Clasificación de objetos similares

Trabajo Fin de Grado

Autor:
Elena Belén Bueno Benito

Tutor:
Miguel Ángel Sánchez Granero

Grado en Matemáticas

Junio, 2018
Universidad de Almería

A mis padres, a mi hermana y a toda mi familia, sólo puedo expresar mi sincero agradecimiento por apoyarme durante la etapa académica que hoy culmina.

Y a mi profesor, Miguel Ángel Sánchez Granero, siempre dispuesto a ayudarme, aconsenjarme y a enseñarme, y cómo no, a arreglarme cualquier fallo que se presentara. He aprendido mucho de él, muchas gracias.

Índice general

1 INTRODUCCIÓN **1**

2 MARCO TEÓRICO **3**
2.1. Nociones básicas **3**
2.2. Distancia de Hausdorff **3**
2.3. Procesamiento de imágenes en MATLAB **8**

3 MARCO METODOLÓGICO **11**
3.1. Requisitos **11**
3.2. Problema inicial **13**
Análisis de los resultados, 16.
3.3. Traslación **19**
Análisis de los resultados, 21.
3.4. Rotación **22**
Análisis de los resultados, 23.
3.5. Homotecia **26**
Análisis de los resultados, 27.

4 APLICACIONES **31**
4.1. Clasificación del calibre **31**
4.2. Clasificación de frutas **33**

5 CONCLUSIONES **37**

Bibliografía **39**

Abstract in English

The Hausdorff metric measures the distance between two sets. Thus, this metric can be used to determine the degree of resemblance between two objects that are superimposed on one another. In this type of systems based on the recognition of an object, Hausdorff's distance turns out to be a very suitable metric to find the most similar image, since it is very accurate and makes the percent error low. Firstly, a theoretical description of Hausdorff's distance and its use in the comparison of images is made. In a second part, efficient algorithms are provided to calculate the Hausdorff distance between two images, focusing on the case of comparing the model and the test image, and then, depending on the problem to be trated, use certain variants of the Hausdorff distance as the techniques to rigid motion (translation and rotation) or similarity. Moreover, we show how the method extends naturally to the problem of comparing real images.

Resumen en español

La distancia de Hausdorff mide la distancia entre dos conjuntos. Por lo tanto, esta distancia puede usarse para determinar el grado de semejanza entre dos objetos que se superponen uno al otro. En este tipo de sistemas basados en el reconocimiento de un objeto, la distancia de Hausdorff resulta ser una métrica muy adecuada para encontrar la imagen más similar, ya que es muy precisa y hace que el porcentaje de error sea bajo. En primer lugar, se realiza una descripción teórica de la distancia de Hausdorff y su utilización en la comparación de imágenes. En una segunda parte, se proporcionan algoritmos eficientes para calcular la distancia de Hausdorff entre dos imágenes, centrándose en el caso de comparar el modelo y la imagen de prueba, para luego, dependiendo del problema que se quiera tratar, usar ciertas variantes de la distancia de Hausdorff como el movimiento rígido o similaridad. Además, se muestra cómo los algoritmos se extienden naturalmente a los problemas de comparar imágenes reales.

1 INTRODUCCIÓN

La clasificación de objetos similares es un estudio donde el problema central se encuentra en la pregunta de en que medida un objeto difiere de otro, por ello su propósito será encontrar la técnica para la determinación de diferencias entre imágenes. A día de hoy, existen multitud de técnicas para el reconocimiento de patrones en cuyos procesos se requiere el cómputo de una medida. Algunas de las medidas más conocidas para detención de diferencias entre dos conjuntos incluye la correlación, el error cuadrático medio, la distancia euclidiana, medidas estadísticas tales como la media, la desviación estándar, la varianza, etc [3]. En este artículo, se trata de estadísticas sobre conjuntos de formas y su enfoque se basa en la distancia de Hausdorff entre las formas. Además, existen diversas transformadas como la transformada de Fourier, Coseno, Hough y otras técnicas a través de la inteligencia artificial tales como las redes neuronales artificiales o sistemas basados en lógica difusa, entre otros [9] [10]. El primer artículo, proponen una nueva medida de similitud basada en el espacio de secuencia doble y la función de módulo. Además, para manejar la incertidumbre de los datos, se utilizó el conjunto difuso intuicionista de Atanassov. El segundo documento [10] sigue un enfoque por medio de algoritmos genéticos para generar un modelo de cara desde cero y optimizarlo en base a una base de datos bastante grande de imágenes de muestra, cuyo objetivo es decidir si hay una cara en una imagen determinada y, en el caso positivo, determinar las coordenadas de la cara.

Recientemente, existen multitud de estudios basados en la determinación del grado en que dos formas difieren entre sí. La distancia de Hausdorff es una métrica frecuentemente utilizada en medidas de similitud [7] [9] y hasta la fecha ha atraído considerable atención de investigación en gráficos por computadora, geometría computacional y modelización geométrica. En el caso unidimensional para aplicaciones tales como el procesado de audio, voz o señales de sensores electrónicos y en el caso bidimensional, la detención de huellas dactilares usandose la distancia de Haudorff, la detección facial basada en una distancia de Hausdorff normalizada, iris de los ojos, palma de la mano, entre otras ([1] [6] [11]).

En este trabajo, presentamos algoritmos para la computacion eficiente de la distancia de Hausdorff sobre imágenes. En una primera parte, se tratará de contextualizar el trabajo bajo un marco teórico, incluyendo unas nociones básicas para después estudiar la distancia de Hausdorff, sus propiedades y modificaciónes. Se finaliza esta primera parte estudiando el procesamiento de imágenes en MATLAB.

En una segunda parte, basándonos en los desarrollos teóricos, implementaremos diferentes algoritmos para trabajar en la clasificación y reconocimiento de patrones entre dos imágenes para después considerar extensiones de movimientos [2]. A su vez, se analizarán los resultados con diferentes ejemplos y se estudiará su tiempo de ejecución algorítmica, aunque pueden acelerarse aún más usando hardware de gráficos de propósito especial. Finalmente, presentamos una serie de ejemplos usando imágenes reales.
Lo más importante de todo es que debe coincidir con la forma intuitiva. En otras palabras, las respuestas deben ser similares a las que un humano podría dar.

Este trabajo sirve como modo de ejemplo para la clasificación de diferentes objetos similares a partir de imágenes, ya que en el desarrollo de los algortimos se han usado imágenes que contienen números pero podría usarse con letras, polígonos o imágenes reales ([2] [8] [12]). Además, estas clasificaciones podrían usarse desde el reconocimiento óptico de caracteres OCR (del inglés Optical Character Recognition) hasta en una planta de procesamiento de verduras para proceder a su clasificación o determinar las verduras que puedan presentar algún defecto para su comercialización. En los artículos [6] y [11], muestran un sistema de detección de caras basadas en la distancia de Hausdorff (normalizada en el artículo [11]) cuyos experimentos indican una detección de caras muy precisa incluso en presencia de fondos complejos o condiciones de iluminación variables, asimismo en el artículo [1], muestra las características y los resultados de los algoritmos para localizar la zona nuclear de una huella dactilar.

Otras aplicaciones, se encuentran en la variación de la métrica, por ejemplo, el uso de la distancia de Hausdorff con traslación para el reconocimiento OCR; el uso de la distancia de Hausdorff con traslación y rotación para la clasificación del calibre de una verdura y el uso de la distancia de Hausdorff con traslación, homotecia y rotación para la clasificación entre distintas verduras.

2 MARCO TEÓRICO

2.1 Nociones básicas

Definición 2.1. *Un espacio métrico es un par $(\boldsymbol{X}, d)$ donde $\boldsymbol{X}$ es el conjunto no vacío y d es una función real definida en $\boldsymbol{X}x\boldsymbol{X}$, llamada distancia o métrica, y que satisface los siguientes axiomas:*

i) $d(x,y) \geq 0 \; \forall x, y \in \boldsymbol{X}$, *y* $d(x,y) = 0 \Leftrightarrow x = y$

ii) $d(x,y) = d(y,x) \; \forall x, y \in \boldsymbol{X}$

iii) $d(x,z) \leq d(x,y) + d(y,z) \; \forall x, y, z \in \boldsymbol{X}$

Definición 2.2. *Sea $(\boldsymbol{X},d)$ un espacio métrico. Se define $\mathcal{P}_0(\boldsymbol{X})$ como todos los subconjuntos no vacíos de $\boldsymbol{X}$.*

Definición 2.3. *Sea $(\boldsymbol{X},d)$ un espacio métrico. Se define $\mathcal{C}_0(\boldsymbol{X})$ como todos los subconjuntos cerrados y acotados no vacíos de $\boldsymbol{X}$.*

Definición 2.4. *Sea $(\boldsymbol{X},d)$ un espacio métrico. Se define $\mathcal{K}_0(\boldsymbol{X})$ como todos los subconjuntos compactos no vacíos de $\boldsymbol{X}$.*

Definición 2.5. *Sea $(\boldsymbol{X},d)$ un espacio métrico. Se define $\mathcal{F}_0(\boldsymbol{X})$ como todos los subconjuntos finitos no vacíos de $\boldsymbol{X}$.*

Nótese que $\mathcal{F}_0(\boldsymbol{X}) \subseteq \mathcal{K}_0(\boldsymbol{X}) \subseteq \mathcal{C}_0(\boldsymbol{X}) \subseteq \mathcal{P}_0(\boldsymbol{X})$

