo:

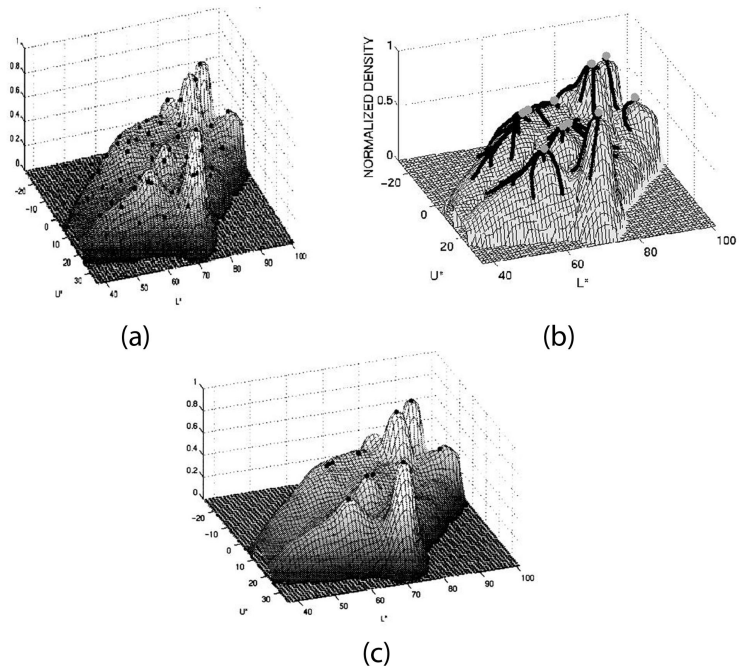

Figura 4.2: Procedimiento de *clustering* utilizando *Mean Shift*. En (a)Se muestran los *data samples* sobre la función de densidad estimada utilizando un *kernel* de Epanechnikov. (b)Los *data samples* convergen a los centros de *clusters*. (c)Los centros de *clusters*.

Algoritmo 1 Clustering utilizando *Mean Shift*

MEAN SHIFT CLUSTERING

1 Defnir una subdivisión del *espacio de características* con hiperesferas $S_h(\mathbf{x})$ de radio h y centradas en $\mathbf{x}$. A fin de reducir el costo computacional, un conjunto de m puntos $\mathbf{x}_1, ..., \mathbf{x}_m$ (data samples) son seleccionados de forma aleatoria como centros del conjunto de hiperesferas definidas, con $m << n$. Dos hiperesferas centradas en $\mathbf{x}_i$ y $\mathbf{x}_j$ con $i \neq j$ no deben estar más cerca que una distancia h. Los puntos elegidos deben estar definidos en áreas populosas, para ello se define un umbral *minMembers* que determina la mínima cantidad de datos que debe contener cada hiperesfera.

2 Aplicar *Mean Shift* desde cada uno de los m puntos seleccionados. Notar que el costo computacional de ésta operación es cercano al de $O(mn)$ con

2 $m << n$. Las m hiperesfereas definidas como puntos de partida utilizando el método de *Mean Shift* deben barrer con casi la totalidad de los puntos de muestra en el espacio.

3 Correr los centros de *cluster* y volver a aplicar el procedimiento de *Mean Shift*. Esto se hace debido a la posible existencia de mesetas. En caso de volver a converger al mismo centro no hay problemas, caso contrario crear un nuevo centro. El centro original y el nuevo centro serán alterados o eliminados en los siguientes pasos del algoritmo.

4 Redefinir los centros de *clúster*. Para cualquier par de centros de *clúster* que estén a una distancia menor que el radio h eliminarlos y crear un nuevo centro de *cluster* situado en la distancia media entre los dos anteriores. Notar que el número de centros de *cluster* disminuye en este paso.

5 Validar los centros de *clusters*. Dados dos centros de *cluster* $\mathbf{x}_i$ y $\mathbf{x}_j$ se testea la presencia de un valle entre ellas. Una esfera $S_h(\mathbf{x})$ determinando la densidad estimada utilizando un *kernel* de Epanechnikov, se translada de $\mathbf{x}_i$ a $\mathbf{x}_j$ con pasos de longitud h. Si la relación entre el $\min(f(\mathbf{x}_i), f(\mathbf{x}_j))$ y la minima densidad encontrada es más grande que un umbral determinado por *lowRegion* se supone que hay un valle, sino se elimina del conjunto de centros de *clusters* aquel con menor densidad entre $\mathbf{x}_i$ y $\mathbf{x}_j$.

6 Delinear los *clusters*. A esta altura están definidos todos los centros de *cluster*. Para asignar a cada punto un centro se emplea la técnica de *k-vecinos más cercanos* puesto que es un algoritmo no computacionalmente costoso.

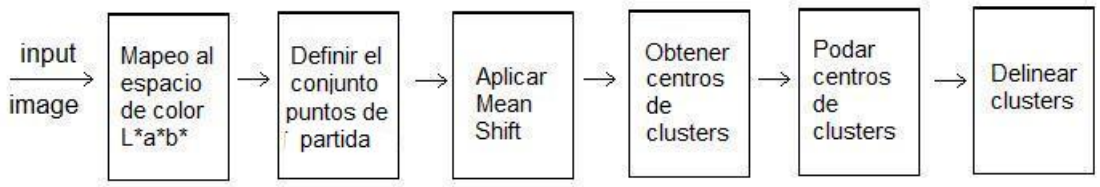

Figura 4.3: Flujo del algorimo de *Clustering* utilizando *Mean Shift*

4.4. Experimentos

Experimento 1. Representación de una imagen clusterizada en el espacio de color.

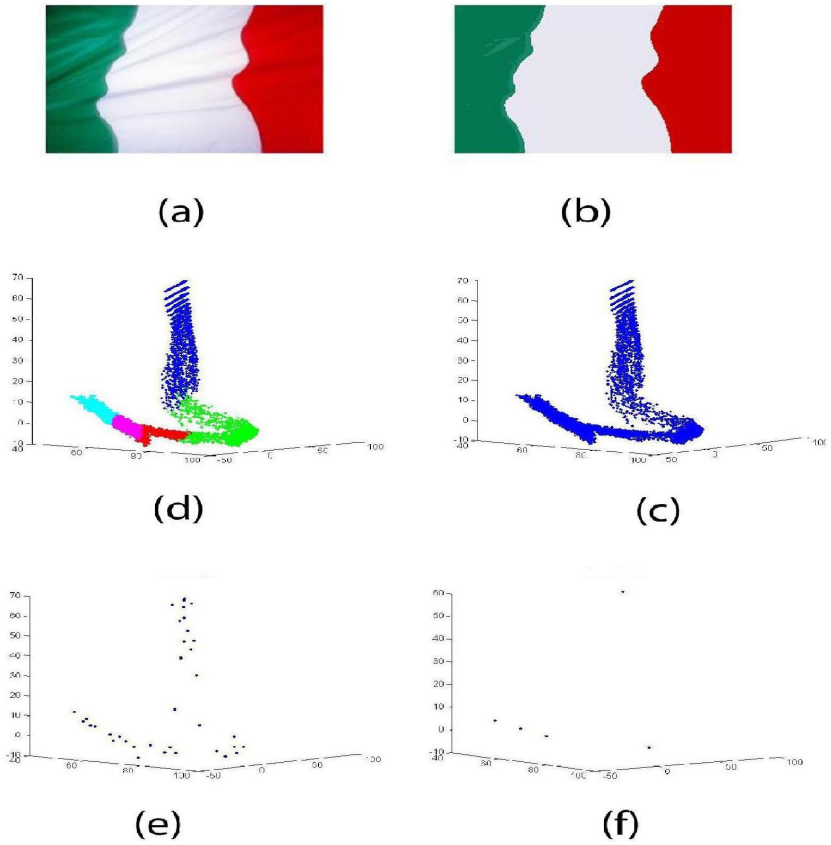

Figura 4.4: Primer experimento. (a)Imagen original. (b)Imagen Segmentada. (c)*Data Points* en el *espacio de características* (43.542 points). (d)*Clustering* (5 centros de *clusters* encontrados). (e)*Data Samples* (41 puntos representativos). (f)Cluster centers (5 centros de *clusters*).

El primer experimento consta de mostrar los grupos de *clusters* formados en una imagen de la bandera italiana la cual posee tres centros principales de *clusters* claramente definidos. Sin embargo se encontraron cinco centros de *cluster*. Los pintados de color azul representan la franja roja de la bandera mientras que aquellos de color verde claro a la franja central de color blanco. También se identificaron tres tonalidades del color verde, éstos son los *clusters* de color cyan, rosa y rojo ordenados de mayor a menor en relación a su intesidad en color. La cantidad de puntos tridimensionales de la imagen es de 43.542 y el de *clusters* encontrados es 5. El *bandwidth* elegido es de $h = 15$ y mínimo de miembros dentro del *kernel* para cada punto de partida igual a $minMembers = 20$. Las gráficas de los *data points*, *data samples* (puntos de partida), cluster centers (centros de cluster) y la clasificación pueden verse en la Figura 4.4.

Experimento 2. Clusterización para diferentes mínimos en la cantidad de miembros de partida.

En el segundo experimento realizamos la clusterización para diferentes cantidades de puntos que debe contener la hiperesfera en el punto de partida establecidos por el umbral *minMembers*. La imagen es la de un pescadito de 50.508 *data points* cuyas tonalidades en color son diferentes pero parecidas lo cual lo hace adecuado para lo que se quiere testear. Se observa, cómo a medida que el umbral aumenta la cantidad de centros de *clusters* disminuye y los detalles en la imagen se pierden. Debido a que el umbral de aceptación para un *data sample* es más estricto a medida que *minMembers* aumenta, *data points* asociados a un centro de *cluster* poco popular no serán tenidas en cuenta y estarán asociados a otro más lejano pero más denso en población. El *bandwidth* con el que se realizaron las pruebas es de $h = 5$. Los diferentes valores de *minMembers* son de 15, 25, 35, 45 y 55 y la cantidad de *clusters* encontrados fue de 20, 17, 14, 10 y 8 respectivamente. Los resultados se pueden ver en la Figura 4.5.

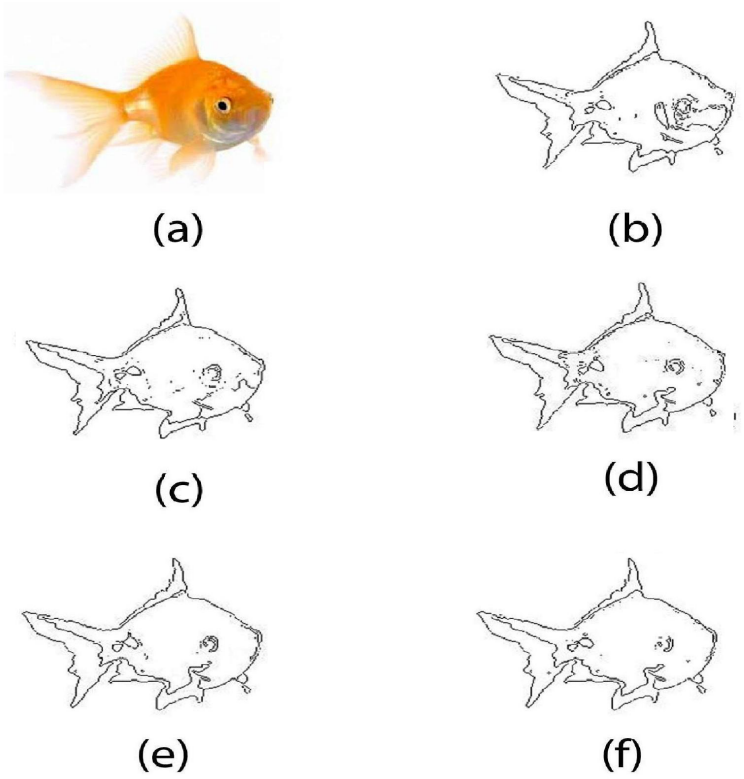

Figura 4.5: Segundo experimento. Contornos de la imagen segmentda. (a)Imagen original. (b)*Clustering* con *minMembers* = 15. (c)*Clustering* con *minMembers* = 25. (d)*Clustering* con *minMembers* = 35. (e)*Clustering* con *minMembers* = 45. (f)*Clustering* con *minMembers* = 55.

Experimento 3. Clusterización para diferentes valores de *bandwidth*.

Como se explicó en el Capítulo 2 el valor del *bandwidth* influye de manera importante en los resultados del *clustering* perdiendo definición a medida que el ancho de ventana crece y robustez al ruido a medida que el ancho disminuye. Se realizó una prueba con la imagen de un camino de 98.784 *data points*. A la segmentación se ha aplicado un detector de bordes.

En el experimento realizado se nota que en (b) discontinuidades no significativas en el pasto son marcadas haciendo que los valores para (c) sean mejores en este caso. En (d) se pierden detalles importantes de la parte superior del edificio y en (e) además, los límites entre el camino y el pasto se funden. Los valores utilizados para los *bandwidth* fueron de 10, 20, 30 y 40 con un número de *clusters* encontrados de 39, 10, 4 y 2 repectivamente. Los resultados son los que se muestran en la Figura 4.6.