Definición 2.6. *Dado $A \in \mathcal{P}_0(\boldsymbol{X})$, la bola abierta centrada en A y radio $\varepsilon > 0$ se define como*

$$B(A,\varepsilon) = \{x \in X \, : \, d(x,A) < \varepsilon\}$$

Definición 2.7. *Dados $(x_1, y_1),(x_2, y_2) \in \mathbb{R}^2$. Se define la norma del máximo como*

$$\left\|((x_1, y_1),(x_2, y_2))\right\|_\infty = \text{máx}\{\, (|x_1 - x_2|, |y_1 - y_2|) \, : \forall (x_1, y_1),(x_2, y_2) \in \mathbb{R}^2 \,\} \tag{2.1}$$

2.2 Distancia de Hausdorff

Definición 2.8. *Dados dos conjuntos finitos $A = \{a_1, a_2, \ldots, a_p\}$ y $B = \{b_1, b_2, \ldots, b_q\}$, la distancia de Hausdorff se define como*

$$d_H = \text{máx}\{h(A,B), h(B,A)\} \tag{2.2}$$

donde

$$h(A,B) = \max_{a \in A} \min_{b \in B} \left\|a - b\right\| \tag{2.3}$$

y $\left\| \, . \, \right\|$ es alguna norma subyacente en los puntos de A y B (por ejemplo, la norma Euclidiana o la norma del máximo)

Proposición 2.1. *La función h (2.3) tiene las siguientes propiedades:*

- *Si* $A \subseteq B$ *entonces* $h(A,B) = 0$.
- $h(A,B) = h(\overline{A},\overline{B})$.
- *Si* $h(A,B) = 0$ *entonces* $A \subseteq \overline{B}$.
- $h(A,B) \leq h(A,C) + h(C,B)$.

Lema 2.1. *Sean A, B y C conjuntos de* $\mathcal{P}_0(\boldsymbol{X})$. *Si* $A \subseteq B(C,\varepsilon)$ *y* $C \subseteq B(B,\delta)$, *entonces* $A \subseteq B(B,\varepsilon+\delta)$.

Este lema permite probar la desigualdad triangular de la función d_H, teniendo en cuenta la siguiente proposición.

Proposición 2.2. *La función* d_H *(2.2) tiene las siguientes propiedades:*

- *Dado* $\varepsilon > 0$, $d_H(A,B) \leq \varepsilon$ *si, y sólo si,* $A \subseteq B(B,\varepsilon)$ *y* $B \subseteq B(A,\varepsilon)$.
- $d_H(A,B) = inf\{\varepsilon > 0 : A \subseteq B(B,\varepsilon) \text{ y } B \subseteq B(A,\varepsilon)\}$.
- *Si* $d_H(A,B) = 0$ *si y sólo si* $\overline{A} = \overline{B}$.
- $d_H(A,B) = d_H(B,A)$
- $d_H(A,B) \leq d_H(A,C) + d_H(C,B)$.

Teorema 2.1. d_H *es una métrica en* $\mathcal{F}_0(\boldsymbol{X})$

Demostración:
Veamos si la distancia de Hausdorff obedece a las propiedades métricas 2.1.

1. d_H es simétrica.
   Es claro por definición de d_H que $d_H(A,B) = d_H(B,A)$, ya que $\max\{h(A,B),h(B,A)\} = \max\{h(B,A),h(A,B)\}$

2. $d_H(A,B) = 0 \Leftrightarrow A = B$
   $\Leftarrow$) Supongamos que A = B, entonces $d_H(A,A) = h(A,A) = \max\min \|a-a\| = 0$.
   $\Rightarrow$) Supongamos que $d_H(A,B) = 0$, entonces $h(A,B) = h(B,A) = 0$.
   Definamos $h(a,B) = \min_{b\in B}\|a-b\|$, $h(b,A) = \min_{a\in A}\|b-a\|$
   Como $h(A,B) = 0$, $\max_{a\in A} h(a,B) = 0$ de donde se sigue que $h(a,B) = 0$ para todo $a \in A$, es decir, $\min_{b\in B}\|a-b\| = 0$, $\forall a \epsilon A$.
   Ahora bien, como B es finito, podemos encontrar $b \in B$ de manera que $b = a$. Así, todo elemento de A está en B, es decir, $A \subseteq B$. Análogamente se puede probar que $B \subseteq A$, de donde se sigue la igualdad entre A y B.

3. Desigualdad triangular: $d_H(A,B) \leq d_H(A,C) + d_H(C,B)$.
   Como $d_H(A,B) = \max\{h(B,A),h(A,B)\}$, vamos a probar, para tener la desigualdad deseada, que se verifican
   (1) $h(A,B) \leq d_H(A,C) + d_H(C,B)$

(2) $h(B,A) \leq d_H(A,C) + d_H(C,B)$
Probemos (1), y análogamente se podrá probar (2). A su vez, para comprobar (1) vamos a demostrar que $h(a,B) \leq d_H(a,C) + d_H(C,B)$, para todo $a \in A$. Para ello, vamos a suponer que $h(a,C) = h(a,c')$ para algún $c' \in C$ y que $h(c',B) = h(c',b')$ para algún $b' \in B$.
Entonces es
$h(a,B) \leq h(a,b') \leq h(a,c') + h(c',b') = h(a,C) + h(c',B) \leq h(A,C) + h(C,B)$
Luego,

$$\begin{aligned} h(a,B) &\leq h(A,C) + h(C,B) \\ &\leq \text{máx}\{h(A,C), h(C,A)\} + \text{máx}\{h(C,B), h(B,C)\} \\ &= d_H(A,C) + d_H(C,B) \end{aligned}$$

Tomando máximo en la desigualdad anterior:

$$h(A,B) \leq d_H(A,C) + d_H(C,B)$$

De manera similar se prueba (2) y juntando (1) y (2), tenemos

$$d_H(A,B) = \text{máx}\{h(A,B), h(B,A)\} \leq d_H(A,C) + d_H(C,B)$$

Luego, d_H es una métrica en $\mathcal{F}_0(\boldsymbol{X})$

■

La función ***h(A,B)*** se denomina ***distancia de Hausdorff dirigida de A a B*** . Identifica el punto $a \in A$ que esta más lejos de cualquier punto de B y mide la distancia desde $a \in A$ hasta su vecino más cercano de B (usando la norma dada $\|\cdot\|$), es decir, $h(A,B)$ clasifica cada punto de A en función de su distancia al punto más cercano de B y luego, toma el punto clasificado de mayor valor como la distancia. Intuitivamente, si $h(A,B) = d$, entonces cada punto de A debe estar dentro de la distancia d de algún punto de B, y también hay algún punto de A que está exactamente a la distancia d desde el punto más cercano de B.

La distancia de Hausdorff $d_H(A,B)$ es ***el máximo de h(A,B) y h(B,A)***. Por lo tanto, mide el grado de desajuste entre dos conjuntos midiendo la distancia del punto de A que está más lejos de cualquier punto de B y viceversa. Intuitivamente, si la distancia de Hausdorff es d, entonces cada punto de A debe estar dentro de una distancia d de algún punto de B y viceversa. A diferencia de la mayoría de los métodos de comparación de formas, no hay un emparejamiento explícito de puntos de A con puntos de B (por ejemplo, muchos puntos de A pueden estar cerca del mismo punto de B).

Es bien sabido que la distancia de Hausdorff $d_H(A,B)$ es una métrica sobre el conjunto de todos los conjuntos cerrados y acotados (probado en [4] y [5]). Aquí, nos restringimos a conjuntos de puntos finitos porque eso es todo lo necesario.

La distancia de Hausdorff mide la falta de coincidencia entre dos conjuntos que se encuentran en posiciones fijas entre sí. En este trabajo, nos interesa principalmente medir la discordancia entre todas las posibles posiciones relativas de dos conjuntos.

Definición 2.9. *Dado un grupo G, definimos la distancia de Hausdorff con respecto a G mínima como*

$$M_G(A,B) = min_{g_1 \in G}\, d_H(A,\, g_1 B)$$

Esta propiedad se cumple cuando G es el grupo de traslaciones, y $\|\,.\,\|$ es cualquier norma, así como cuando G es el grupo de movimientos rígidos y $\|\,.\,\|$ la norma Euclidea. En estos casos, consideramos que la distancia mínima de Hausdorff obedece a las propiedades métricas 2.1 (Es decir, la función es en todas partes positiva y tiene las propiedades de identidad, simetría y desigualdad triangular)

Definición 2.10. *El valor mínimo de la distancia de Hausdorff bajo traslaciones se define como*

$$M_T(A,B) = min_t\, d_H(A, B \oplus t) \tag{2.4}$$

donde d_H es la distancia de Hausdorff definida en (2.2), y $\oplus$ es la notación de suma estándar de Minkowski (es decir, $B \oplus t = \{b + t \,|b \in B\}$)

Definición 2.11. *El valor mínimo de la distancia de Hausdorff bajo rotaciones se define como*

$$M_R(A,B) = min_\theta\, d_H(A, (R_\theta B)) \tag{2.5}$$

donde d_H es la distancia de Hausdorff definida en (2.2), y R_θ es la matriz estándar de rotación.

Cuando se trata de la distancia de Hausdorff bajo rotaciones, en este caso, sería interesante probar: $M_R(A,B) = 0$ *si y sólo si existe una rotación* R_θ *tal que* $A = R_\theta B$
Demostración:

$$M_R(A,B) = 0 \;\Leftrightarrow\; \exists R_\theta \;\; tal\; que\; A = R_\theta B$$

$\Leftarrow$) Supongamos que si $A = R_\theta B$ para algún θ, entonces por las propiedades de d_H se tiene,

$$d_H(A,\, R_\theta B) = 0, \text{ con lo que } M_R(A,B) = min_\theta\, d_H(A,\, R_\theta B) = 0$$

Por lo tanto, $M_R(A,B) = 0$

$\Rightarrow$) Supongamos ahora que $M_R(A,B) = 0$, entonces

$\exists\theta$ tal que $d_H(A, R_\theta B) = 0$ y por las propiedades de la distancia de Hausdorff, se tiene que $A = R_\theta B$

Por lo tanto, $\exists\theta \;\; tal\; que \;\; A = R_\theta B$

■

Definición 2.12. *El valor mínimo de la distancia de Hausdorff bajo homotecias se define como*

$$M_H(A,B) = min_c\, d_H(A,\, cB) \tag{2.6}$$

donde d_H es la distancia de Hausdorff definida en (2.2), y c es el factor de homotecia.

Cuando se trata de la distancia de Hausdorff bajo homotecias, de forma similiar, será interesante probar: $M_H(A,B) = 0$ *si y sólo si existe un factor de homotecia* c *tal que* $A = cB$
Demostración:

$$M_H(A,B) = 0 \Leftrightarrow \exists c \ tal\ que\ A = cB$$

$\Leftarrow$) Supongamos que si $A = cB$ para algún c, entonces por las propiedades de d_H se tiente,

$$d_H(A, cB) = 0, \text{ con lo que } M_H(A,B) = min_c\, d_H(A,cB) = 0$$

Por lo tanto, $M_H(A,B) = 0$

$\Rightarrow$) Supongamos que $M_H(A,B) = 0$, entonces

$\exists c$ tal que $d_H(A,cB) = 0$ y por las propiedades de la distancia de Hausdorff, se tiene que $A = cB$

Por lo tanto, $\exists c \ tal\ que \ A = cB$

■

Podemos calcular una aproximación a la distancia mínima de Hausdorff bajo el movimiento rígido. La idea básica es calcular la distancia de Hausdorff entre el modelo y todas las transformaciones de la imagen que se quiere clasificar para después, encontrar la distancia mínima. El modelo se trata de una imagen fija y la imagen de prueba es la que se transforma para encontrar en que posición se asemeja más al modelo. La rotación se realizará por medio de la matriz de rotación.

Otros problemas a determinar son el mínimo de la distancia de Hausdorff compuesto por traslación y rotación, y el mínimo de la distancia de Hausdorff compuesto por traslación, rotación y homotecia.

Definición 2.13. *El valor mínimo de la distancia de Hausdorff compuesto de una rotación y traslación se define como*

$$M_{TR}(A,B) = min_{t,\theta}\, d_H(A, (R_\theta B) \oplus t) \tag{2.7}$$

donde d_H *es la distancia de Hausdorff definida en* (2.2), *y* $((R_\theta B) \oplus t) = \{R_\theta b + t \,|b \in B\}$, *y* R_θ *es la matriz estándar de rotación.*

Definición 2.14. *El valor mínimo de la distancia de Hausdorff compuesto por traslación, rotación y homotecia se define como*

$$M_{TRH}(A,B) = min_{t,\theta,c}\, d_H(A, (cR_\theta B) \oplus t) \tag{2.8}$$

donde d_H *es la distancia de Hausdorff definida en* (2.2), *y* $((cR_\theta B) \oplus t) = \{cR_\theta b + t \,|b \in B\}$, *c el factor de homotecia y* R_θ *es la matriz estándar de rotación.*

2.3 Procesamiento de imágenes en MATLAB

Como veremos en el capítulo siguiente, se trabajará con imágenes en MATLAB, por tanto hará falta tener un buen conocimiento de como el programa procesa las imágenes que se implementarán.

Para comenzar, una *imagen* es una función bidimensional $f(x,y)$, donde x e y representan las coordenadas espaciales y el valor de f en cualquier par de coordenadas (x,y) representa la intensidad de la imagen en dicho punto una imagen digital. Una *imagen digital* $f[x,y]$ esta compuesta de píxeles los cuales pueden definirse de alguna manera como pequeños puntos en la pantalla o imagen. Cada píxel es capaz de proporcionar información visual acerca de una pequeña región en particular de la imagen.

A partir de esto, *MATLAB* almacena las imágenes como vectores bidimensionales (matrices), en el que cada elemento de la matriz corresponde a un sólo pixel. En general se puede decir que una imagen de m por n esta compuesta de m píxeles en la dirección vertical y n píxeles en la dirección horizontal. El sistema de coordenadas empleado para la ubicación de cada píxel de la imagen es como se muestra la figura 2.1,

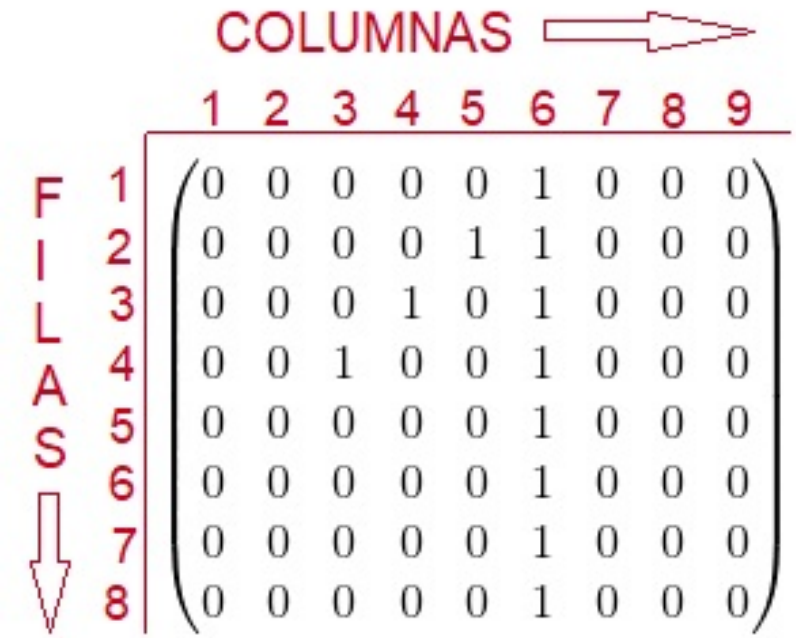

Figura 2.1: Sistema de coordenadas en MATLAB para una imagen 8x9

En segundo lugar, MATLAB soporta los siguientes formatos de imágenes:

.BMP .HDF .JPEG .PCX .TIFF .XWB

La mayoría de las imágenes disponibles hoy en día, se encuentran en el formato JPEG, el cual es el nombre para el standard de compresión de imágenes que más se utiliza. El tipo de dato que contendrá una imagen puede ser de varios tipos, según el tipo de dato de cada pixel. Pueden ser, por ejemplo, de clase double, uint8, uint16, logical, etc.

Las imágenes que se desean procesar, normalmente se encuentran en la forma de un archivo, mientras que el trabajo en MATLAB se desarrolla con una imagen vista como una matriz. La *función imread*, la cual usaremos para impletementar las imágenes, lee imágenes que se encuentren en cualquiera de los formatos de archivo de gráficos de imágenes citados anteriormente, con cualquier profundidad de bit. La mayoría de las imágenes utiliza 8 bits para guardar los valores de los píxel. Cuando estos son guardados, MATLAB los guarda como clase uint8. Para imágenes indexadas, imread siempre lee el mapa de color dentro de una matriz de clase double, aunque el mismo arreglo de la imagen sea clase uint8 o uint16. La *función imshow* la usaremos para visualizar una imagen.

Cuando los elementos de una imagen de intensidad son de *clase uint8* (enteros almacenados en 8 bits) o de *clase uint16* (enteros almacenados en 16 bits), pueden almacenar, respectivamente, $2^8 = 256$ valores en el rango $[0 : 255]$ o $2^{16} = 65536$ valores en el rango $[0 : 65535]$. Una imagen puede ser truecolor o indexada. Una imagen indexada, de altura m y anchura n, se almacena mediante dos matrices: una matriz de índices tamaño m x n, y un mapa de colores.

La *matriz de índices* es una matriz donde cada celda corresponde a un pixel de la imagen y contiene un índice a un color concreto del mapa de colores. Por ejemplo, para una imagen en color de 8 bits donde hay 256 posibles colores; cada celda de la matriz de índices tiene un valor entero entre 0 y 255 que es un índice a uno de los 256 colores descritos en el mapa de colores. Al trabajar en MATLAB se deben considerar múltiples factores como la manera en la que se carga una imagen, el formato correcto, guardar la información de diferentes maneras, como desplegar una imagen, conversiones entre diferentes formatos de imágenes, etc.

El *mapa de colores* es una matriz que tiene tres columnas, una por cada color básico (rojo, verde y azul o RGB) y una fila por cada posible color. Para el ejemplo de una imagen en color de 8 bit que tiene 256 colores, el mapa de colores sería una matriz de tamaño 256 x 3. Suponiendo que se trabaja con un formato uint8, cada fila del mapa de colores describe un color concreto y contiene tres números enteros entre 0 y 255, que dan las cantidades de rojo, verde y azul que tiene el color. Así, por ejemplo, si la fila 4 contiene los valores [15 7 0] ello indica que, en ese mapa de colores, el color número 4 no contiene azul y contiene intensidades pequeñas de rojo y verde. Muchas imágenes en blanco y negro se almacenan con un mapa de 256 colores (niveles de gris), pero en este caso, cada nivel contiene las mismas cantidades RGB, es decir, las tres celdas de cada fila contienen el mismo valor.