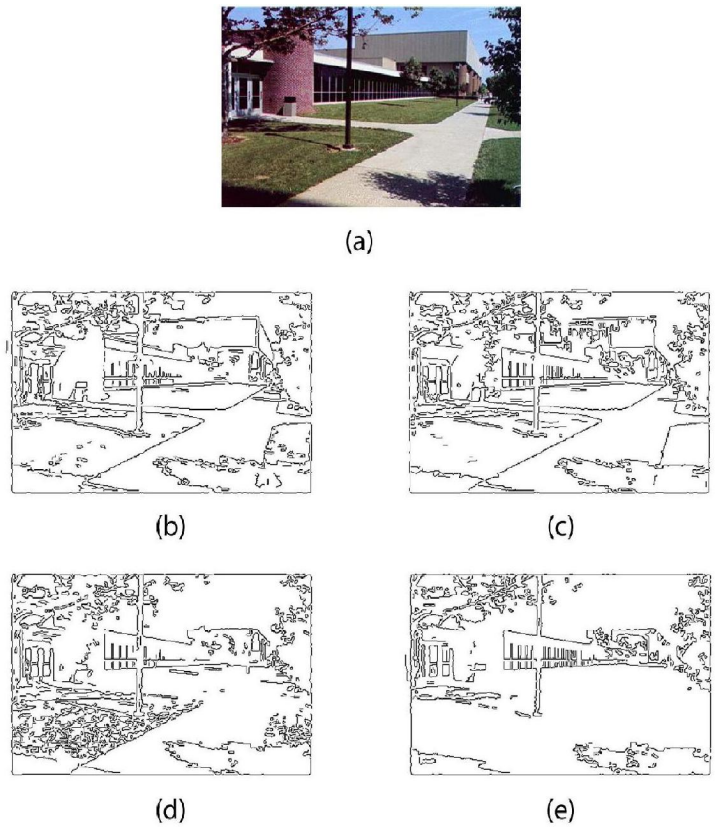

Figura 4.6: Tercer experimento. Contornos de la imagen segmentada. (a)Imagen original. (a)*Clustering* con $bw = 10$ (39 cluster encontrados). (b)*Clustering* con $bw = 20$ (10 cluster encontrados). (c)*Clustering* con $bw = 30$ (4 cluster encontrados). (d)*Clustering* con $bw = 40$ (2 cluster encontrados).

Capítulo 5

Procesamiento en el dominio del rango-espacio

Pese a que los resultados de *clusterización* obtenidos en el Capítulo 4 pueden ser utilizados para segmentación, algunos aspectos importantes no son tenidos en cuenta. Pequeños cambios en la iluminación y deformaciones en los objetos impactan de forma negativa en el desempeño del algoritmo propuesto. Estas cuestiones pueden resolverse si tenemos en cuenta aspectos relacionados a la información espacial, es decir el procesamiento en el *dominio empalmado*.

5.1. Dominio empalmado

Una imagen es representada como una matriz dimensional de puntos r-dimensionales (o pixeles), donde $r = 1$ en el caso de imágenes en escala de grises, $r = 3$ imágenes a color y $r > 3$ en el caso multiespectral. La localización de los pixeles está determinada por el dominio espacial, mientras que la dimensionalidad por el dominio del rango. Ambos dominios conforman el *dominio empalmado* (o su traducción en inglés *join domain*) de dimensión $d = r + 2$. Para ambos dominios se supone el uso de una métrica euclideana.

El *kernel* propuesto resulta del producto de otros dos radialmente simétricos y cuya expresión es la siguiente:

$$k_{h_s,h_r}(x) = \frac{C}{h_s^2 h_r^r} k\left(\left\|\frac{\mathbf{x}^s}{h_s}\right\|^2\right) k\left(\left\|\frac{\mathbf{x}^r}{h_r}\right\|^2\right) \tag{5.1}$$

donde h_r y h_s son los *bandwidths* en el dominio del rango y el espacio respectivamente, C es la constante de normalización y, $\mathbf{x}^s$ y $\mathbf{x}^r$ son los puntos en el dominio espacial y del rango. Los *kernels* utilizados en ambos dominios deben ser del mismo tipo. En la práctica, los *kernels* de *Epanechnikov* producen buenos resultados, aunque siempre dependiendo de una elección

adecuada de los *bandwidth*.

Este procesamiento en el *dominio empalmado* es utilizado para algoritmos de suavizado preservando bordes y segmentación de imágenes que veremos en detalle en las siguientes secciones. Se utilizará el espacio de color $L^*u^*v^*$ aunque en el caso de imágenes en escala de grises sólo la componente L^* es tenida en cuenta.

5.2. Suavizado preservando bordes

Dada una imagen de n píxeles, sean $\{\mathbf{x}_i\}_{i=1\ldots n}$ y $\{\mathbf{z}_i\}_{i=1\ldots n}$ los puntos de entrada d-dimensionales original y suavizados en el *dominio empalmado*. Por cada pixel,

Algoritmo 2 Algoritmo de Suavizado preservando bordes

MEAN SHIFT SMOOTHING

1 Inicializar $j = 1$ y $\mathbf{y}_{i,j} = \mathbf{x}_i$,
2 Computar $\mathbf{y}_{i,j+1} = \frac{1}{n_j} \sum_{x \in S(\mathbf{y}_i)} \mathbf{y}_{i,j}$
3 $j \leftarrow j + 1$ volver a 2 hasta converger
4 Asignar $\mathbf{z}_i = (\mathbf{x}_i^s, \mathbf{y}_{i,conv}^r)$

Los superíndices r y s hacen referencia a las coordenas del dominio del rango y el espacio respectivamente y n_j es la cantidad de puntos que caen dentro del *kernel* en cada iteración.

En el paso 2 del Algoritmo 2, para encontrar los puntos que pertenecen a $S(\mathbf{y}_j)$, se tiene en cuenta el dominio espacial y luego entre los puntos seleccionados (aquellos puntos que se encuentran dentro de la hiperesfera de radio h_S) aplicamos *Mean Shift* entre los que están dentro de la vecindad en el dominio del rango. El *espacio de características* se podría describir como una grilla determinada por las coordenadas espaciales, donde cada posición contiene un punto característico r-dimensional (ver la Figura 5.1). Sea k_c el número de iteraciones necesarias por *Mean Shift* para converger utilizando un *kernel* de tamaño $(2h_s + 1)^2$ píxeles de la imagen, la complejidad del algoritmo de suavizado es de $k_c(2h_s + 1)$ por pixel de la imagen. Para más detalles consultar a la bibliografía[CM99].

5.3. Segmentación

La segmentación vía *Mean Shift* en el *dominio empalmado* utiliza entre sus pasos al algoritmo 2 de suavizado. Cada pixel tiene asociado el color del

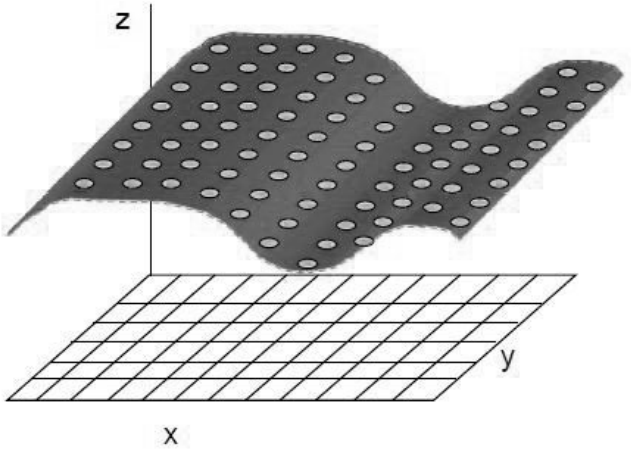

Figura 5.1: La grid de coordeanas x e y, representa las coordenadas espaciales de la imagen y los puntos, que determinan la sábana, representan el nivel de gris en la ccordenada z. En este gráfico el valor de $r = 1$ pero bien podria ser de $r = 3$ en el caso de imagenes en color.

centro de cluster al que pertenece y luego se fusionan aquellos centros que se encuentran cercanos en el *dominio empalmado*. Sean $\{\mathbf{x}_i\}_{i=1...n}$ y $\{\mathbf{z}_i\}_{i=1...n}$ los puntos de entrada d-dimensionales y suavizados en el dominio empalmado. Sea $\{L_i\}_{i=1...n}$ la etiqueta asignada del i-esimo pixel en la imagen segmentada.

Algoritmo 3 Algoritmo de Segmentación

MEAN SHIFT SEGMENTATION

1 Correr el procedimiento de suavizado explicado en el algoritmo 2 para la imagen y almacenar toda la información en $\mathbf{z}$, es decir $\mathbf{z}_i = \mathbf{y}_{i,c}$
2 Delinear los clusters $\{C_p\}_{p=1...m}$ agrupando todos los $\mathbf{z}_i$ que estén más cerca que h_s en el dominio espacial y h_r en el dominio del rango
3 Para cada $i = 1...n$, asignar $L_i = \{p|\mathbf{z}_i \in C_p\}$, es decir clasificar cada pixel de acuerdo al cluster al que pertence
4 Opcional: Eliminar aquellas regiones de la imagen con menos de M pixels pertenecientes a un mismo cluster

El primer paso es el mismo que se hace en el filtrado con la diferencia de que toda la información es almacenada en $\mathbf{z}_i$, no sólo la parte del rango. Este primer paso (que corresponde al suavizado) es el más costoso computacionalmente haciendo a los pasos siguientes depreciables, es decir que el orden de

complejidad para ambos algoritmos es el mismo.

5.4. Experimentos. Comparación en la segmentación utilizando clustering y el dominio empalmado.

A continuación se mostrará los resultados de la segmentación obtenida con el Algoritmo 3 y lo compararemos con los resultados obtenidos del Algoritmo 1 de *clustering* del Capitulo 4. La implementación fue desarrollada por la Universidad de Rutgers bajo el nombre EDISON e implementada en C++. El código y ejecutable se encuentran disponibles en internet. A diferencia de la implementación del *clustering*, la segmentación no fue desarrollada en este trabajo puesto que el tratamiento en una imagen pixel a pixel no tiene un buen desempeño en MATLAB. Se realizaron dos experimentos con dos imágenes diferentes.

En la primer Figura 5.2 es la de una sala de estar. Las diferencias que en-

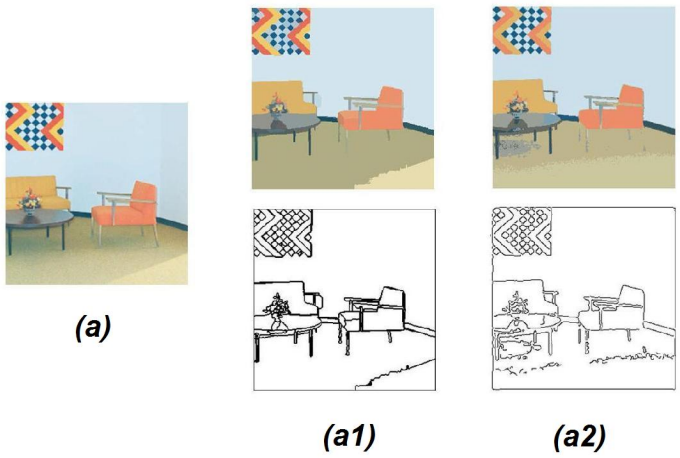

Figura 5.2: Clustering vs. Segmentación. (a)Imagen original de un living (90.000 *data points*). (a1)Segmentación y contorno utilizando en el *dominio empalmado* ($h_s = 7$) y ($h_r = 7$). (a2)Segmentación y contorno utilizando en el dominio del rango ($h_s = 7$).

contramos entre el *clustering* y la segmentación propuesta se encuentran en los límites de la alfombra con el suelo y la sombra producida por la mesa ratona. El método *clustering* realiza una mala segmentación debido a pequeñas variaciones en color las cuales no son tenidas en cuenta, a diferencia de cuando se utiliza un *dominio empalmado*.

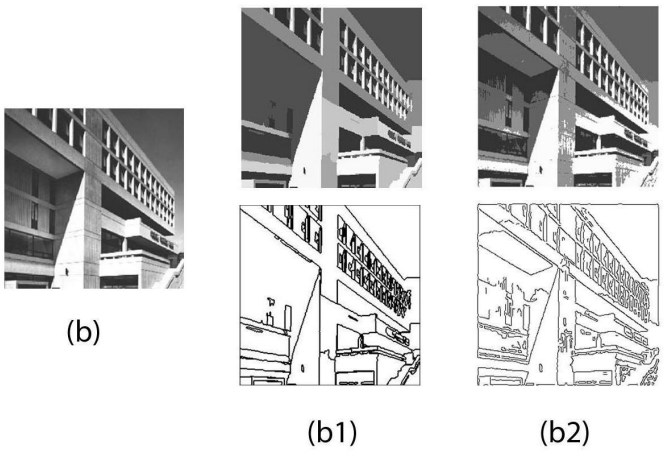

Figura 5.3: Clustering vs. Segmentación. (b)Imagen original del MIT (90.000 *data points*). (b1)Segmentación y contorno utilizando en el *dominio empalmado* ($h_s = 5$ y $h_r = 5$). (b2)Segmentación y contorno utilizando en el dominio del rango ($h_s = 5$).

En la segunda Figura 5.3 del MIT (Instituto tecnológico de Massachusetts) se observan falsos contornos debido a la reflectancia del sol en algunas partes del edificio. En la clusterización, pequeñas variaciones de color son tenidas en cuenta haciendo que regiones uniformes sean detectadas con ruido.

Capítulo 6

Seguimiento

El *seguimiento* de objetos en tiempo real es de gran relevancia en muchas aplicaciones de visión como cámaras de vigilancia e interfaces de percepción de usuarios, entre otros. En aplicaciones en tiempo real, es imprescindible un buen uso de los recursos del sistema y por esta razón es deseable mantener una complejidad computacional lo más baja posible.

Existen dos enfoques para alcanzar esto, uno es *bottom-up* y el otro *top-down*. Los métodos relacionados con el primer enfoque, clasifican dentro del grupo "*Representación y Localización del Objetivo*" donde se dá un modelo del objetivo (o *target*) a seguir y luego se utiliza una técnica para su localización cuadro a cuadro. La técnica desarrollada en este Capítulo pertenece a este enfoque. Aquellas técnicas relacionadas al segundo enfoque, se clasifican como "*Filtrado y Asociación de Datos*" en donde la información que caracteriza al objetivo en cada cuadro está dada por un estado $\mathbf{x}_k$ a estimar que varía en el tiempo. Por otro lado, contamos con una serie de observaciones $\{\mathbf{z}_{1:k}\}$ que permiten estimar el estado actual o, dicho en términos probabilisticos buscamos la función de probabilidad $p(\mathbf{x}_k|\mathbf{z}_{1:k})$. La técnica que presentaremos en la siguiente sección, el *Filtro de Kalman*, pertenece a esta familia.

En este Capítulo se verá como se puede realizar seguimiento de objetos no rígidos utilizando la técnica de *Mean Shift*. Para ello se define una representación del objetivo o *target* a través del uso de un *kernel* isotrópico y su histograma de color. Se determina una función de similaridad entre el *modelo objetivo* y el *objetivo candidato* la cual se maximiza utilizando *Mean Shift*, obteniendo mejores resultados en comparación con métodos que utilizan la fuerza bruta.

6.1. Representación del objetivo

El *modelo objetivo* puede ser representado en primera instancia por su histograma de color normalizado $\widehat{q}$ (de la ecuación 6.1) después de haber escogido un *espacio de características* adecuado. Sin perder generalidad, se supone que el *modelo objetivo* está centrado en el origen de coordenadas. En los cuadros o *frames* siguientes, el *objetivo candidato* se define centrado en la posición $\mathbf{y}$ y está caracterizado por su correspondiente histograma normalizado $\widehat{p}(\mathbf{y})$ de la ecuación 6.2. Para satisfacer un bajo costo computacional y cumplir con el requisito de que una aplicación funcione en tiempo real, es necesario utilizar un histograma de m particiones (o *bines*).

$$\widehat{q} = \{\widehat{q}_u\}_{u=1\ldots m} \text{ con } \sum_{u=1}^{m} \widehat{q}_u = 1 \tag{6.1}$$

$$\widehat{p}(\mathbf{y}) = \{\widehat{p}_u(\mathbf{y})\}_{u=1\ldots m} \text{ con } \sum_{u=1}^{m} \widehat{p}_u = 1 \tag{6.2}$$

Otras estimaciones discretas de densidad podrían ser utilizadas, sin embargo $\widehat{p}$ y $\widehat{q}$ resultan suficientemente buenas para nuestro fin.

6.1.1. Coeficiente de *Bhattacharyya*

Se define el coeficiente de *Bhattacharyya* como

$$\rho(\mathbf{y}) = \rho[\widehat{r}(\mathbf{y}), \widehat{t}] = \frac{\widehat{r}(\mathbf{y})\widehat{t}}{\|\widehat{r}(\mathbf{y})\|_2 \left\|\widehat{t}\right\|_2} = \sum_{u=1}^{m} \sqrt{\widehat{p}_u(\mathbf{y}), \widehat{q}_u} \tag{6.3}$$

donde $\widehat{r}(\mathbf{y}) = \sqrt{\widehat{p}(\mathbf{y})}$ y $\widehat{t} = \sqrt{\widehat{q}}$. Se debe maximizar el coeficiente de *Bhattacharyya* ρ respecto a $\mathbf{y}$, buscando el mejor *matching* entre $\widehat{p}(\mathbf{y})$ y $\widehat{q}$ en el próximo *frame* a partir de un entorno centrado en su posición $\mathbf{y}$ en el anterior. El coeficiente de *Bhattacharyya* puede interpretarse como el coseno del ángulo (ver Figura 6.1) entre los vectores unitarios m dimensionales $\widehat{r}(\mathbf{y}) = (\sqrt{\widehat{p}_1(\mathbf{y})},\ldots,\sqrt{\widehat{p}_m(\mathbf{y})})$ y $\widehat{t} = (\sqrt{\widehat{q}_1},\ldots,\sqrt{\widehat{q}_m})$.

Si sólo es tenida en cuenta la información de color para caracterizar el objetivo, esta función de similaridad presenta problemas debido a que los bordes suelen estar expuestos a deformaciones u oclusiones, y además un procedimiento basado en la estimación del gradiente resulta dificil de aplicar y es costoso computacionalmente. Por esta razón aplicamos máscaras espaciales, mediante el uso de *kernels* isotrópicos que ponderan los datos teniendo en cuenta el dominio espacial.

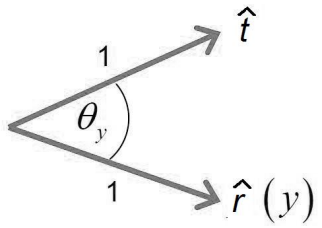

Figura 6.1: Interpretación geométrica del coeficiente de *Bhattacharyya*. El mejor *matching* para las densidades se produce cuando θ_y es igual a cero representando el máximo valor para la función coseno.

6.1.2. Modelo Objetivo

Sean $\{\mathbf{x}_i^*\}_{1=1...n}$ las posiciones relativas respecto al centroide del objetivo (ver Figura 6.2), de los n pixeles de la región definida como el objetivo centrado en el origen. Se utiliza un *kernel* isotrópico, con un perfil k convexo y monotonamente decreciente que asigna pesos según su distancia al origen. El uso de estos pesos hace más robusta a la estimación de la función de densidad propuesta puesto que son los píxeles extremos los más propensos a oclusiones e interferencias con el fondo. Sea la función $b : Color \rightarrow \{1...m\}$ que dado el color de un pixel en la imagen retorna el bin al que pertenece e $I : Posicion \rightarrow Color$ la función que dada una posición en una imagen retorna el color del pixel en esa posición, se define el *modelo objetivo* como:

$$\widehat{q}_u = C \sum_{i=1}^{n} k(\|\mathbf{x}_i^*\|^2)\delta(b(I(\mathbf{x}_i^*)), u) \text{ con } u = 1...m \tag{6.4}$$

donde $\delta(.)$ es la función delta de *Kronecker*

$$\delta(i,j) = \begin{cases} 1 & \text{si } i = j \\ 0 & \text{caso contrario} \end{cases}$$

y C es la constante de normalización que hace que $\sum_{u=1}^{m} \widehat{q}_u = 1$, es decir que el valor de C es:

$$C = \frac{1}{\sum_{i=1}^{n} k(\|\mathbf{x}_i^*\|^2)}$$

6.1.3. Objetivo Candidato

Sean $\{\mathbf{x}_i\}_{i=1...n_h}$ las posiciones absolutas de los píxeles del *objetivo candidato* centrado en $\mathbf{y}$ en el cuadro actual (ver Figura 6.2), utilizando el mismo *kernel* parametrizado por h que determina su ancho, el *objetivo candidato* está dado por

$$\widehat{p}_u(\mathbf{y}) = C_h \sum_{i=1}^{n_h} k\left(\left\|\frac{\mathbf{y} - \mathbf{x}_i}{h}\right\|^2\right)\delta(b(I(\mathbf{x}_i)), u) \text{ con } u = 1...m \tag{6.5}$$

donde C_h es la constante de normalización $C_h = \frac{1}{\sum_{i=1}^{n_h} k\left(\left\|\frac{\mathbf{y}-\mathbf{x}_i}{h}\right\|^2\right)}$. Notar que C_h no depende de $\mathbf{y}$, sólo determina la posición en la que está centrada. Por esta razón, puede ser precalculada dado un *kernel* y su ancho h. El parámetro h se utiliza para poder escalar el *objetivo candidato* permitiendo cambios en su tamaño.

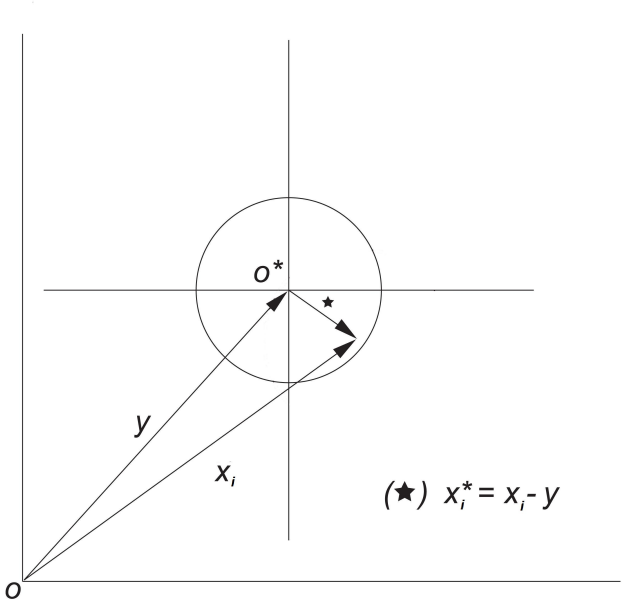

Figura 6.2: El eje cartesiano representa las posiciones en una imagen. El círculo representa el objetivo a seguir y su posición relativa al centroide O^* es cero mientras que su posición absoluta respecto al origen O es $\mathbf{y}$. Las posiciones absolutas para cada $\mathbf{x}_i^*$ queda determinada por $\mathbf{x}_i - \mathbf{y}$ respecto al origen.

6.1.4. Función suave de similaridad

El coeficiente de *Bhattacharyya* con $\widehat{p}(\mathbf{y})$ y $\widehat{q}$ definidos en 6.4 y 6.5, hereda las propiedades del perfil de un *kernel* expuestas en la sección 3.2.2. Esto garantiza una función de similaridad diferenciable e ideal para la aplicación de un procedimiento de estimación del gradiente para la búsqueda de un máximo local. Existen otras funciones de similaridad las cuales son estudiadas en [CRH01]. Un gráfico del coeficiente de *Bhattacharyya* se puede observar en la Figura 6.3 extraida del trabajo de Comaniciu y Meer[CRM00].

6.2. Métrica basada en el coeficiente de *Bhattacharyya*

Para definir una métrica de similaridad entre el *modelo objetivo* y el candidato, ésta debe cumplir con las propiededes de distancia. Definimos

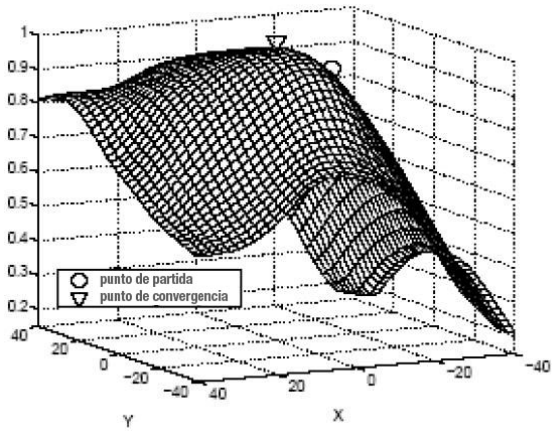

Figura 6.3: Se muestran los valores arrojados por el coeficiente de *Bhattacharyya* centrados en un punto determinado de la imagen. Con un círculo se indica el punto de partida de la iteración de *Mean Shift* y con un triángulo el de convergencia.

entonces la distancia entre dos distribuciones discretas como

$$d(\mathbf{y}) = \sqrt{1 - \rho[\widehat{r}(\mathbf{y}), \widehat{t}]} \tag{6.6}$$

La ecuación 6.6 cumple con las propiedades de distancia.

6.3. Localización del objetivo

Para obtener la localización de un objetivo en un cuadro en el instante t, se debe minimizar la función de distancia de la ecuación 6.6 respecto a $\mathbf{y}$. El procedimiento de localización, tiene en cuenta como punto de partida, la posición del centroide del objeto en el cuadro anterior. Puesto que nuestra función distancia es suave, el método de *Mean Shift* que utiliza información del gradiente, es ideal.

La función de densidad está determinada en este caso por la información de color, pero bien podria aplicarse en base a texturas, bordes o combinaciones de ellos. En esta sección se supone que:

1. Se cuenta con la detección y localización del objeto a seguir en el primer cuadro.

2. Se realizan constantes ajustes al *modelo objetivo* por cambios significativos en el color.

6.3.1. Minimización de distancia

La minimización de la distancia propuesta en la ecuación 6.6, se traduce en maximizar el coeficiente de *Bhattacharyya* $\widehat{\rho}(\mathbf{y})$. La búsqueda de la nueva posición comienza en la posición $\widehat{\mathbf{y}}_0$ que representa la posición del objetivo en el cuadro anterior y requiere del cálculo de $\{\widehat{p}_u(\widehat{\mathbf{y}}_0)\}_{u=1...m}$ antes de cada corrimiento. Utilizando la serie de Taylor alrededor del punto $\widehat{p}_u(\mathbf{y}_0)$, podemos aproximar el coeficiente de *Bhattacharyya* de la siguiente manera:

$$\rho[\widehat{r}(\mathbf{y}),\widehat{t}] \approx \frac{1}{2}\sum_{u=1}^{m}\sqrt{\widehat{p}_u(\widehat{\mathbf{y}}_0)\widehat{q}_u} + \frac{1}{2}\sum_{u=1}^{m}\widehat{p}_u(\mathbf{y})\sqrt{\frac{\widehat{q}_u}{\widehat{p}_u(\widehat{\mathbf{y}}_0)}} \tag{6.7}$$

La aproximación es válida cuando la posición $\mathbf{y}$ que maximiza la función no dista demasiado de la posición $\mathbf{y}_0$, lo cual se puede suponer cierto entre cuadros consecutivos. Los valores del histograma ponderado $\widehat{p}_u(\mathbf{y})$ para $u = 1...m$ cercanos o iguales a cero no son tenidos en cuenta para evitar problemas numéricos. Reescribiendo el resultado anterior obtenemos que:

$$\rho[\widehat{p}(\mathbf{y}),\widehat{q}] \approx \frac{1}{2}\sum_{u=1}^{m}\sqrt{\widehat{p}_u(\widehat{\mathbf{y}}_0)\widehat{q}_u} + \frac{C_h}{2}\sum_{i=1}^{n_h} w_i k\left(\left\|\frac{\mathbf{y}-\mathbf{x}_i}{h}\right\|^2\right) \tag{6.8}$$

donde

$$w_i = \sqrt{\frac{\widehat{q}_u}{\widehat{p}_u(\widehat{\mathbf{y}}_0)}} \text{ tal que } b(I(\mathbf{x}_i)) = u \tag{6.9}$$

Para minimizar la distancia de la ecuación 6.8, sólo el segundo término debe ser maximizado ya que el primero no depende de la posición $\mathbf{y}$, y por lo tanto no es tenido en cuenta. Notar que el segundo término representa la densidad estimada, computada con un *kernel* de perfil $k(x)$ centrada en $\mathbf{y}$ con los datos $\mathbf{x}_i$ ponderados por w_i. La moda de esa densidad dentro de la vencindad local se obtiene utilizando el procedimiento de *Mean Shift* visto en el Capítulo 3

$$\widehat{\mathbf{y}}_{n+1} = \frac{\sum_{i=1}^{n_k}\mathbf{x}_i w_i g\left(\left\|\frac{\widehat{\mathbf{y}}_n-\mathbf{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^{n_k}\mathbf{x}_i g\left(\left\|\frac{\widehat{\mathbf{y}}_n-\mathbf{x}_i}{h}\right\|^2\right)} \tag{6.10}$$

donde $g(\mathbf{x}) = -k'(\mathbf{x})$, suponiendo que k es derivable en $[0,\infty)$, excepto para una cantidad finita de puntos. Como es de suponer, $\mathbf{y}_n$ e $\mathbf{y}_{n+1}$ representan la posición inicial y final respectivamente y w_i los pesos asociados a cada uno de los puntos de la muestra.