Una vez implementadas las imágenes en MATLAB, el programa trabajará esas imágenes en formato binario, están representadas por píxeles que constan de 1 bit cada uno, que pueden representar dos tonos (típicamente negro y blanco), utilizando los valores 0 para el negro y 1 para el blanco o viceversa. En este trabajo, se usará el valor 1 para el negro y el 0 para el blanco. El conjunto de valores que toman valor 1, será el conjunto donde se encuentra el objeto, para proceder a su clasificación.

3 MARCO METODOLÓGICO

3.1 Requisitos

Una de las condiciones más importantes para llevar a cabo esta clasificación será que las imágenes analizadas sean imágenes binarias, es decir, aquellas cuyos únicos valores posibles son 0 o 1, [13]. Para ello, se realizará esta modificación mediante el operador de Sobel.

El operador de Sobel calcula el gradiente de la intensidad de una imagen en cada punto (píxel). Así, para cada punto, este operador da la magnitud del mayor cambio posible, la dirección de éste y el sentido desde oscuro a claro. El resultado muestra cómo de abrupta o suave cambia una imagen en cada punto analizado y, en consecuencia, cuán probable es que éste represente un borde en la imagen y, también, la orientación a la que tiende ese borde. Ya que el resultado del operador Sobel es un mapeo de dos dimensiones del gradiente de cada punto, éste puede ser procesado y ser visto como una imagen, con las áreas de gradiente elevado (equivalentes a bordes) en blanco y con los demás como negro (el fondo de la imagen generada). Las siguientes imágenes 3.1 y 3.2 ilustran lo anterior, se muestra el cálculo del operador Sobel sobre una imagen. Observese las diferencias de gradiente (zonas blancas) dadas al aplicar únicamente un gradiente.

Figura 3.1: Modelo de imagen

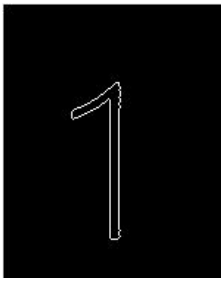

Figura 3.2: Imagen aplicando Sobel

Para la ejecución del algoritmo de Sobel, primero, se implementa las imágenes al programa y se convierten en matrices donde cada posición es un píxel de la imagen que toma un valor del 0 al 255. Un ejemplo es:

```
uno = imread('uno.jpg');
```

Se pasa la matriz a escala de grises donde los extremos son el negro y el blanco, que toman los valores 0 y 255, respectivamente. Para después, llamar a la función Sobel y transformar la imagen a 3.2. Luego, 3.2 se trata de una matriz que toma los valores 0 o 255.

Se quiere hacer el procedimiento píxel por píxel, es decir, ir cambiando el valor de los píxeles de acuerdo al valor que queramos considerar como 1 o como 0. Los valores que se quiere considerar como 1 son los que toman el valor 255 y 0 los que

toman el valor 0. Por tanto, se recorre toda la matriz obteniéndose una nueva matriz con la mismas dimensiones y cuyos objetos que se mostraban en la imagen 3.2 (zonas blancas), tomarán el valor 1. El fichero *BinarioSobel* es 3.3

```
function [X] = BinarioSobel(P)
A=rgb2gray(P); % Convierte RGB a escala de grises.

%Calculo sobel
bt = edge(A,'sobel','Sobel');

sX=size(bt);
for i=1:sX(1)
    for  j=1:sX(2)
        if(bt(i,j)==0)
            bt(i,j)=0;
        else
            bt(i,j)=1;
        end
    end
end
X=bt;
end
```

Figura 3.3: Algoritmo Sobel

Una vez obtenido la matriz en forma binaria, se sigue creando una función cuyo objetivo será la eliminación de columnas y filas donde no se encuentre ninguna coordenada del objeto en cuestión. La función es llamada *deleteZeros* y la primera parte del algoritmo es

```
function [S]= deleteZeros(P)
X=BinarioSobel(P);

newX= all(~X); % 1 si las columnas son 0 y 0 si hay algún 1
k1=sum(newX); %numero de elementos que hay igual a 2
n=zeros(1,k1);
n = uint8(n);
a=1;
for i =1:size(newX,2)
    if newX(i)==1
        n(a)=i;
        a= a+1;
    end
end
X(:,n)=[];     %se elimina las columnas
```

El comando $all(X)$ devuelve una matriz 1xn, donde n es el número de columnas que tiene la matriz X, que toman el valor 1 o 0 dependiendo de cada posición. Si en la posición i hay un 1, la columna i de la matriz X solo hay unos, por el contrario, si en la columna i de la matriz X hay algún cero, entonces la matriz resultante en la posición i tomará el valor 0. Como se quiere identificar la columnas de ceros, es por eso que se llama al comando $all(\sim X)$. La matriz resultante tomará el valor 1, si esa columna está compuesta por solo ceros y 0, si hay algún uno. Finalmente se identifica que columnas toman el valor 1 y se eliminan de la matriz inicial X. Una vez eliminadas las columnas, para eliminar las filas el proceso es equivalente, aunque se toma la traspuesta a la matriz X.

Así, el cálculo de la distancia de Hausdorff entre imágenes, se restringirá al cómputo de los objetos de la imagen modificada, es decir, la matriz donde mínimo en cada columna y fila exista una coordenada que tome valor 1, que es el objeto en cuestión.

3.2 Problema inicial

Una vez obtenidas las imágenes en forma binaria [13], haciendo uso de la definición de distancia de Hausdorff d_H (2.2) vamos a realizar un algoritmo que pueda calcular dicha distancia. Como se ha dicho en la introducción, el objetivo pricipal será crear un algoritmo cuyos resultados sean eficientes, por tanto se evaluará la sensibilidad con abundantes ejemplos.

Dadas dos imagénes cualesquiera A y B, como hemos visto en la definición, la distancia de Hausdorff será el resultado de calcular el máximo de $h(A,B)$ y $h(B,A)$ (2.3).

En primer lugar, el fichero $distHausdorff$ requiere la introducción de dos imágenes A y B. Estas no están en forma binaria ni modificadas, luego se llama a la función $deleteZeros$ para transformar cada imagen en su correspondiente imagen recortada y en forma binaria. Una vez hecho este proceso, la nuevas imágenes binarias serán X e Y, de A y B, respectivamente. El siguiente paso a seguir será calcular la distancia dirigida de X a Y y la distancia dirigida de Y a X, el proceso de ambas será equivalente. Expliquemos el primero:

Este proceso trata de recorrer la matriz X y calcular la distancia máxima de los puntos de $(x_1,x_2) \in X$ a toda la matriz Y. Este punto $(x_1,x_2) \in X$ su valor es 1, ya que como se ha dicho anteriormente, es donde estará el objeto en cuestión. Para ello, se tiene que calcular la distancia mínima que hay de un punto fijo $(x_1,x_2) \in X$ a Y que tenga como valor 1. Esto se realiza en otra función llamada $dPuntoConjunto(x_1,x_2,Y)$.

Se introduce el punto fijo de X y se calcula la distancia mínima entre el punto fijo de X y cada punto de Y. En un principio, se realizó este cálculo recorriendo todos los puntos de Y en un bucle y calculando la distancia más pequeña entre las distancias del punto fijo de X y todos los puntos que toman valor 1 de Y. Su tiempo de ejecución no era veloz, por lo que se tuvo que realizar una reforma. La función

$dPuntoConjunto(x_1, x_2, Y)$ calculará dichas distancias con la norma del máximo, definida en (2.1).

En un primer lugar, si la coordenada de (x_1, x_2) existe en Y y, además, $Y(x_1, x_2) = 1$ entonces distancia de Hausdorff es 0. Si no es así entonces pasamos al caso general. Suponemos que la distancia es un número bastante más alto que las que pueden salir, por ejemplo $d = 100000$. La idea es calcular los puntos de su entorno más próximo, luego se inicia un bucle que recorre un número suficientemente alto y se calcula los puntos de las esquinas de un cuadrado donde la coordenada (x_1, x_2) está centrada en el medio de ese cuadrado. Esto es sería:

```
sY= size(Y); % devuelve las dimensiones de Y
for i =1:max(sY)
    amin=max([1,r-i]);
    amax=min([sX(1),r+i]);
    bmin=max([1,s-i]);
    bmax=min([sX(2),s+i]);
...
end
```

Para entenderlo mejor, si tomamos $(x_1, x_2) = (2, 2)$, señalado en las imágenes como color azul, y una imagen Y de tamaño $8x10$. Las siguientes imágenes muestran las primeras iteraciones del bucle for en la imagen Y,