6.3.2. Intuición en los pesos asociados

Se interpretará el significado de la ponderación de w_i mediante un sencillo ejemplo. Suponiendo que se cuenta con un *modelo objetivo* que consta de

dos colores ($m = 2$), tal como se muestra en la Figura 6.4, de modo que los valores de su histograma son $q_1 = 0{,}6$, $q_2 = 0{,}4$ y $q_i = 0$ para todo otro i. Se van a enumerar los histogramas de cuatro posibles *objetivos candidatos* centrados en diferentes posiciones, alrededor de la posición del *target*, de manera que los valores para cada uno de ellos son:

1. $p_1 = 0{,}6$, $p_2 = 0{,}4$ y $p_i = 0$ para todo otro i
2. $p_1 = 0{,}5$, $p_2 = 0{,}5$ y $p_i = 0$ para todo otro i
3. $p_1 = 0{,}2$, $p_2 = 0{,}8$ y $p_i = 0$ para todo otro i
4. $p_1 = 0{,}7$, $p_2 = 0{,}3$ y $p_i = 0$ para todo otro i

Se analizará para cada caso, el corrimiento del Algoritmo de *Mean Shift* que se realiza para la localización del *target*. En el primer caso, el *matching* entre el *objetivo candidato* y modelo es exacto, los valores de $w_1 = \sqrt{0{,}6/0{,}6} = 1$, $w_2 = \sqrt{0{,}4/0{,}4} = 1$ son los óptimos. En el segundo caso el *objetivo candidato* está centrado de tal forma que contiene ambos colores en igual proporción, de modo que $w_1 = \sqrt{0{,}6/0{,}5} > 1$, $w_2 = \sqrt{0{,}4/0{,}5} < 1$ y $w_i = 0$. Los pixeles color gris oscuro, tendrán asociados un mayor peso haciendo que la ventana se desplace hacia la posición en la que está el objetivo. En el tercer ejemplo, $w_1 = \sqrt{0{,}6/0{,}2} >> 1$ y $w_2 = \sqrt{0{,}4/0{,}8} << 1$ provocando una ponderación aún mayor para un corrimiento hacia la región de pixeles grises oscuros. El último caso es contrario a los dos últimos, puesto que hay una proporción de gris claro menor a la del *modelo objetivo* haciendo que $w_1 = \sqrt{0{,}6/0{,}7} < 1$ y $w_2 = \sqrt{0{,}4/0{,}3} > 1$; y en consecuencia también es menor el corrimiento hacia estos píxeles.

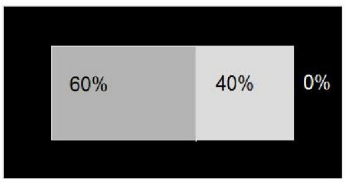

Figura 6.4: El objetivo tiene 60 porciento de gris oscuro y 40 porciento de gris claro, esto se traduce a $q_1 = 0{,}6$, $q_2 = 0{,}4$ y $q_i = 0$ para el resto de los valores de i.

6.4. Algoritmo de Seguimiento

Dado un *modelo objetivo* $\{\widehat{q}_u\}_{u=1...m}$ y su posición $\mathbf{y}_0$ en el cuadro anterior presentamos el algoritmo de seguimiento propuesto por Comaniciu, Meer y Ramesh

Algoritmo 4 Seguimiento utilizando Mean Shift

MAXIMIZACIÓN DEL COEFICIENTE DE BHATTACHARYYA($\widehat{q}_u, \mathbf{y}_0$)

1 Inicializar la posición del *target* en el *frame* actual con la posición $\mathbf{y}_0$,
2 Computar el histograma ponderado $\{\widehat{p}_u(\widehat{\mathbf{y}}_0)\}_{u=1\ldots m}$ y evaluar

$$\rho[\widehat{p}(\widehat{\mathbf{y}}_0), \widehat{q}] = \sum_{u=1}^{m} \sqrt{\widehat{p}_u(\widehat{\mathbf{y}}_0)\widehat{q}_u}$$

3 Derivar las ponderaciones $\{w_i\}_{i=1\ldots n_k}$ acorde a los visto en 6.9
4 Encontrar la nueva posición del *objetivo candidato* $\widehat{y}_1$ acorde a 6.10
5 Computar el histograma ponderado $\{\widehat{p}_u(\widehat{\mathbf{y}}_1)\}_{u=1\ldots m}$ y evaluar el
6 coeficiente de *Bhattacharyya*

$$\rho[\widehat{p}(\widehat{\mathbf{y}}_1), \widehat{q}] = \sum_{u=1}^{m} \sqrt{\widehat{p}_u(\widehat{\mathbf{y}}_1)\widehat{q}_u}$$

7 **while** $\rho[\widehat{p}(\widehat{\mathbf{y}}_1), \widehat{q}] < \rho[\widehat{p}(\widehat{\mathbf{y}}_0), \widehat{q}]$
8 $\quad \widehat{\mathbf{y}}_1 \leftarrow \frac{1}{2}(\widehat{\mathbf{y}}_0 + \widehat{\mathbf{y}}_1)$
9 $\quad$ evaluar $\rho[\widehat{p}(\widehat{\mathbf{y}}_1), \widehat{q}]$
10 **endwhile**

11 **if** $\|\widehat{\mathbf{y}}_1 - \widehat{\mathbf{y}}_0\| < \epsilon$
12 $\quad$ parar
13 **else**
14 $\quad \widehat{\mathbf{y}}_0 \leftarrow \widehat{\mathbf{y}}_1$
15 $\quad$ ir al paso 2
16 **endif**

17 Fin

La línea 7 tiene como objetivo evitar problemas numéricos en la maximización llevada a cabo por Mean Shift aunque en la práctica, después de realizar varios experimentos con diferentes tipos de objetos se llegó a la conclusión de que sólo se aplica en el 0.1 por ciento [CRM00] de las veces. El algoritmo de seguimiento en ejecución podría resumirse a la línea 3, donde se computan los pesos asociados a cada iteración, la derivación de una nueva posición en la línea 4 y la 11 donde el criterio de parada se chequea entre cada iteración. Los pasos expuestos en el algoritmo se realizan entre cada par de cuadros consecutivos.

El umbral ϵ utilizado en el paso 10 como criterio de parada es tomado para que $\mathbf{y}_0$ e $\mathbf{y}_1$ caigan en el mismo pixel de la imagen original. Un umbral más pequeño podría ocasionar problemas en la convergencia. Para cumplir con

los requerimientos en tiempo real se establece una cantidad máxima de iteraciones aunque en promedio no se superan las cuatro.

En la implementación del algoritmo se han tomado varias decisiones para cumplir con los requerimientos en tiempo real que son especificadas en el Apéndice A.

6.4.1. Cambios en la escala

Uno de los supuestos que hace el Algoritmo 4, es la actualización cuadro por cuadro del *modelo objetivo* para contemplar el caso de que éste presente alteraciones de algún tipo. Los cambios de escala que sufre el objetivo cuando se aleja o se acerca de la cámara, no son tenidos en cuenta por el algoritmo de seguimiento y por ende se presenta la siguiente propuesta.

La siguiente ecuación[CRM+03] propone actualizar el ancho del bandwidth actual h_{act} de la forma:

$$h_{act} = \tau h_{opt} + (1 - \tau) h_{prev} \tag{6.11}$$

donde $\tau = 0{,}1$ es el valor por defecto, h_{prev} es el bandwidth utilizado en el *frame* anterior y h_{opt} es el mejor bandwidth (según la medida obtenida con el coeficiente de *Bhattacharyya*) entre h_{prev}, $(h_{prev} + \Delta h)$ y $(h_{prev} - \Delta h)$, donde el valor por defecto de Δh es $0{,}1 h_{prev}$.

6.5. Experimentos

Experimento 1. Seguimiento aplicando el Algoritmo de *Mean Shift*.

En este experimento se analiza la performance del seguimiento utilizando *Mean Shift* especificado en el Algoritmo 4 para el seguimiento de un jugador que viste una casaca de color rojo en un partido de fútbol. El video consta de una secuencia de 441 cuadros, de 474 x 358 pixels cada uno, tomados desde una cámara en movimiento. El *modelo objetivo* está representado con un histograma bineado de 16 x 16 siendo R y G (en el espacio de color RGB) las bandas seleccionadas y ponderado por un *kernel* de *Epanechnicov*. El *modelo objetivo* fue elegido manualmente, abarca desde los hombros hasta el pantalón del jugador visto de atrás y tiene un tamaño $(h_x, h_y) = (28, 38)$.

El algoritmo de seguimiento tiene un buen comportamiento frente a deformaciones, oclusiones y difuminaciones del objetivo en cuestión. En la Figura 6.5 se muestran los nueve *frames* más representativos del video y en la Figura 6.6 las distancias (por *frame*) basadas en el coeficiente de *Bhattacharyya* descrita en la ecuación 6.6. En el cuadro de las mediciones, obsevamos un aumento constante de la distancia de *Bhattacharyya* hasta el *frame* 80 donde

Figura 6.5: Seguimiento del jugador de casaca colorda. Los *frames* mostrados son los número 1, 80, 100, 235, 240, 241, 254, 255 y 265 ordenados de izquierda a derecha y de arriba hacia abajo.

se produce una oclusión parcial por parte de un jugador rival produciendo el primer pico significativo. En los *frames* que siguen, el jugador objetivo se separa de su rival como se muestra en el *frame* 100, haciendo que las distancias sean más pequeñas entre el *modelo objetivo* y *modelo candidato*. Luego el objetivo sufre una deformación consecuencia del cambio de perfil respecto a la posición de la cámara haciendo que las distancias vuelvan a incrementarse.

En el *frame* 240 se alcanza un máximo debido a una oclusión que dura un par de *frames*. En éstos, el metodo de *Mean Shift* converge a una posición muy cercana casi sin sufrir un desplazamiento. En los *frames* siguientes, la oclusión deja de existir y el método *Mean Shift* se ajusta nuevamente al objetivo. Es importante notar que la cámara que sigue al jugador, se mueve con él, haciendo que cuando éste reaparezca se encuentre dentro de la zona delimitada por el ancho del *bandwidth* permitiendo ser detectado por *Mean Shift*.

El segundo pico se produce en los *frames* 254 y 255 debido a la pérdida del objetivo, aunque *frames* más tarde se vuelve a encontrar. La secuencia final de valores distancia son altos debido a distorciones y difuminaciones del objetivo. Cambios bruscos en la posición y velocidad del jugador respecto a la cámara, hacen que partes del fondo de la imagen (cesped del estadio), formen parte de la representación del *objetivo candidato*.

Si bien los resultados obtenidos para este ejemplo son satisfactorios, debemos ser concientes de que esto se debe a que en parte se trata con un video bien condicionado. Esto se debe a varios factores, en primer lugar la diferencia de color entre las casacas de los equipos y el fondo son notorias. Segundo, los movimientos del jugador no son bruscos (esto se debe en parte a que la camara se mueve con el jugador). En tercer lugar, después de la oclusión, el objetivo reaparece dentro de la region de interés posibilitando su seguimiento, de otra manera se hubiese perdido. Por último, casi no existen cambios en la profundidad del objetivo a lo largo del tiempo permitiendo obtener buenos resultados con los parámetros propuestos en la sección 6.4.1 (en el siguiente experimento analizaremos este aspecto).

Experimento 2. Cambios en la escala del *target*.

Con el fin de comprobar aquellas situaciones en donde los cambios de escala del objetivo son importantes, se realizó el siguiente experimento con la implementación propuesta en la sección 6.4.1 de este Capítulo. En la Figura 6.7 se muestran seis *frames* de un video en donde un elemento de color uniforme (pelotita de ping pong) se aleja y luego retorna a su posición original respecto de una cámara fija. En una primera instancia, se utilizaron los valores de los parámetros dados por defecto pero los resultados obtenidos no fueron satisfactorios. Existen dos factores importantes que deben ser tenidos

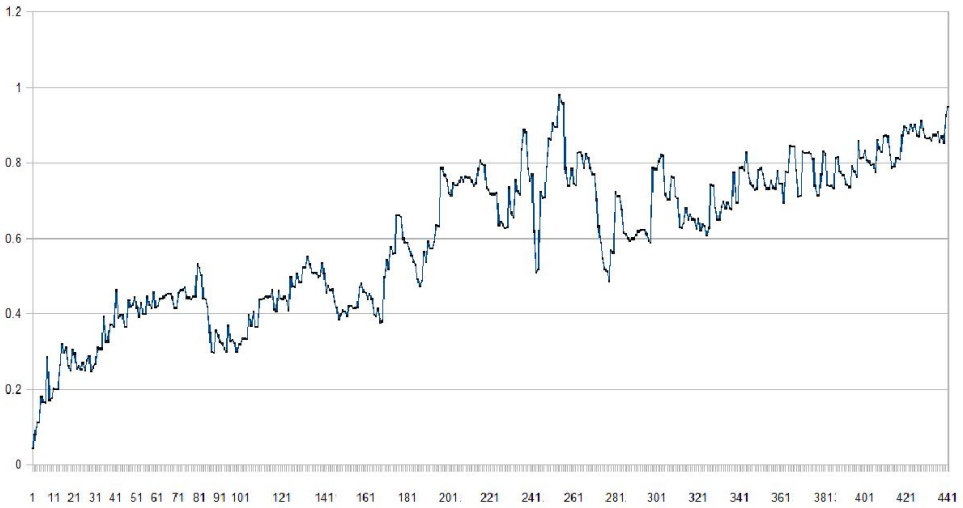

Figura 6.6: Distancias basadas en el coeficiente de *Bhattacharyya*

en cuenta a la hora de implementar el algoritmo que soporta cambios en la dimensión del objeto a seguir.

En primer lugar, según lo propuesto en la sección 6.4.1 el bandwidth debe recalcularse entre cada *frame* de la siguiente manera

$$h_{act} = \tau h_{opt} + (1 - \tau) h_{prev} \tag{6.12}$$

donde el parámetro $\tau = 0{,}1$ sugerido, representa la variación de tamaño entre *frames* consecutivos del objeto. El valor debió cambiarse a $\tau = 0{,}3$ para que funcione adecuadamente en nuestro caso en particular.

En segundo lugar, el criterio de elección de h_{opt} que surge de obtener el mejor *bandwidth* entre el utilizado en la iteración anterior, uno más pequeño y uno más grande debe ser modificado. Cuando nuestro objetivo se trata de un objeto uniforme en color, hay una tendencia a escoger uno más pequeño puesto que el *matching* es sólo por color y los bordes del objetivo en el *frame* actual suelen introducir ruido al histograma de color. Por esta razón es que se dió una ponderación para la elección de un h más grande del 10 por ciento. La elección adecuada de estos parámetros influye en el rendimiento cuando acercamientos y alejamientos del objeto a seguir son importantes de detectar.

Figura 6.7: Zoom de un objeto uniforme. Los *frames* mostrados son 16, 176,339, 600, 695, 800. En los tres primeros *frames* el objeto se aleja respecto de la cámara mientras que en los tres ultimos se acerca. En el cuarto *frame* el objeto se acerca pero el algoritmo todavia no lo ha detectado.

Capítulo 7

Mejora utilizando Filtro de Kalman

En este Capítulo se introducirá una mejora al algoritmo de seguimiento propuesto en el Capítulo 6 aplicando la Teoría del *Filtro de Kalman*[WB95]. A modo de introducción se dará una definición de *procesos estocásticos* para luego presentar el proceso de estados y de observaciones sobre los que se basa la Teoría del *Filtro de Kalman*. Luego se presentará la *ecuación principal de Kalman* la cual determina la estimación del estado y se deducirá cada una de las subecuaciones que la conforman. Seguidamente, se presentará el algoritmo que implementa el *Filtro de Kalman* adaptado al mejoramiento de nuestro algoritmo de seguimiento brindando una predicción de la ubicación del objetivo (*target*) en el siguiente *frame* de video. Por último se realizarán experimentos para comparar los resultados obtenidos con y sin la introducción de esta mejora.

7.1. Procesos estocásticos

Los procesos estocásticos están relacionados con el modelado de sistemas que evolucionan con el tiempo o el espacio de forma no determinística. Situaciones que podrían modelarse con estos procesos son por ejemplo, las fluctuaciones de los valores de la bolsa segundo a segundo, la tasa de mortalidad infantil anual que tiene un país, el tiempo de espera en una cola de pedidos, etcétera.

Un proceso estocástico es una sucesión de variables aleatorias $\{X_t\}, t \in \mathbb{R}$ que evolucionan en el tiempo $t \in T$ donde T representa un rango en los números reales aunque en nuestro caso de estudio trataremos con un dominio discreto. Cada una de la variables aleatorias del proceso tiene su propia distribución de probabilidad, y pueden estar o no correlacionadas entre sí. Por ejemplo, el número de personas en la cola de un banco en un instante dado

entre los segundos dos y cinco, se representa con la variable aleatoria X_t con $t = 1, 2, 3, 4, 5$.

7.2. El Filtro de Kalman

El *Filtro de Kalman*[SHB07] es un método matemático que obtiene de forma recursiva un estimador lineal, insesgado y óptimo (estadísticamente) del estado de un proceso en un instante k, en base a sus observaciones en los instantes $1, ..., k$ y un error en la estimacón asociado. Se dice que es recursivo ya que el filtro se retroalimenta de su salida en el paso anterior para la realización de la nueva estimación.

7.2.1. Proceso de estados y de observaciones

Dado un proceso de estados $\{\mathbf{x}_t\}_{t=0,1,...}$ Gaussiano con media cero donde t representa diferentes instantes discretos de tiempo, se busca una estimación del mismo a partir de una sucesión de observaciones $\{\mathbf{y}_0, ..., \mathbf{y}_t\}$ la cual a su vez tiene asociado ruido Gaussiano de manera que el *proceso de observacines* queda definido como

$$\mathbf{y}_t = H\mathbf{x}_t + \mathbf{z}_t \tag{7.1}$$

donde la matriz H es una combinación lineal del estado $\mathbf{x}_t$ a estimar llamada *matriz de observación* y $\mathbf{z}_t$ representa ruido blanco gaussiano con media cero y varianza σ_z^2. Se supone para simplificar el modelo, que el estado $\mathbf{x}_t$ y el ruido $\mathbf{z}_t$ son independientes de manera que $E(\mathbf{x}_n\mathbf{z}_t') = 0$ para $0 \leq n < t$.

Por otro lado se supone que el proceso de estados es *Markoviano* en el sentido de que sólo se basa en el estado del instante anterior. El estado del sistema está representado por un vector de números reales donde en cada incremento del tiempo de aplica una operación lineal para el cálculo del nuevo estado. De esta forma el *proceso de estados* queda definido de la siguiente forma

$$\mathbf{x}_{t+1} = F_t\mathbf{x}_t + \mathbf{u}_t \tag{7.2}$$

donde F_t es la *matriz de transición* que relaciona el estado previo con el actual y $\mathbf{u}_t$ es un vector ruido Gaussiano con media cero y varianza σ_u^2. Asimismo se supone que ambos ruidos son independientes, es decir que $E(\mathbf{u}_t\mathbf{z}_t') = 0$.

La solución óptima se alcanza minimizando el error cuadrático medio

$$\epsilon_{t+1}^2 = cov(\mathbf{x}_{t+1} - \widehat{\mathbf{x}}_{t+1}) = E[(\mathbf{x}_{t+1} - \widehat{\mathbf{x}}_{t+1})(\mathbf{x}_{t+1} - \widehat{\mathbf{x}}_{t+1})'] \tag{7.3}$$

y para alcanzarlo, la mejor estimación para el estado es

$$\widehat{\mathbf{x}}_{t+1} = E[\mathbf{x}_{t+1}|\mathbf{y}_0, ..., \mathbf{y}_t] \tag{7.4}$$

7.2.2. Ecuaciones del Filtro de Kalman

Sea $\widehat{\mathbf{x}}_{m|n}$ la estimación del estado en el instante m a partir de las observaciones $\mathbf{y}_0, ..., \mathbf{y}_n$, se dice que el Filtro de Kalman estima el estado $\widehat{x}_{t|t}$ de manera tal que la covarianza del error $\epsilon^2_{t|t} = cov(\mathbf{x}_t - \widehat{\mathbf{x}}_{t|t})$ es mínima. El *Filtro de Kalman* consta de una ecuación para la estimación del estado y otra para la estimación de la covarianza del error. Una deducción para el cálculo de ambas puede encontrarse en [LS79].

El algoritmo que implementa el *Filtro de Kalman* consta de cinco ecuaciones que surgen al descomponer las dos antes mencionadas. Éstas están clasificadas en dos fases, la de *predicción* y la de *actualización*. Las ecuaciones en el instante t pertenecientes a la primer fase estiman el estado y la covarianza del error tienendo en cuenta sólo las $t-1$ primeras observaciones, por otro lado las ecuaciones de la fase de actualización son las encargadas de actualizar las estimaciones teniendo en cuenta la nueva información dada por la observación en el instante t.

Fase de predicción

- Predicción de la estimación del estado
  $\widehat{\mathbf{x}}_{t|t-1} = F_t\widehat{\mathbf{x}}_{t-1|t-1}$
- Predicción de la estimación de la covarianza del error
  $\epsilon^2_{t|t-1} = cov(\mathbf{x}_t - \widehat{\mathbf{x}}_{t|t-1})$

Fase de actualización

- Residuo de la observación
  $\widehat{\mathbf{r}}_t = \mathbf{y}_t - H\widehat{\mathbf{x}}_{t|t-1}$
- Covarianza del residuo
  $S_t = cov(\widehat{\mathbf{r}}_t)$
- Matriz de la Ganancia de Kalman
  $G_t =$ mínino valor de $E[(\mathbf{x}_t - \widehat{\mathbf{x}}_{t|t})^2]$
- Actualización de la estimación de estado
  $\widehat{\mathbf{x}}_{t|t} = \widehat{\mathbf{x}}_{t|t-1} + G_t\widehat{\mathbf{r}}_t$
- Actualización de la matriz de covarianza
  $\epsilon^2_{t|t} = cov(\mathbf{x}_t - \widehat{\mathbf{x}}_{t|t})$

7.2.3. Derivaciones de las ecuaciones del Filtro de Kalman

Se deduce la expresión de la predicción en la estimación de la covarianza del error

$$\begin{aligned}\epsilon^2_{t|t-1} &= cov(\mathbf{x}_t - \widehat{\mathbf{x}}_{t|t-1}) \\ &= cov(F_t\mathbf{x}_{t-1} + \mathbf{u}_t - F_t\widehat{\mathbf{x}}_{t-1|t-1}) \\ &= cov(F_t(\mathbf{x}_{t-1} - \widehat{\mathbf{x}}_{t-1|t-1}) + \mathbf{u}_t) \\ &= F_t cov((\mathbf{x}_{t-1} - \widehat{\mathbf{x}}_{t-1|t-1}))F_t' + \sigma_u^2 \\ &= F_t\epsilon^2_{t-1|t-1}F_t' + \sigma_u^2 \end{aligned} \tag{7.5}$$

Desarrollando la expresión de la *covarianza del* $\widehat{\mathbf{r}}_t$ definida

$$\begin{aligned} S_t &= cov(\widehat{\mathbf{r}}_t) \\ &= cov(\widehat{\mathbf{y}}_t - H\widehat{\mathbf{x}}_{t|t-1}) \\ &= cov(H\mathbf{x}_t + \mathbf{z}_t - H\widehat{\mathbf{x}}_{t|t-1}) \\ &= E[(H(x_t - x_{t|t-1}) + \mathbf{z}_t)(H(x_t - x_{t|t-1}) + \mathbf{z}_t)'] \\ &= H\epsilon^2_{t|t-1}H' + E[\mathbf{z}_t\mathbf{z}_t'] \\ &= H\epsilon^2_{t|t-1}H' + \sigma_z^2 \end{aligned} \tag{7.6}$$

Respecto a la ecuación de la *covarianza del error*

$$\begin{aligned}\epsilon^2_{t|t} &= cov(\mathbf{x}_t - \widehat{\mathbf{x}}_{t|t}) \\ &= cov(\mathbf{x}_t - (\widehat{\mathbf{x}}_{t|t-1} + G_t(\mathbf{y}_t - H\widehat{\mathbf{x}}_{t|t-1}))) \\ &= cov(\mathbf{x}_t - (\widehat{\mathbf{x}}_{t|t-1} - G_t(H\mathbf{x}_t + \mathbf{z}_t - H\widehat{\mathbf{x}}_{t|t-1}))) \\ &= cov(\mathbf{x}_t - (\widehat{\mathbf{x}}_{t|t-1} - G_t(H(\mathbf{x}_t - \widehat{\mathbf{x}}_{t|t-1}) + \mathbf{z}_t))) \\ &= cov((I - G_tH)(\mathbf{x}_t - \widehat{\mathbf{x}}_{t|t-1}) - G_t\mathbf{z}_t) \\ &= cov((I - G_tH)(\mathbf{x}_t - \widehat{\mathbf{x}}_{t|t-1})) + cov(G_t\mathbf{z}_t) \\ &= (I - G_tH)cov(\mathbf{x}_t - \widehat{\mathbf{x}}_{t|t-1})(I - G_tH)' + G_t cov(\mathbf{z}_t)G_t' \\ &= (I - G_tH)\epsilon^2_{t|t-1}(I - G_tH)' + G_t\sigma_z^2G_t' \end{aligned} \tag{7.7}$$