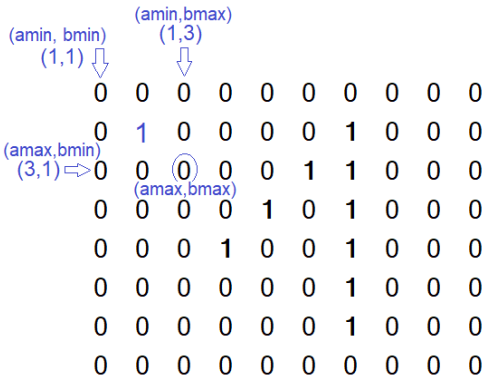

Figura 3.4: Para i = 1

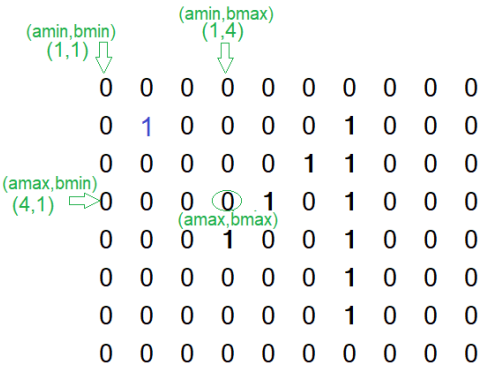

Figura 3.5: Para i = 2

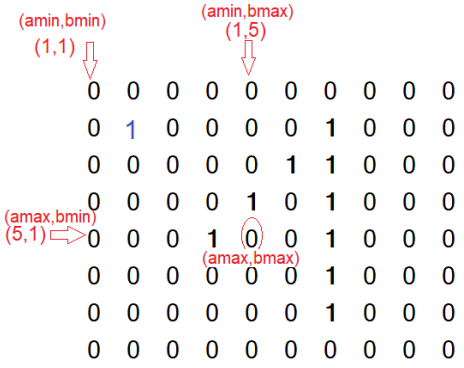

Figura 3.6: Para i = 3

Luego, habrá 4 aristas que recubran la coordenada (x_1, x_2), estas son

```
X(amin:amax,bmin) % Corresponde a la arista inferior
X(amin:amax,bmax) % Corresponde a la arista superior
X(amin,bmin:bmax) % Corresponde a la arista lateral izquierda
X(amax,bmin:bmax) % Corresponde a la arista lateral derecha
```

Si la suma de los componentes de cada arista es mayor que cero, para algún i, esto quiere decir que una o más de las coordenadas que recubre el punto (x_1, x_2) es igual a 1. Luego siguiendo la definición de la norma del máximo, $d = i$ y el progama termina, puesto que buscamos el mínimo. En el caso de que la suma sea igual a cero, entonces pasa al recubrimiento donde $i = i + 1$.

Así, el programa evalua menos puntos y su ejecución es mas eficiente y la función $dPuntoConjunto(x_1, x_2, Y)$ devolverá la distancia mínima del punto $(x_1,\ x_2) \in X$ con Y,

$$d_{(x_1,x_2)}(Y) = min\{ \left\|((x_1,\ x_2) - (y_i,\ y_j))\right\|_\infty : (y_i, y_j) \in Y\}$$

Una vez que se encuentre la distancia mínima entre el punto fijo de X con la matriz Y, la función $distHausdorff(A, B)$, en una primera parte, recorrerá la matriz de X, obteniendo todas las distancias mínimas para cada punto de X, que toman valor 1, con la matriz Y. El valor final de la distancia de Hausdorff dirigida de X a Y será:

$$h(X, Y) = max\{\ d_{(x_1,x_2)}(Y) :\ \ (x_1, x_2) \in X\ \}$$

En una segunda parte, se calculará la distancia de Hausdorff dirigida de B a A. De forma equivalente, se llega a:

$$h(Y, X) = max\{\ d_{(y_1,y_2)}(X) :\ \ (y_1, y_2) \in Y\ \}$$

Finalmente, la función $distHausdorff(A, B)$ devolverá el valor

$$d_H = max\{\ h(X, Y),\ h(Y, X)\ \}$$

Una vez explicado el cálculo de la distancia de Hausdorff, se pasa ha realizar una función que, dada una imagen y una serie de modelos, nos devuelva la imagen de los modelos con menor distancia a la imagen que queremos reconocer. De esta forma, usaremos la distancia de Hausdorff para reconocimiento o clasificación de imágenes. Llamamos a esta función $Hausdorff$.

Se trata de un algoritmo más sencillo, en el cual, se calculan todas las distancias entre la imágen de prueba P y cada una de las imágenes modelo, y la que tenga el resultado menor, será la imágen que reconozca más parecida a la imagen de prueba. Para ello, se calcula la distancia de Hausdorff de las diez imágenes,

```
d0=distHausdorff(cero,P);
d1=distHausdorff(uno,P);
d2=distHausdorff(dos,P);
d3=distHausdorff(tres,P);
d4=distHausdorff(cuatro,P);
d5=distHausdorff(cinco,P);
d6=distHausdorff(seis,P);
d7=distHausdorff(siete,P);
d8=distHausdorff(ocho,P);
d9=distHausdorff(nueve,P);
```

A continuación, se introducen en un array y se calcula el mínimo donde devuelve la distancia d menor y el índice i donde se encuentra dicha distancia. Por tanto, la imágen de los modelos que reconoce más parecida a la imagen de prueba P será la que se encuentre en la posición i, para $i = 1,\ldots,10$.

Por tanto, el número que reconoce más próximo será $A = i-1$ para todo $i = 1,\ldots,10$ y la imagen resultante será la de la posición i. Una vez explicado los algoritmos iniciales, se procede a analizar los resultados.

Análisis de los resultados

En este apartado, se evaluará el algoritmo que compara dos imágenes mediante la distancia de Hausdorff y se pondrá a prueba su fiabilidad. Para ello, se considerarán como modelos las siguientes imágenes 3.7 y se compararán con otras imágenes distintas para su análisis. Además, se estudiará la eficiencia algorítmica, ya que es de suma importancia que el algoritmo, además de ser fiable, proporcione los resultados en el menor tiempo posible.

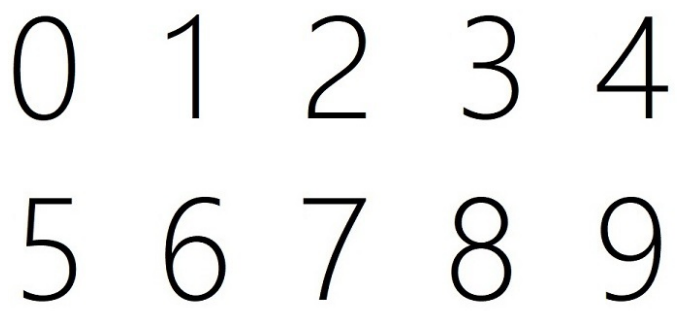

Figura 3.7: Modelos de imágenes

Para empezar, se realiza una imágen de prueba, por ejemplo la imagen 3.8

Figura 3.8: Imagen a probrar

Llamamos a la función *Hausdorff.m* introduciendo la imagen a probar *dosamano* y todas las imágenes modelo.

```
>>  Hausdorff(dosamano, cero,uno,dos,tres,cuatro,cinco,seis,siete,ocho,nueve)
numero =

     2

ans =

     11
```

Devuelve el modelo de imagen a la que más se asemeja y la distancia de Hausdorff. Se trata del modelo de imagen del número 2. Por tanto, por el momento parece que su resultado es equivalente a la intuición humana.

```
>> tic
   Hausdorff(dosamano,cero,uno,dos,tres,cuatro,cinco,seis,siete,ocho,nueve)
   toc

Elapsed time is 0.498448 seconds.
```

Se observa un buen resultado, tanto en el reconocimiento de la imagen como el tiempo de ejecución. Aunque por el momento no podemos verificar si es totalmente eficiente para todo tipo de imagen.

Veamos un segundo ensayo. La segunda imagen se trata de 3.9,

Figura 3.9:

```
>>  Hausdorff(seism, cero,uno,dos,tres,cuatro,cinco,seis,siete,ocho,nueve)
numero =

     6

ans =

     15
```

```
>>  tic
    Hausdorff(seism, cero,uno,dos,tres,cuatro,cinco,seis,siete,ocho,nueve)
    toc
Elapsed time is 0.658599 seconds.
>>
```

Devuelve el modelo del número 6, por lo que en un principio podemos suponer que el algortimo realiza bien el reconocimiento de la imagen incluso con colores más oscuros y con una escritura peor hecha.

La siguiente imagen 3.10 se trata de una imagen trasladada y realizada a mano, además con un fondo de color más oscuro,

Figura 3.10: Tercera imagen a probrar

```
>>  Hausdorff(unom, cero,uno,dos,tres,cuatro,cinco,seis,siete,ocho,nueve)
numero =

      1
ans =

     25

>>  tic
    Hausdorff(seism, cero,uno,dos,tres,cuatro,cinco,seis,siete,ocho,nueve)
    toc
Elapsed time is 0.531287 seconds.
>>
```

Nuestro objetivo es evaluar el algoritmo y buscar donde da fallos, es por eso que las imágenes realizadas se tienen con una caligrafía en peor calidad.
Finalmente, dos ejemplos más

Figura 3.11:

Figura 3.12:

Se obtienen como resultado,

```
numero =

     4
ans =

    15

Elapsed time is 0.643801 seconds.
```

```
numero =

     9
ans =

    13

Elapsed time is 0.456027 seconds.
```

En las siguientes secciones se mejorará el programa mediante técnicas de *Rotación*, *Traslación* y *Homotecia*, además, de la composición de dichas funciones, para obtener resultados con mejor precisión para cualquier tipo de posición en la que se encuentre el objeto.

El proceso de eliminación de filas y columnas en blanco se puede ver como una *Traslación*, con lo que usarla para la mejora no deberá ser necesaria para el cálculo de la distancia. En este trabajo, se realizará dicha función para evitar los casos particulares en los que si sea necesaria usar la *Traslación*.

3.3 Traslación

En esta sección, primero construiremos la función de traslación 2.4 para después encontrar para que valores del vector de traslación (a, b) la distancia de Hausdorff es la más pequeña.

En la primera parte, crear la función de traslación no fue necesaria, puesto que existe una en MATLAB que introduciendole el vector de traslación (a, b) y las opciones que se elige, la traslación era correcta. La función que se selecciona será

```
imtranslate(Q,[round(x(1)),round(x(2))],'FillValues',255,
                                                'OutputView','full');
```

'FillValues' rellena los valores para el conjunto de datos generado al trasladar la matriz, en nuetro caso queremos que tome el 255, que corresponde al color blanco.
'OutputView' corrige la imagen y junto al parámetro establecido en *'Full'* para evitar recortar la imagen trasladada.
'round(x)' trata de coger solo los enteros del valor x, puesto que la función de traslación no hace lo esperado cuando no se trata de enteros.