Para obtener la expresión de la ecuación de la *Ganancia de Kalman*, buscamos minimizar el error cuadrático medio, es decir $E[(x_t - \widehat{x}_{t|t})^2]$. Ésto es equivalente a minimizar la traza de la matriz de covarianza del error $\epsilon_{t|t}$. Expandiendo la expresión de covarianza del error obtenida en 7.7

$$\begin{aligned}\epsilon^2_{t|t} &= (I - G_tH)\epsilon^2_{t|t-1}(I - G_tH)' + G_t\sigma_u^2G_t' \\ &= \epsilon^2_{t|t-1} - G_tH\epsilon^2_{t|t-1} - \epsilon^2_{t|t-1}H'G_t' + G_tH\epsilon^2_{t|t-1}H'G_t' + G_t\sigma_u^2G_t' \\ &= \epsilon^2_{t|t-1} - G_tH\epsilon^2_{t|t-1} - \epsilon^2_{t|t-1}H'G_t' + G_tS_tG_t' \end{aligned} \tag{7.8}$$

Derivamos $\epsilon^2_{t|t}$ respecto de G_t para encontrar el mínimo

$$\frac{\delta tr(\epsilon^2_{t|t})}{\delta G_t} = -2(H\epsilon^2_{t|t-1})' + 2G_tS_t = \mathbf{0} \tag{7.9}$$

despejando G_t obtenemos

$$\begin{aligned} G_tS_t = (H\epsilon^2_{t|t-1})' = \epsilon^2_{t|t-1}H' \\ G_t = \epsilon^2_{t|t-1}H'S_t^{-1} \end{aligned} \tag{7.10}$$

Para obtener una expresión computacionalmente menos costosa de la ecuación de la covarianza del error en 7.7, se multiplica la ecuación de la Ganancia de Kalman (7.10) por S_tG_t'

$$G_tS_tG_t' = \epsilon^2_{t|t-1}H'S_t^{-1}S_tG_t' = \epsilon^2_{t|t-1}H'G_t' \tag{7.11}$$

Reemplazando en la fórmula 7.8,lo obtenido en 7.11 se llega a que

$$\begin{aligned} \epsilon^2_{t|t} &= \epsilon^2_{t|t-1} - G_tH\epsilon^2_{t|t-1} - \epsilon^2_{t|t-1}H'G_t' + G_tS_tG_t' \\ &= \epsilon^2_{t|t-1} - G_tH\epsilon^2_{t|t-1} - \epsilon^2_{t|t-1}H'G_t' + \epsilon^2_{t|t-1}H'G_t' \\ &= \epsilon^2_{t|t-1} - G_tH\epsilon^2_{t|t-1} \\ &= (I - G_tH)\epsilon^2_{t|t-1} \end{aligned} \tag{7.12}$$

7.2.4. Algoritmo del Filtro de Kalman

A continuación se presenta el *Algoritmo de Kalman* donde las fases de predicción y actualización antes mencionadas están claramente marcadas. Además se agrega un tercer grupo compuesto por dos ecuaciones para la inicialización de la estimación del estado y la covarianza de error.

Algoritmo 5 Filtro de Kalman

KalmanFilter()
1 /* inicialización de la variable de estado */
2 $\widehat{\mathbf{x}}_{0|0} = E(\mathbf{x}_0)$
3 /* inicialización de la covarianza del error */
4 $\epsilon_{0|0} = E[(\mathbf{x}_0 - E(\mathbf{x}_0))(\mathbf{x}_0 - E(\mathbf{x}_0))']$
5
6 /* predicción del estado */
7 $\widehat{\mathbf{x}}_{t|t-1} = F_t \widehat{\mathbf{x}}_{t-1|t-1}$
8 /* predicción de la covarianza del error */
9 $\epsilon^2_{t|t-1} = F_t \epsilon^2_{t-1|t-1} F'_t + \sigma^2_u$
10
11 /* cálculo de la matriz de ganancia de Kalman */
12 $G_t = \epsilon_{t|t-1} H' (H \epsilon_{t|t-1} H' + \sigma^2_z)^{-1}$
13 /* actualización de la estimación del estado actual */
14 $\widehat{\mathbf{x}}_{t|t} = \widehat{\mathbf{x}}_{t|t-1} + G_t(\mathbf{y}_t - H\widehat{\mathbf{x}}_{t|t-1})$
15 /* actualización de la covarianza del error */
16 $\epsilon_{t|t} = (I - G_t H)\epsilon^2_{t|t-1}$
17 ir al paso 6

7.3. Algoritmo de Kalman para seguimiento

Suponiendo que el movimiento de un objeto a seguir a través de una secuencia de cuadros (o *frames*) es continuo, el Algoritmo de *Kalman* predice la posición del objetivo (o *target*) en el instante de tiempo siguiente teniendo en cuenta su trayectoria determinada por sus posiciones pasadas. Para ello, se analizan las diferencias en distancia entre las posiciones pasadas del objetivo entre cuadros consecutivos.

Se supone que el intervalo de tiempo entre cuadro y cuadro es lo suficientemente pequeño como para considerar que el movimiento de nuestro objetivo (que puede estar representado por un punto, borde o región) puede ser aproximado por medio de una función lineal con un margen de error pequeño.

El Algoritmo de *Kalman* es un procedimiento que brindará la posición y el error asociado a un objetivo en movimiento en el siguiente cuadro. En otras palabras, permitirá realizar la búsqueda dentro de un área reducida en el siguiente cuadro con un cierto grado de confianza.

En este contexto, el *Algoritmo de Kalman* realiza una predicción de la posición del objetivo en el próximo cuadro, que utilizará el Algoritmo de *Mean Shift* como punto de partida. De esta manera se parte de una posición más

cercana a la exacta, reduciendo la cantidad de iteraciones necesesarias para alcanzar el mejor *matching* entre el *modelo objetivo* y *objetivo candidato*. La posición a la que converge *Mean Shift*, retroalimentará, como una nueva observación, al *Algoritmo de Kalman* para la estimación de una nueva posición en el siguiente cuadro.

7.3.1. Interpretación de las componentes de Kalman para Seguimiento

A continuación se decribe los valores de las matrices del *Algoritmo de Kalman*, para lograr una predicción de la posición del objetivo en el algoritmo de *tracking*.

Se supone que el objetivo está enmarcado por un *bounding box* de ancho h_x y alto h_y, y su posición por el centroide $(\frac{h_x}{2}, \frac{h_y}{2})$. Se denota al vector de estado $\mathbf{x}_t = [x_t, y_t, 1]'$ donde x_t e y_t denotan las coordenadas espaciales del objetivo y a la matriz de transición F_t definida como:

$$F_t = \begin{pmatrix} 1 & 0 & dx_{t,t+1} \\ 0 & 1 & dy_{t,t+1} \\ 0 & 0 & 1 \end{pmatrix}$$

donde $dx_{t,t+1}$ y $dy_{t,t+1}$ denotan las translaciones horizontal y vertical del objetivo entre el instante de tiempo t y $t+1$. El vector $\mathbf{u}_t = [u_{t_x}, u_{t_y}, 1]'$ representa el ruido del sistema y su matriz de covarianza está dada por

$$\sigma_u^2 = \begin{pmatrix} \sigma_u(x) & 0 & 0 \\ 0 & \sigma_u(y) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

donde $\sigma_u(x) = h_x$ y $\sigma_u(y) = h_y$, por lo que se supone que las perturbaciones del centroide del objetivo están delimintadas por la *bounding box* que lo enmarca.

El vector de observación se compone de la siguiente manera, $\mathbf{y}_t = [x_{t+1}^m, y_{t+1}^m]'$ donde x_t^m e y_t^m son las coordenadas determinadas por el Algoritmo de *Mean Shift* que difieren de x_t e y_t por la presencia de ruido $\mathbf{z}_t$. La relación entre la observación $\mathbf{y}_t$ y el estado $\mathbf{x}_t$ está dado por la matriz de observación definida como

$$H = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

y el ruido de la observación $\mathbf{z}_t = [z_t(x), z_t(y)]'$ cuya matriz de covarianza es

$$\sigma_z^2 = \begin{pmatrix} \sigma_z^2(x) & 0 \\ 0 & \sigma_z^2(y) \end{pmatrix}$$

donde $\sigma_z^2(x) = h_x$ y $\sigma_z^2(y) = h_y$. Lo que significa que la *bounding box* que contiene el objetivo se solapa con la de *Kalman*.

Sólo resta entonces detallar cómo se genera de forma dinámica los valores de $dx_{t,t+1}$ y $dy_{t,t+1}$. Del Algoritmo 4 se obtienen dos datos fundamentales

1. La posición $\mathbf{y}$ calculada del *target*
2. La medida de distancia $d(\mathbf{y})$ basada en el coeficiente de *Bhattacharrya* vistan en la fórmula 6.6 del Capítulo 6, entre el *modelo objetivo* y el *objetivo candidato* centrado en $\mathbf{y}$

A partir de estos datos, se busca una métrica a que brinde información relacionada con la confiabilidad de las posiciones obtenidas a partir de las medidas de distancia suponiendo que para resultados pequeños de d obtenemos buenas estimaciones y viceversa. A fines prácticos buscamos una función f decreciente, tal que si la distancia d encontrada es pequeña el resultado sea cercano a uno y en caso contrario cercano a cero. La métrica a queda definida como

$$a(\mathbf{y}) = f(d(\mathbf{y})) \tag{7.13}$$

El parámetro $\mathbf{d}_{t,t+1} = [dx_{t,t+1}, dy_{t,t+1}]'$ es actualizado de la siguiente manera

$$\mathbf{d}_{t+1,t+2} = (1-a)\mathbf{d}_{t,t+1} + a(\hat{\mathbf{x}}_{t+1} - \hat{\mathbf{x}}_t) \tag{7.14}$$

donde $\hat{\mathbf{x}}_t$ y $\hat{\mathbf{x}}_{t+1}$ son las estimaciones realizadas de las posiciones del objeto en el instante t y $t+1$. Notar que la estimación $\hat{\mathbf{x}}_{t+1}$ contribuye a la actualización cuando la distancia entre el objetivo y el candidato es pequeña, es decir $a(\mathbf{y}) \to 1$. Caso contrario, se considera que ocurrió una oclusión y por tanto $a(\mathbf{y}) \to 0$ haciendo que la matriz F_{t+2} permanezca casi sin inalteraciones.

7.3.2. Seguimiento para un caso particular

A continuación se da un ejemplo para clarificar lo antes expuesto. Los *target* en la parte superior del dibujo de la Figura 7.1, representan al objetivo a lo largo del tiempo $t = 1, ..., 5$ y $d_1, ..., d_4$ los diferentes desplazamientos entre los instantes $t_1, ..., t_5$. En la fila central, se muestra el resultado del tracking utilizando sólo *Mean Shift* (MS) y en la inferior los resultados utilizando *Mean Shift* con la incorporación de la mejora utilizando el *Kalman* (MSK). Las líneas discontinuas muestran los corrimientos realizados por *Mean Shift* y las continuas a las predicciones obtenidas por *Kalman*. En el instante de tiempo cuatro ocurre una oclusión y el objeto desaparece de la escena. Se va a describir el comportanmiento de este escenario para cada uno de los algoritmos.

- En un principio, la matriz F_1 tiene sus componentes $dx_{1,2} = 0$ y $dy_{1,2} = 0$. En el instante dos, la localización del objetivo es exitosa haciendo que $a(y) \to 1$ y en consecuencia, la matriz de estado F_2 tiene sus valores $dx_{2,3} = d_1$ y $dy_{2,3} = 0$.

- En el instante $t = 3$, la posición inicial y_2 es diferente para cada uno de los algoritmos ya que existe una predicción, producto de haber utilizado *Kalman*. Ésto hace que las iteraciones necesarias por el procedimiento *Mean Shift* en MSK sean menos que en MS, obteniendo un mejor desempeño. La matriz de estado F_3 estará dada por $dx_{3,4} = d_2$ y $dy_{3,4} = 0$.

- En el siguiente instante $t = 4$, se produce una oclusión completa. El Algoritmo MS no encuetra el objetivo perdiendo así la trayectoria del objetivo. En el caso de MSK, la distancia d entre el *modelo objetivo* y *objetivo candidato* será grande haciendo que $a(y) \to 0$, la predicción que en este caso es la distancia recorrida entre el instante dos y tres, será entonces lo único que cuente. La matriz de estado $F_4 = F_3$ no sufrirá alteraciones según la definición dada en 7.14.

- Por último, en el instante t igual a cinco, debido a que el valor de $dx_{3,4}$ es igual a la distancia del último corrimiento después de una oclusión, se produce una estimación que excede en distancia a la trayactoria del objetivo, aunque luego ésta es corregida por el procedimiento de *Mean Shift*.

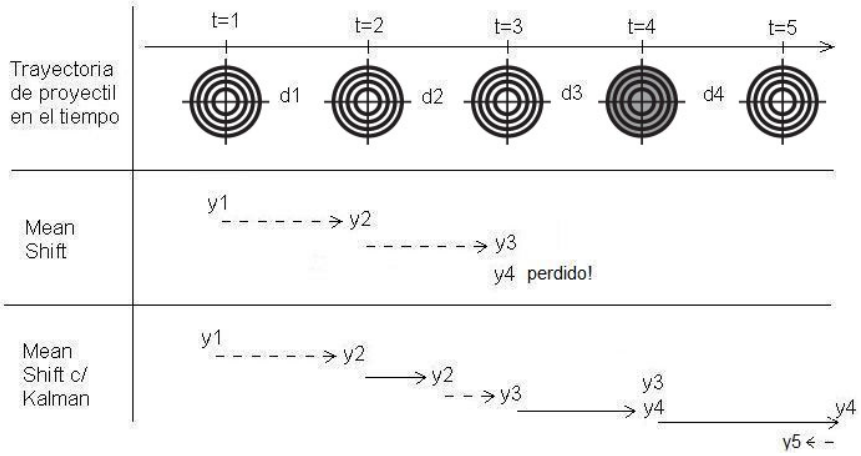

Figura 7.1: Los desplazamientos d_1, d_2, d_3, d_4 son los desplazamientos del objetivo entre pares de *frames* consecutivos. La primer fila muestra los corrimientos utilizados utilizando sólo *Mean Shift*, mientras que la segunda fila aquellos realizados por *Mean Shift* con la predicción de *Kalman*.