A continuación, usando *TOOLBOX*, buscamos una función que minimice nuestra función de traslación, la cual, se trata de una función no lineal de dos variables. Así, usamos la función *fminsearch* de *MATLAB*. Encuentra un mínimo de función multivariable no restringida utilizando un método libre de derivados.

Definición 3.1. *La función fminsearch busca el mínimo de un problema especificado por*

$$min_x\, f(x) \tag{3.1}$$

donde $f(x)$ es una función que devuelve un escalar, y x un vector o una matriz.

La sintasis puede ser de diferentes formas. Solo nos interesa saber el valor minimo de la función luego se elige

```
[~, fval] = fminsearch(f, x0, options)
```

El símbolo ~ en *MATLAB* se usa para obviar la salida de dicho valor al ejecutarse. Dado que solo interesa el valor minimal y no donde lo obtiene, usamos dicho símbolo. x_0 corresponde al punto de partida que se quiera establecer. En *Options* existe diferentes optiones para usar, se usa

```
options=optimset('TolFun',1e-10,'MaxIter',20,'Display','iter',
                                        'Algorithm','interior-point-legacy');
```

Se prueba que algortimo es más eficiente para el problema no lineal es *'interior – point–legacy'*. Las optiones restantes, *'Display','iter'* son la realización de varias iteraciones; *'MaxIter'*, 20 es el máximo número de iteraciones que se necesita; *'TolFun'*, $1e-10$ es un límite inferior en el cambio en el valor de la función objetivo durante un paso.

Finalmente, una vez estudiadas las opciones de la función *fminsearch*, se crea el fichero *fminRotacion* que es una función donde se introducen las dos imágenes, la modelo y la imagen a clasificar, y se obtiene como resultado el valor mínimo. Esto es,

```
1  function [dHT] = fminTraslacion(P,Q)
2  options = optimset('TolFun',1e-10,'MaxIter',20,'Display','iter','Algorithm','interior-point-legacy');
3  fun1 = @(x)distHausdorff(P,imtranslate(Q,[round(x(1)),round(x(2))],'FillValues',255,'OutputView','full'));
4  [~,d1]=fminsearch(fun1,[round(size(Q,1)/2),round(size(Q,2)/2)],options);
5  fun2 = @(x)distHausdorff(Q,imtranslate(P,[round(x(1)),round(x(2))],'FillValues',255,'OutputView','full'));
6  [~,d2]=fminsearch(fun2,[round(size(P,1)/2),round(size(P,2)/2)],options);
7
8  dHT = min(d1,d2);
9  end
```

Figura 3.13: Algoritmo que minimiza la composición de la distancia de Hausdorff con la función de traslación

Donde escogemos como punto de partida, el punto medio del cuadrante de la derecha superior y, además, dado que la función de traslación para los números negativos no se traslada correctamente, evaluamos los dos posibles resultados. El primero, minimizar la $distanciadeHausdorff(P,Q)$ con la función de traslación para la imagen Q y después, para la imagen P. En este caso, la imagen modelo si será modificada puesto que la función *imtraslate* solo toma vectores positivos.

Pasamos al análisis de los resultados.

Análisis de los resultados

En primer lugar, se observa como se ejecuta el programa $fminTraslacion(P,Q)$ para el ejemplo del modelo 1 con el tercer ejemplo 3.10,

```
>> fminTraslacion(uno,unom)
 Iteration   Func-count     min f(x)         Procedure
 0            1               88
 1            3               88         initial simplex
 2            5               81         expand
 3            6               81         reflect
 4            8               73         reflect
...         ...  ...  ...
20           58              25         shrink
ans =

     25

Elapsed time is 0.413479  seconds.
```

Se trata de una pequeña muestra de lo que el programa nos devuelve, ya que se trata de 20 iteraciones. En los siguientes problemas, solo se mostrará el resultado final.

La función $HausdorffT$ reconocerá la imagen de prueba con su correspondiente modelo, luego modificamos la función $Hausdorff$ y en vez de calcular directamente todas las distancias de Hausdorff, se evaluará cada una de ellas junto a la traslación. Probemos con un ejemplo, se trata de la imagen de prueba correspondiente al número 5 rotado 3.14,

Figura 3.14:

```
>> HausdorffT(cincom, cero,uno,dos,tres,cuatro,cinco,seis,siete,ocho,nueve)
numero =

    0
ans =

    48
Elapsed time is 8.742163 seconds.
```

La función de translación no es suficiente a la hora de clasificar imágenes, se observa como con una imagen rotada no realiza un buen reconocimiento del objeto. Debido a esto, se pasará a implementar la *Rotación*.

3.4 Rotación

En esta sección, se contruirá la función de rotación definida en 2.5 y seguidamente, se minimizará para que ángulo θ la distancia de Hausdorff es menor.

Esta vez no se utilizará la funcion de Rotación que existe en *MATLAB*. Se creará una función en el fichero *Rotacion* donde necesite implementar la imagen y el ángulo θ para efectuar la rotación. Esta función será calculada con la matriz de rotación,

$$\begin{pmatrix} cos(\theta) & sen(\theta) \\ -sen(\theta) & cos(\theta) \end{pmatrix}$$

Luego, se recorre en dos bucles las dimensiones de la matriz que se incorpora, donde cada coordenada será,

```
coords=zeros(size(I,1),size(I,2),2);
coseno=cosd(theta);
seno = sind(theta);
for i=1:size(I,1)
    for j=1:size(I,2)
        nx=(j*coseno)-(i*seno);
        ny=(j*seno)+(i*coseno);
        coords(i,j,1)=nx;
        coords(i,j,2)=ny;
    end
end

 coords=round(coords);
 minx=min(min(coords(:,:,1)));
 miny=min(min(coords(:,:,2)));
 if(minx <= 0)
     coords(:,:,1)=coords(:,:,1)+abs(minx)+1;
 end
 if (miny <= 0)
     coords(:,:,2)=coords(:,:,2)+abs(miny)+1;
 end
```

Finalmente, se recorre la matriz que se quiere rotar, I, y se implementan los cálculos para cada coordenada $(i,j)\epsilon I$ en una nueva matriz G, cuya dimensión es la coordenada máxima en el eje x e y y donde en un momento inicial toma el valor 255 en toda la matriz, que corresponde al color blanco. Muchas coordenadas serán modificadas por el correspondiente cómputo y los restantes tomarán el valor 255.

Una vez explicado el algoritmo que rota la imagen I con un ángulo θ, se procede a crear una función que minimice el valor de la distancia de Hausdorff con la función de rotación. Esta vez se tiene una función no lineal con una variable y sin restricciones. Para ello, se investiga las funciones que existen en $MATLAB$ para este caso. Se encuentra la función $fminbnd$. Se trata de un método que encuentra un mínimo de función de variable única en un intervalo fijo.

Definición 3.2. *La función fminbnd es un minimizador unidimensional que encuentra un mínimo para un problema especificado por*

$$min_x f(x) \quad x_1 < x < x_2$$

donde x_1, x, x_2 son escalares finitos y $f(x)$ una función que devuelve un escalar.

Existen varias sintasis de esta función, en este caso escogeremos la misma forma que en la función usada en el apartado de traslación, ya que lo que nos interesa es el valor minimal y no el punto en donde se encuentra. Luego, se crea un archivo llamado $fminRotacion$ donde nuestras opciones serán las mismas a la anterior. El algoritmo será,

```
1   function [d] = fminRotacion(P,Q)
2 -   options = optimset('TolFun',1e-14,'MaxIter',20,'Display','iter','Algorithm','interior-point-legacy');
3
4 -   fun1 = @(x) distHausdorff(P,Rotacion(Q,x(1)));
5 -   [~,d]=fminbnd(fun1,0,360,options);
6 -   end
```

Figura 3.15: Algoritmo que minimiza la composición de la distancia de Hausdorff con la función de rotación

Se procede al análisis de los resultados.