7.3.3. Algoritmo de *Mean Shift* utilizando *Kalman*

A continuación se presenta el Algoritmo de *Mean Shift* con la mejora del *Filtro de Kalman* para el tratamiento de oclusiones.

Algoritmo 6 Mean Shift con Filtro de Kalman

TRACKING MSK(Video v)

Inicialización de la estimación del estado y la covarianza del error
$\hat{\mathbf{x}} \leftarrow$ ubicación inicial del objeto
$\epsilon^2 = 0_{3x3}$

Inicialización de matrices de observación, transición y ruidos blanco
$H = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, F = I_{3x3}, \sigma_u^2 = \begin{pmatrix} h_x & 0 & 0 \\ 0 & h_y & 0 \\ 0 & 0 & 0 \end{pmatrix}, \sigma_z^2 = \begin{pmatrix} h_x & 0 \\ 0 & h_y \end{pmatrix}$

Computar el histograma ponderado q del *modelo objetivo* definido en 6.4

Para cada frame en v

Ecuaciones de predicción
$\widehat{\mathbf{x}} = F\widehat{\mathbf{x}}$
$\epsilon^2 = F\epsilon^2 F' + \sigma_u^2$

Computar la nueva posición $\mathbf{y}$, $p(\mathbf{y})$ y la distancia entre q y p usando el
Algoritmo de *Mean Shift* visto en 4 del Capítulo 6

Ecuaciones de actualzación
$G = \epsilon^2 H'(H\epsilon^2 H' + \sigma_z^2)^{-1}$
$\widehat{\mathbf{x}} = \widehat{\mathbf{x}} + G(\mathbf{y} - H\widehat{\mathbf{x}})$
$\epsilon^2 = (I - GH)\epsilon^2$

Actualizar los elementos de F basado en la ecuación 7.3.1

Fin Para cada

Fin

7.3.4. Experimentos

Experimento 1. Seguimiento con oclusión.

En este primer experimento se analiza un video que muestra la oclusión de una pelota de ping pong de recorrido horizontal (de izquierda a derecha) que pasa por detrás de un pomo en un lapso cuatro *frames* y a velocidad constante. En la Figura 7.2 y la Figura 7.3 se muestran los resultados obtenidos

utilizando el algoritmo de *tracking* sin y con la mejora de *Kalman*, respectivamente. El video sobre el cual se realizó la prueba consta de 136 *frames*.

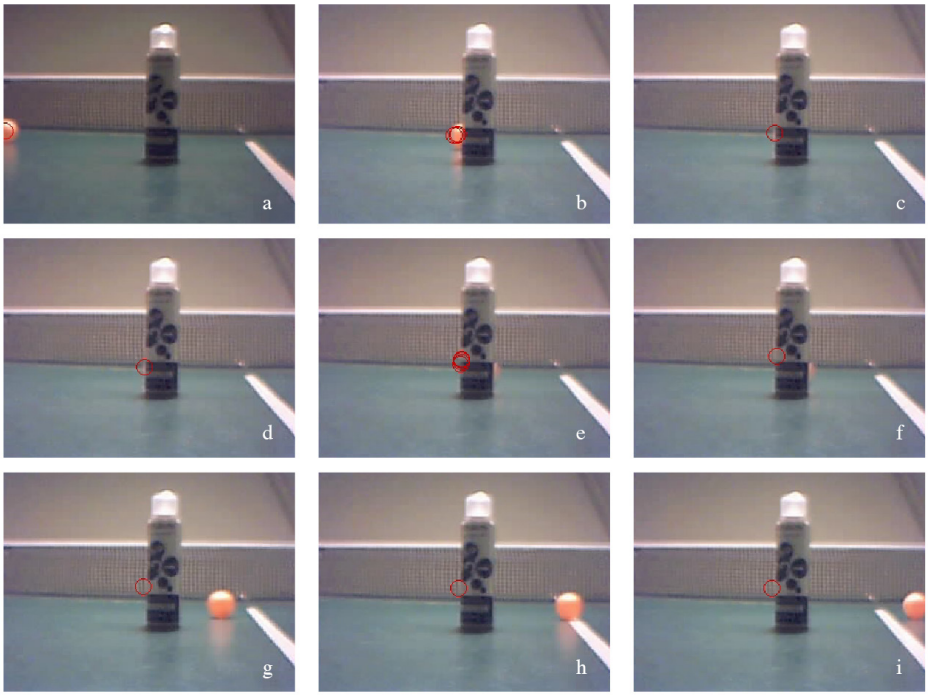

Figura 7.2: Oclusión sin utilizar *Kalman*. Los *frames* mostrados son 10, 27, 47, 48, 49, 50, 65, 80 y 128 ordenados de izquierda a derecha de arriba hacia abajo.

En el *frame* 43 se produce la primera oclusión parcial de la pelota, en el 47 la oclusion ya es total, en el *frame* 51 la pelota vuelve a aparecer en escena y a partir del 55 la oclusión deja de existir.

El seguimiento de la pelota utilizando el Algoritmo de *Tracking* con y sin *Kalman* es satisfactorio hasta el *frame* 47 que es cuando el objetivo sufre una oclusión total. En los cuatro *frames* siguientes, durante la oclusión total, la métrica basada en el coeficiente de *Bhattacharyya* crece de forma repentina y el método de *Mean Shift* pierde el objetivo. Con la mejora introducida por *Kalman*, se realiza una predicción de la ubicación del objetivo en base a los desplazamientos en los *frames* anteriores, posibilitando que pueda ser capturado nuevamente por el procedimiento *Mean Shift* cuando termina la oclusión. En la Figura 7.4 se muestran las distancias basadas en el coeficiente de *Bhattacharyya* entre el *modelo objetivo* y el *objetivo candidato* en cada uno de los *frames* del video con y sin *Kalman*. Las distancias resultado hasta el *frame* 43 sin *Kalman*, presentan un mejor desempeño cuando se aplica

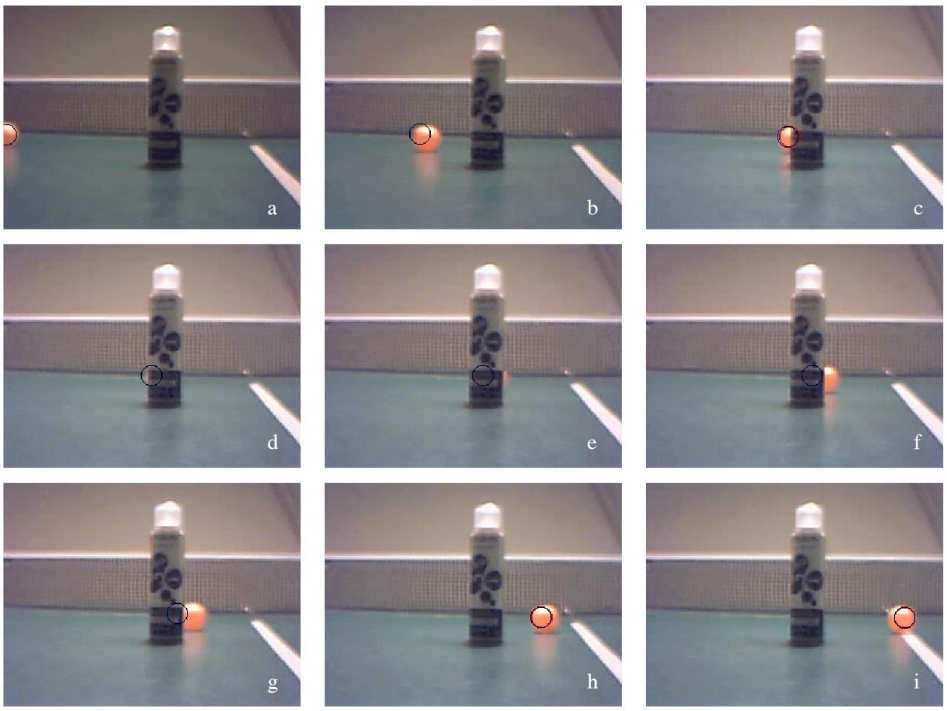

Figura 7.3: Oclusión utilizando *Kalman*. Los *frames* mostrados son 1,27,42,44,48,51,53,65 y 80 ordenados de izquierda a derecha y de arriba hacia abajo

Kalman. Esto se debe a que las predicciones de *Kalman* necesitan de varios *frames* (o iteraciones) para ajustar las posiciones exactas. A partir del *frame* 43, las distancias crecen rapidamente debido a que el objetivo sufre una oclusión hasta el *frame* 55 que es cuando ésta finaliza. En este intervalo, el algoritmo sin *Kalman* se desplaza hacia la posición que más se asemeja a las caraterísticas de color del objetivo, mientras que el que utiliza *Kalman* lo hace en base a las predicciones dadas por el filtro, obteniendo en esos *frames* distancias más grandes. Del *frame* 55 en adelante las distancias utilizando *Kalman* bajan debido a que se detecta nuevamente el objetivo mientras que sin *Kalman* las distancias permanecen altas ya que el objetivo se pierde.

Experimento 2. Oclusión por un objeto de similares características.

En este segundo experimento se prueba la oclusión de un objeto por otro cuyo histograma de color es el mismo, sin y con la mejora introducida por el predictor de *Kalman*. Ambos resultados se muestran en la Figuras 7.5 y 7.6 en los *frames* 5, 42, 65, 72, 103 y 141. El video muestra la trayectoria de dos pelotitas idénticas que se mueven en sentidos opuestos y se cruzan en el

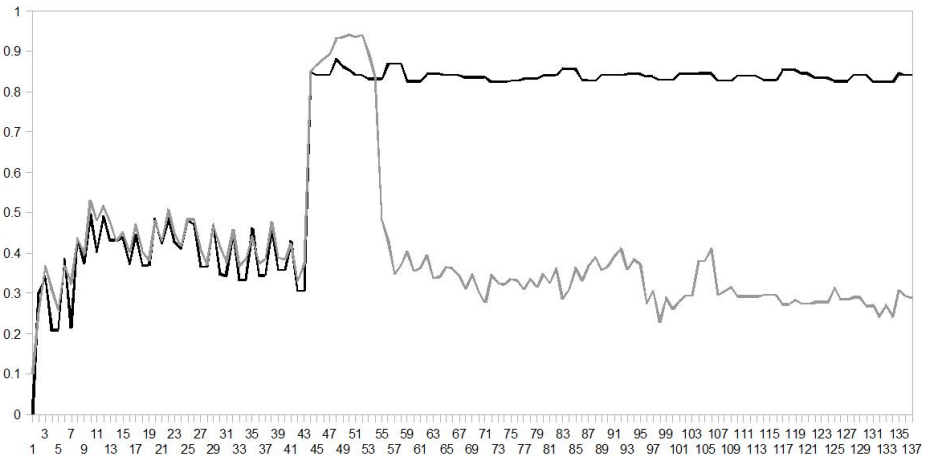

Figura 7.4: Distancia bajo la métrica de *Bhattacharyya* entre el *modelo objetivo* y *objetivo candidato* a lo largo de la secuencia de *frames*

centro de la escena respecto a una cámara fija. La pelotita a la que se desea aplicar un seguimiento es la que va de derecha a izquierda.

En el *frame* 65 se produce una oclusión parcial de la pelotita a seguir por otra

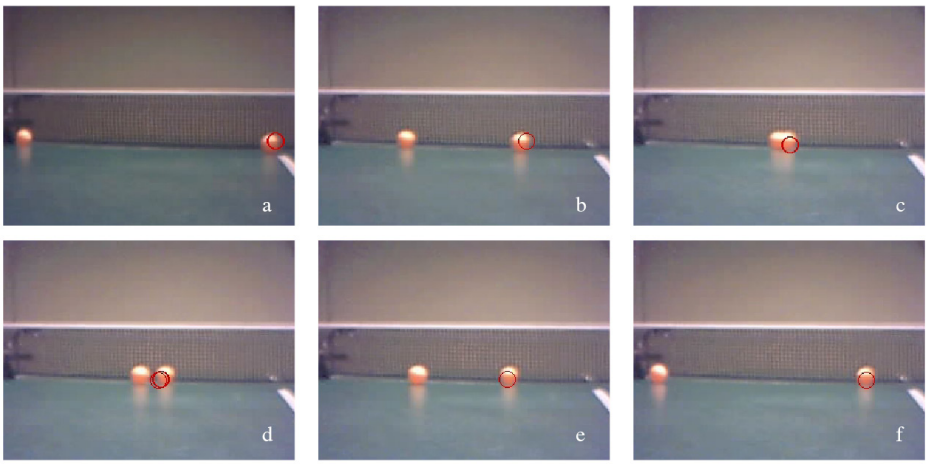

Figura 7.5: Oclusión de una pelotita de ping pong por otra de las mismas características sin utilizar la mejora introducida por *Kalman*.

cuya trayectoria tiene sentido opuesto. Desde este *frame* hasta el número 72 se produce el cruce de ambas. Dentro de este intervalo el algoritmo de tracking con y sin *Kalman* operan de manera diferente. En el *frame* 65 la pelotita objetivo se encuentra a la derecha mientras que en el 72 es la de la izquierda. Al no haber una predicción respecto de la trayectoria del objetivo, la pelota

a seguir se confunde con la otra interpretando que ésta se frena y vuelve en sentido contrario perdiendo el objetivo inicial. Cuando se aplica *Kalman* esto no sucede puesto que en el instante en que se produce la oclusión, *Kalman* estima la próxima posición según la trayectoria y obteniendo en consecuencia un seguimiento satisfactorio.