Análisis de los resultados

Recordemos que la imagen de prueba precedente 3.14 se trataba de una imagen rotada que tenía como objeto el número 5, pero el programa lo reconocía como el número 0. Si comparamos la imagen modelo 5, luego aplicamos la función $fminRotacion(cinco, cincom)$ obteniendose,

```
>> fminRotacion(cinco, cincom)
Func-count      x          f(x)       Procedure
 1          137.508          60         initial
 2          222.492          53         golden
 3          275.016          16         golden

...  ...    ...    ...
20           273.349          10         golden
```

```
ans =

    10

Elapsed time is 1.470605 seconds.

>> >> HausdorffR(cincom, cero,uno,dos,tres,cuatro,cinco,seis,siete,ocho,nueve)
numero =

    5

ans =

    10

Elapsed time is 17.693736 seconds.
```

Para este ejemplo el algoritmo funciona correctamente, aunque la función de traslación se ejecuta con mayor rápidez, su resultado no es eficiente, que como vemos la función de rotacion si nos proporciona un resultado correcto. Sin embargo, no siempre la función de rotación proporcionará el resultado más óptimo, al igual que pasa con la traslación, es por eso que se realizará la composición de las dos. Esta vez, nos encontraremos con la minimización de tres variables, el vector de traslación (a, b) y el ángulo de rotación θ, cuya definición se encuentra en (2.7). Llamamos a dicha función $fminRT(P, Q)$ y realicemos un análisis de la siguiente imagen de prueba,

Figura 3.16:

Para la distancia de Hausdorff sin traslación ni traslación, la imagen 3.16 da como clasificado:

```
>> Hausdorff(tresm, cero,uno,dos,tres,cuatro,cinco,seis,siete,ocho,nueve)
numero =

    8

ans =

    19

Elapsed time is 0.433356 seconds.
```

Para el caso de *Traslación*,

```
>> HausdorffT(tresm, cero,uno,dos,tres,cuatro,cinco,seis,siete,ocho,nueve)
numero =

    8

ans =

    19

Elapsed time is 4.339238 seconds.
```

Para el caso de *Rotación*,

```
>> HausdorffR(tresm, cero,uno,dos,tres,cuatro,cinco,seis,siete,ocho,nueve)
numero =

    3

ans =

    8

Elapsed time is 12.117258 seconds.
```

Para el caso de la composición de *Traslación* y *Rotación*,

```
>> HausdorffRT(tresm, cero,uno,dos,tres,cuatro,cinco,seis,siete,ocho,nueve)
numero =

    3


ans =

    10

Elapsed time is 38.660965 seconds.
```

Como vemos en este problema, la composición de ambas proporciona una distancia mayor, aunque obtiene un reconocimiento correcto. Se puede observar que el tiempo de ejecución es eficiente. Aunque se piense que dicha función sea la más óptima para todo tipo de problemas, la respuesta es incorrecta.

A continuación, se planteará otro problema y veremos su progreso obtenido. Se trata de la alteración de las dimensiones de un objeto o una imagen de prueba, ya que la imagen podría reconocerse con una que no le corresponde si usamos solamente la composición de traslación y rotación. Es por eso que se va estudiar la *Homotecia*.

3.5 Homotecia

Una vez probado que la composición de la traslación y la rotación es la más óptima por el momento, realizaremos la composición pero esta vez incluyendo la Homotecia definida en (2.8). Esta vez, no hará falta crear una función donde solo se realice la homotecia, sino se transformará la función de *Rotación* explicada en la sección precedente, y se renombrará como *Rot_Tras_Hom*. El cambio se producirá en la introducción del vector de traslación y del factor de homotecia en la matriz de rotación. Dada la imagen I, el ángulo de rotación, θ, el vector de traslación, (a, b), y el factor de la homotecia, c, el cambio que ha de producirse será el siguiente:

```
coords=zeros(size(I,1),size(I,2),2);
coseno=cosd(theta);
seno = sind(theta);
for i=1:size(I,1)
    for j=1:size(I,2)
        nx=a+c*((j*coseno)-(i*seno));
        ny=b+c*((j*seno)+(i*coseno));
        coords(i,j,1)=nx;
        coords(i,j,2)=ny;
    end
end
```

Luego, se trata de una función no lineal donde se deberá de minimizar la distancia de Hausdorff con 4 variables. El algoritmo que minimiza dicha función se asemeja a las anteriores, esta es 3.17

```
1   function dHT = fminRTHomotecia(P,Q)
2 -   options = optimset('TolFun',1e-10,'Display','iter','Algorithm','interior-point-legacy');
3     % x(1) el ángulo de rotación
4     % x(2) y x(3) el vector de traslación
5     % x(4) el factor de homotecia
6 -   fun1 = @(x)distHausdorff(P,Rot_Tras_Hom(Q,x(1),round(x(2)),round(x(3)),x(4)));
7 -   [~,d1]=fminsearch(fun1,[180,0,0,1],options);
8
9 -   dHT = d1;
10 -  end
```

Figura 3.17: Algoritmo que minimiza la distancia de Hausdorff con la composición de homotecia, rotación y traslación

Se prodece a la evaluación de la composición de homotecia, rotación y traslación.

Análisis de los resultados

Supongamos que se quiere realizar el reconocimiento de una imagen cuyo objeto que esta contenido se encuentra con otra dimensión, rotada y trasladada. A continuación, se hará un análisis de las soluciones que nos aportarían las funciones estudiadas anteriormente (*disHausdorff*, *fminTraslación*, *fminRotación* y *fminRT*) junto con la función actual (*fminRTH*).

Figura 3.18:

Para la distancia de Hausdorff inicial, la imagen 3.18 realiza el reconocimiento siguiente

```
>> Hausdorff(sietem, cero,uno,dos,tres,cuatro,cinco,seis,siete,ocho,nueve)
numero =

     4


ans =

     67

Elapsed time is 0.758546 seconds.
```

Para el caso de *Traslación*,

```
>> HausdorffT(sietem, cero,uno,dos,tres,cuatro,cinco,seis,siete,ocho,nueve)
numero =

     4


ans =

     67

Elapsed time is 10.025597 seconds.
```

Para el caso de *Rotación*,

```
>> HausdorffR(sietem, cero,uno,dos,tres,cuatro,cinco,seis,siete,ocho,nueve)
numero =

     0


ans =

     47

Elapsed time is 25.546080 seconds.
```

Para el caso de la composición de *Traslación* y *Rotación*,

```
>> HausdorffRT(sietem, cero,uno,dos,tres,cuatro,cinco,seis,siete,ocho,nueve)
numero =

     4


ans =

     25



Elapsed time is 64.063563 seconds.
```

Finalmente, para la composición de *Homotecia*, *Traslación* y *Rotación* se obtiene,

```
>> HausdorffRTH(sietem, cero,uno,dos,tres,cuatro,cinco,seis,siete,ocho,nueve)
numero =

     7


ans =

     25


Elapsed time is 70.717363 seconds.
```

Observamos que cuando se trata de imagenes con diferentes tamaños las funciones estudiadas anteriormente podrían dar fallos en su reconocimiento. Esta última función ha obtenido el resultado deseado y, además, su tiempo de ejecución, es semejante a la composición de *Rotación* y *Traslación*.

Veamos otros dos ejemplos, cuyas imágenes son las siguientes

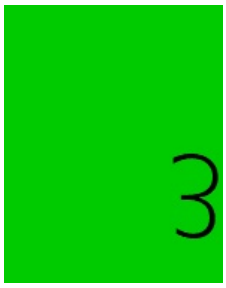

Figura 3.19:

```
numero =

     3
ans =

     5

Elapsed time is 28.479822 seconds.
```

Figura 3.20:

```
numero =

     0
ans =

    14

Elapsed time is 104.816531 seconds.
```

Figura 3.21:

```
numero =

     2
ans =

    10

Elapsed time is 53.664845 seconds.
```

Figura 3.22:

```
numero =

     4
ans =

    14

Elapsed time is 106.682170 seconds.
```

Los tiempos de ejecución se ven incrementados si la foto de prueba tiene dimensiones mayores. Las pruebas estudiadas para la composición de las tres funciones parece abarcar cualquier tipo de problema que se puede encontrar al reconocimiento de una imagen. Aunque el tiempo de ejecución no llegue a 2 minutos, en aplicaciones para tiempo real sería un inconveniente. El reconocimiento puede ser mucho más aplio que otras técnicas, pero a cambio de un tiempo de ejecuión mayor.

En el capítulo siguiente, se verá algún ejemplo de aplicación, mencionados en la introducción, como por ejemplo, clasificar un verdura por su calibre o el reconocimiento de frutas o verduras.

4 APLICACIONES

En esta sección, se profundiza los ejemplos estudiados en la sección precedente 3. Se ilustrará algunos de los aspectos cualitativos de la función de distancia de Hausdorff comparando algunas imágenes reales simples utilizando los algoritmos descritos en la sección anterior 3. Además de proporcionar una distancia d_H, entre dos imágenes, los algoritmos proporcionarán la imagen modelo a la que menos difiere de la imagen de prueba. En una primera parte, se expondrá diferentes tipos de tomates como modelo, para luego, clasificarlo mediante el calibre. En segundo lugar, se presentarán diferentes tipos de verduras o frutas para proceder a la clasificación.

4.1 Clasificación del calibre

En primer lugar, la imagen de entrada debe ser transformada en una imagen binaria. Esta imagen binaria debería contener suficiente información para obtener un correcto reconocimiento. En la Figura 4.1 se muestran los diferentes tipos de tomate que se van a considerar. Para este caso, se usará solamente la rotación y la traslación, ya que queremos medir el calibre de un tomate, luego cada foto modelo debe estar en su proporción. Dada la Figura 4.1 se clasificará en tres tipos según su calibre, los pequeños, medianos y grandes.

En el artículo [11], explican la importancia de un correcto umbral para la detección facial. En su caso, si aplica un umbral bajo obtienen como resultado una imagen conteniendo demasiadas aristas, incluyendo una gran cantidad de puntos correspondientes al fondo que afectarían en gran medida al paso de la localización facial y si se aplica un umbral mayor, el número detectado de aristas disminuiría, pero también desaparecerían puntos muy importantes, como los relacionados con elementos faciales. En las Figura 4.3, 4.2 y 4.4 se muestran los modelos de tomates en forma binaria, con su umbral más bajo, más alto y normal.