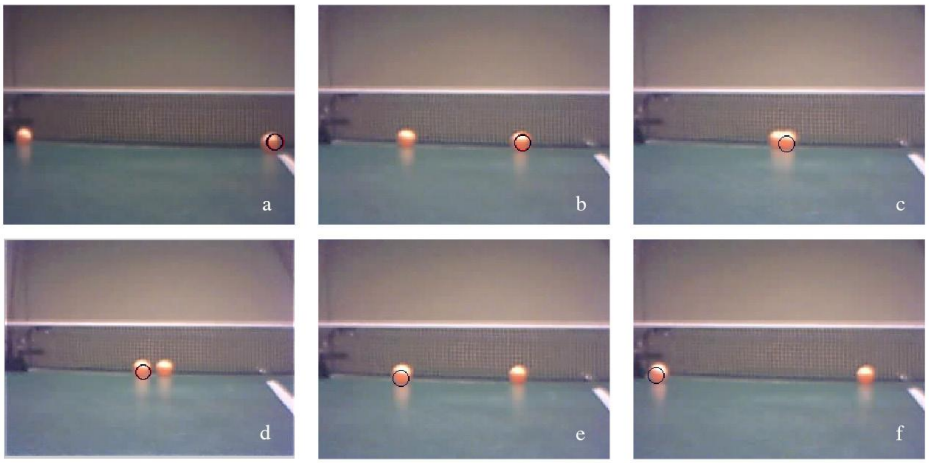

Figura 7.6: Oclusión de una pelotita de ping pong por otra de las mismas características utilizando la mejora introducida por *Kalman*.

Limitaciones

Un supuesto importante que debe cumplirse para realizar un tracking efectivo requiere que la distancia entre el mismo objetivo entre *frames* consecutivos sea pequeño. Si bien en teoria, *Kalman* puede predecir un desplazamiento grande, necesita que los desplazamientos en los *frames* anteriores también lo sean y en el mismo sentido. Es decir que no es robusto a movimientos bruscos, repentinos y en donde haya un cambio de sentido. Por otro lado, si la oclusión dura varios *frames* y en este intervalo el objetivo cambia su dirección o se produce una aceleración, *Kalman* no podrá hacer nada al respecto.

Para obtener buenos resultados en tiempo real, la velocidad con la que se mueve el objetivo no sólo depende de bajos tiempos de respuesta en el procesamiento por *frame* sino también de características propias del dispositivo de entrada con el que se capta el video, como son la cantidad de *frames* por segundo y/o la calidad con que son tomadas cada uno de ellos. En muchos casos se ha notado que ante un movimiento repentino, aparece el alo del objeto o desplazamientos demasiado grandes entre *frames* provocando la pérdida del objetivo o *target*.

Capítulo 8

Conclusiones

En este trabajo se ha realizado un estudio teórico de la técnica de *Mean Shift* y sus propiedades. A pesar de que su fundamentación matemática no es sencilla de entender, su aplicación práctica es intuitiva y fácil de realizar. A partir de los estudios realizados por Comaniciu, Ramesh y Meer se han encontrado aplicaciones para algoritmos de agrupamiento, segmentación y seguimiento.

En lo que respecta al agrupamiento (*clustering*), son varios los parámetros que influyen en el desempeño del algoritmo. La elección del tamaño del *bandwidth* influye en la calidad de los resultados, la cual la hace una tarea no trivial. Otros parámetros de entrada necesarios son *minMembers* y *lowRegion*, que determinan la mínima cantidad de puntos que debe contener la hiperesfera centrada en el punto de partida usada por *Mean Shift* y la cantidad de puntos en una región entre dos clusters para ser interpretadas como una depresión en la función de densidad estimada asociada a la muestra. Como vimos en la sección resultados 4.4 del capítulo 4, cuanto menores son los valores del *bandwidth*, *minMembers* y *lowRegion* la cantidad de centros de *clusters* encontradas será mayor, obteniendo mayor grado de detalle pero al mismo tiempo menor tolerancia al ruido de la muestra en el espacio de características. El desempeño del algoritmo depende fuertemente de esta elección y de las características de los datos de entrada.

El *clustering* tiene varias aplicaciones aunque si es utilizado para segmentar una imagen no se obtienen buenos resultados ya que no se tiene en cuenta la distribución espacial de los píxeles. Para ello se propuso un algoritmo de segmentación en el dominio empalmado que tiene en cuenta no sólo la carateristica de cada píxel sino su distribución espacial en la imagen permitiendo que pequeñas variaciones en color no repercutan en falsos contornos. En contra partida, el orden de complejidad se ve agrabado puesto que la dimensionalidad de los puntos/píxeles de la muestra/imagen aumenta en dos (se tiene en cuenta la posición horizontal y vertical) y el procedimiento de *Mean Shift* es aplicado por cada pixel para su clasificación.

El tracking utilizando *Mean Shift* presenta en general buenos resutlados para el seguimiento de objetos en video. Cuenta con varios parámetros que pueden ser manejados de manera que pueden ser aplicados para el seguimiento de objetos otorgando flexibilidad al método. En el agregado que se aplica para detectar cambios en el tamaño del *modelo objetivo*, los valores de los parámetros que determinan la magnitud en los cambios de tamaño entre cuadro y cuadro, y la elección para determinar un crecimiento o decrecimiento, son vitales para un buen desempeño.

También se ha introducido una mejora utilizando la teoría del *Filtro de Kalman* para hacer robusta la aplicación a oclusiones parciales y totales obteniendo resultados satisfactorios pero con algunas limitaciones como la duración de dicha oclusión y cambios bruscos en la trayectoria y velocidad del objetivo o *target*. Sin embargo, el desempeño de la aplicación está relacionado a la correcta elección de los parametros utilizados y técnicas de mejoramiento. Es por eso que para cada contexto en particular se requerirá siempre de pruebas a priori con el fin de poder ajustar los parámetros adecuadamente.

8.1. Mejoras futuras

Una de las tareas pendientes en este trabajo es la implementación de las técnicas de segmentación y tracking utilizando el procedimiento de *Mean Shift* en un lenguaje de bajo nivel para evaluar el tiempo de respuesta en tiempo de ejecución. El código desarrollado en *MATLAB* es un lenguaje interpretado y adecuado para probar la lógica del funcionamiento ya que cuenta con herramientas que facilitan la programación. Pero un lenguaje compilado como *C++* puede alcanzar un desempeño hasta 200 veces más rápido que *MATLAB* para una misma aplicación.

Respecto al *tracking* propuesto, además de la mejora introducida utilizando el predictor de Kalman, existen una gran cantidad de alternativas para hacer más robusto el seguimiento de objetos. En el trabajo de Comaniciu y Meers[CRM+03] se proponen tres mejoras. La primera está relacionada con la robustez de la técnica respecto a los cambios que el fondo de la escena pueda sufrir, el segundo a los cambios de escala del objeto a seguir en el transcurso del tiempo y el tercero a un tracking específico para caras de personas.

La primer mejora consta en cambiar los histogramas ponderados del *modelo objetivo* y *objetivo candidato* de forma tal que las ponderaciones se vean afectadas por la características y cambios del fondo de la escena repecto al objetivo a seguir. Los detalles para su implementación pueden encontrarse en [CRM+03].

Respecto a los cambios de escala que sufre el objetivo, si bien propone una solución en la sección 6.4.1 existen varias cuestiones de contexto a tener en

cuenta la cuales vimos en los resultados. Un estudio más profundo acerca de éste tema es el realizado por Collins [Car03] en donde se presenta un nuevo *kernel* y se generaliza el algoritmo de *Mean Shift* para permitir pesos negativos asociados a los puntos o píxeles de la imagen.

El *tracking* aplicado a caras es una tarea importante en el área de investigación para el desarrollo de interfaces de percepción de humanos. En [Bra98] publicado por Gary R. Bradski, se realiza una modificación al algoritmo de *Mean Shift* para hacer frente a los cambios dinámicos en las distribución de probabilidad de color producidos entre *frames* consecutivos. El nuevo algoritmo propuesto es presentado con el nombre de *Continuosly Adaptative Mean Shift* (*CAMSHIFT*) el cual es robusto al ruido y otras distorciones. En la actualidad, *CAMSHIFT* es una herramienta muy usada para la interfaz hombre-computador en juegos de computadoras y/o gráficos 3D.

Apéndice A

Implementación de Mean Shift Tracking

En este apéndice se detallarán las decisiones de implementación para cumplir con los requerimientos de una aplicación en tiempo real y se describirán los métodos principales utilizados en cada uno de los puntos del algoritmo básico (sin mejoras) para un mejor entendimiento. La implementación fue hecha en *MATLAB* 2010.

Para la representación del *modelo objetivo* y *objetivo candidato* se eligió un *espacio de características* en coordenadas cromáticas. La transformación descrita en A.1 posee varias virtudes además del hecho de permitir reducir los colores a dos.

$$r = \frac{R}{R+G+B}, g = \frac{G}{R+G+B}, b = \frac{B}{R+G+B} = 1 - r - g \qquad \text{(A.1)}$$

En este caso, b está en función de los otros dos colores, aunque ésto bien podría ocurrir con r o g. Las coordenadas cromáticas son más independientes de la iluminación y aumentan la discriminación entre colores. Este *espacio de características* nos permite generar histogramas 2D para dos coordenadas rgb con *bines* de 16 x 16.

Para la representación de los modelos se escogió el *kernel de Epanechnikov* debido a que es sombra de uno *Uniforme* (ambos vistos en 3.2.2) con un perfil constante $g(x)$ y fácil de calcular

$$\widehat{\mathbf{y}}_1 = \frac{\sum_{i=1}^{m} \mathbf{x}_i w_i}{\sum_{i=1}^{m} w_i} \qquad \text{(A.2)}$$

Para obtener la representación del *modelo objetivo* y su ubicación se utiliza la función *recorte* la cual permite al usuario marcar el objetivo a seguir con el *mouse*, luego se llama la función *histograma* que es quien genera el histograma ponderado y a su vez llama a *maskEpanech* encargado de generar

una máscara de pesos según su distancia euclidiana al origen de un ancho y alto igual al objetivo recortado. Dado que el kernel es el mismo a través de toda la implementación, se calcula una única vez y al principio.

Una vez que contamos con las representaciones del modelo objetivo y candidato calculamos el coeficiente de *Bhattacharyya* el cual queda determinado por su definición matemática y se encuentra implementado en la función *bhatCoef*.

El cálculo de las ponderaciones $\{w_i\}$ se lleva a cabo desestimando aquellos valores del histograma del objetivo candidato cercanos o iguales a cero para evitar problemas numéricos en una división por cero.

El paso 4 del algoritmo, donde se aplica *Mean Shift* para disminuir la distancia entre el modelo candidato y objetivo en términos del coeficiente de *Bhattacharyya* se encuentra implementado en *meanshift.m*. Notar que lo que retorna esta función es un corrimiento respecto al origen por lo tanto la nueva posición estimada $\mathbf{y}_1$ resulta de sumar al punto original $\mathbf{y}_0$ lo retornado por *Mean Shift*.

En el paso 6, se chequea la magnitud del paso, en caso de ser demasiado largo se parte a la mitad. Después de cada partición, se determina si el resultado es distinto de $\mathbf{y}_0$ lo cual es factible en una aritmética finita. En tal caso se sale del loop.

En el último paso, se cuenta con la nueva posición $\mathbf{y}_1$ y la anterior $\mathbf{y}_0$, el valor de epsilon que determina el umbral de corrimiento entre un *frame* y otro es de 3. Por otro lado para cumplir con requerimientos en tiempo real, se incrementa un contador que determina la cantidad de veces en que hemos realizado *Mean Shift* entre un *frame* y otro, en este caso si se supera un umbral de 10 veces se sale del ciclo.

Bibliografía

[Bra98] Gary R. Bradski. Computer vision face tracking for use in a perceptual user interface, 1998.

[Car03] Robert Collins Carnegie. Mean-shift blob tracking through scale space, 2003.

[Che95] Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE Trans. Pattern Anal. Mach. Intell.*, 17(8):790–799, 1995.

[CM98] Dorin Comaniciu and Peter Meer. Distribution free decomposition of multivariate data. *Pattern Analysis and Applications*, 2:22–30, 1998.

[CM99] Dorin Comaniciu and Peter Meer. Mean shift analysis and applications, 1999.

[CMM02] Dorin Comaniciu, Peter Meer, and Senior Member. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:603–619, 2002.

[CRH01] Yunqiang Chen, Yong Rui, and Thomas S. Huang. Jpdaf based hmm for real-time contour tracking. pages 543–550, 2001.

[CRM00] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Real-time tracking of non-rigid objects using mean shift. pages 142–149, 2000.

[CRM01] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. The variable bandwidth mean shift and data-driven scale selection. In *in Proc. 8th Intl. Conf. on Computer Vision*, pages 438–445, 2001.

[CRM+03] Dorin Comaniciu, Visvanathan Ramesh, Peter Meer, Senior Member, and Senior Member. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25:564–577, 2003.

[DH73] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Willey & Sons, New Yotk, 1973.

[FH75] Keinozuke Fukunaga and David Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE transactions on information theory*, IT-21(1):32–40, 1975.

[Fuk90] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, 2 edition, 1990.

[LS79] H.J. Larson and B.O. Shubert. *Probabilistic models in engineering sciences*. Number v. 1 in Probabilistic Models in Engineering Sciences. Wiley, 1979.

[SHB07] Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image Processing, Analysis, and Machine Vision*. Thomson-Engineering, 2007.

[WB95] Greg Welch and Gary Bishop. An introduction to the kalman filter, 1995.