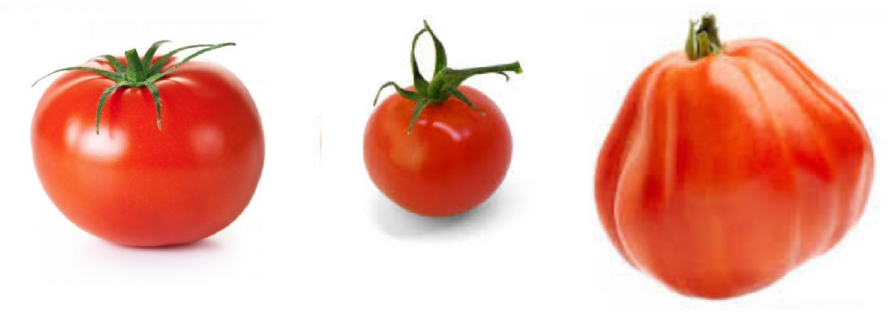

Figura 4.1: Diferentes tamaños de tomates

Figura 4.2: Umbral alto

Figura 4.3: Umbral bajo

Figura 4.4: Umbral medio

Para cada imagen real habría que encontrar el mejor umbral, para evitar datos irrelevantes al objeto en cuestión. Procedemos a realizar la clasificación de los objetos según el calibre. La Figura 4.5 se trata de un pequeño tomate cuyo programa deberá de identificarlo como un tomate de tamaño pequeño y la Figura 4.6 se trata de un tomate raf de tamaño mediano. Se ejecuta el programa *HausdorffRT* obteniendose los resultados siguientes,

Figura 4.5: Tomate cherry

Figura 4.6: Tomate raf

```
>> HausdorffRT(P,peq,med,grand)

numero =

     1
ans =

     7

Elapsed time is 14.177385 seconds.
```

```
>> HausdorffRT(P,peq,med,grand)

numero =

     2
ans =

    22

Elapsed time is 30.57652 seconds.
```

Donde el valor uno equivale al tamaño pequeño, el valor 2 al medio y 3 al grande. Se observa unos resultados eficientes.

Se finaliza esta sección mostrando dos últimos ejemplos de figuras de imágenes reales. La figura 4.7 corresponde a un tomate Corazón de Buey, donde suelen ser grandes y la figura 4.8 corresponde a un tomame pera, donde suelen ser pequeños

Figura 4.7: Tomate Corazón de Buey

Figura 4.8: Tomate pera

```
>> HausdorffRT(P,peq,med,grand)

numero =

     3

ans =

     39

Elapsed time is 42.776074  seconds.
```

```
>> HausdorffRT(P,peq,med,grand)

numero =

     1

ans =

     23

Elapsed time is 15.731949 seconds.
```

4.2 *Clasificación de frutas*

A continuación, mediante el uso de la distancia de Hausdorff con traslación, rotación y homotecia se clasificarán diferentes tipos de frutas. La figura 4.9 representa todas las frutas que se usan como modelos.

Figura 4.9: Frutas a clasificar

Donde los modelos de imágenes, incorporados al programa, se encuentran en el orden de la imagen precedente 4.9.

Ejecutando el programa *HausdorffRTH* con la Figura 4.10 nos devuelve

Figura 4.10: Prueba 1

```
>> HausdorffRTH(prueba1,fresa,pera,naranja,platano,pina)
imagen =

     5


ans =

     9

Elapsed time is 55.548017 seconds.
```

Esta imagen cuyas dimensiones son mayores, obtiene un resultado correcto. El siguiente ejemplo se trata de la Figura 4.11, cuyas dimensiones son mayores a la del modelo además de estar rotada y trasladada.

Figura 4.11: Prueba 2

```
>> HausdorffRTH(prueba2,fresa,pera,naranja,platano,pina)
imagen =

     1


ans =

     5

Elapsed time is 28.984597 seconds.
```

Figura 4.12: Prueba 4

Figura 4.13: Prueba 5

```
>> HausdorffRTH(prueba,fresa,pera,naranja,platano,pina)
```

```
Prueba 4

numero =

     4
ans =

     24

Elapsed time is 51.433765  seconds.
```

```
Prueba 5

numero =

     2
ans =

     14

Elapsed time is 32.815524 seconds.
```

Observamos como el programa nos da resultados óptimos y eficientes. El tiempo y los resultados cumplen con la calidad propuesta y constituyen elementos inherentes a la eficiencia.

5 CONCLUSIONES

En este trabajo, se ha puesto en práctica conocimientos de asignaturas cursadas a lo largo de esta carrera. Primordialmente, los conocimientos de los distintos lenguajes de programación vistos durante la carrera han sido esenciales, en especial, la asigantura de optimización me ha brindado la oportunidad de poner en práctica conocimientos de Toolbox en *MATLAB* para la minimización de funciones. No obstante, también han sido fundamentales los conocimientos de topología y geometría, para aplicarlos en la construcción del marco teórico y para el cálculo de la distancia de Hausdorff.

El marco teórico ha representado una parte importante para el cálculo de la distancia de Hausdorff. Gracias a la base teórica, se han podido realizar los diferentes algoritmos anteriores. Para ello, se ha debido incorporar desde las nociones más básicas hasta el estudio de las propiedades métricas para la distancia de Hausdorff bajo rotaciones y homotecias.

La realización de las comprobaciones de los algoritmos con diferentes objetos nos ha permitido elaborar nuevos algoritmos donde mejoraba la eficiencia y el cálculo de la distancia de Hausdorff. Esto ha contribuido a alcanzar uno de nuestros objetivos más importantes, realizar una clasificación de objetos. La primera clasificación trabajada ha sido con objetos simples, números, en diferentes posiciones y dimensiones para la obtención de un buen resultado, para luego aplicarlos a objetos reales y de gran relevancia en la industria, como puede ser la clasificación del calibre de un tomate o la clasificación de frutas.

Aquí proporcionamos algoritmos que son altamente eficientes. Estos métodos operan en forma binaria, haciéndolos especialmente adecuados para el procesamiento de imágenes y aplicaciones de visión artificial, entre otros. Las tres ventajas clave del enfoque son la insensibilidad relativa a pequeñas perturbaciones de la imagen, simplicidad y velocidad de cálculo, y reconocimiento natural de una imagen de dimensión mayor en comparación con otra. Esto conlleva a una felixibilidad mayor que a los distintos métodos mencionados en la introducción, además, la distancia Hausdorff ha mostrado sus capacidades de discriminación en diversas aplicaciones de reconocimiento de patrones, como la comparación de imágenes faciales o de huellas dactilares. Se puede deducir fácilmente que una de las deficiencias de la distancia de Hausdorff es que presenta sensibilidad en el cálculo cuando las imágenes presentan ruido en el fondo, ya que puede variar el valor tomando datos irrelevantes al objeto. Este inconveniente de la distancia Hausdorff origina que si se desean comparar de forma exacta dos imágenes cuya imágen de prueba contiene un fondo animado, se deberá analizar las imágenes por porciones más pequeñas, como lo efectuan en diversos estudios [1], [6], [8], [11].

Este algoritmo ha ido cambiando de forma hasta el final. En una primera parte, se realizó el cálculo de la distancia de Hausdorff con la norma Euclidea y sin realizar las funciones de minimización de la imagen, pero a medida que se realizaba el análisis de los resultados, éstos presentaban numerosos problemas, como la velocidad de ejecución. Por ello, se ha realizado diferentes modificaciones hasta llegar a la presentada en este trabajo.

Por las razones dichas en la introducción, este proceso ha sido efocado en el desarrollo de algorimos para el reconocimiento de patrones, una materia que está a la vanguardia y que tiene gran relevancia por el avance tecnológico que está sucediendo a gran velocidad en el mundo. Como consecuencia se están produciendo multitud de investigaciones con diferentes técnicas matemáticas y con diversas aplicaciones.

A lo largo de este proyecto, nos hemos encontrado aspectos en los que se podría seguir investigando tanto en la parte teórica como en la práctica, por ejemplo, la elaboración de un cálculo más rápido y exacto de la distancia de Hausdorff y sus modificaciones. Además, una elaboración de ese ámbito podría generar menos limitaciones al relacionar objetos reales.

Para finalizar mi trabajo, se citarán las palabras de dos grandes matemáticos

En matemáticas uno no entiende las cosas, se acostumbra a ellas.
John Von Neumann

El razonamiento matemático puede considerarse más bien esquemáticamente como el ejercicio de una combinación de dos instalaciones, que podemos llamar la intuición y el ingenio.
Alan Turing

Bibliografía

[1] Henry ARGUELLO, *Comparación de huellas dactilares usando la distancia Hausdorff*, Sistemas de Cibernetica e Informática, Vol 5, No 2, 2008 (1690-8627).

[2] E. Arkin, L. P. Chew, D. P. Huttenlocher, K. Kedem, and J. S. B. Mitchell, *An efficiently computable metric for comparing polygonal shapes*, IEEE Trans. Patt. Anal. Machine Intell., Vol. 13, No. 3 (1991), 209-216.

[3] G. Charpiat, O. Faugeras, R. Keriven, P. Maurel, *Distance based shape statistics*, IEEE Trans. Patt. Anal. Machine Intell., 2006.

[4] R. Engelking, *General topology, Heldermann*, Verlag Berlin, 1989.

[5] Franco Barragán Mendoza, *Breve introducción a la Métrica de Hausdorff*, Universidad Tecnológica de la Mixteca, 2014.

[6] Y. Gao, *Efficiently comparing face images using a modified Hausdorff distance*, IEEE 2003.

[7] D. P. Huttenlocher, K. Kedem, J. M. Kleinberg, *On dynamic Voronoi diagrams and the minimum Hausdorff distance for point sets under Euclidean motion in the plane*, Proc. 8th Annu. Symp. Comput. Geometry, pp. 110-119, Jun. 1992.

[8] D.P. Huttenlocher, G.A. Klanderman, W.J. Rucklidge, *Comparing images with Hausdorff distance*, IEEE, Septembre 1993.

[9] C.M. Hwang, M.-S. Yang, W.-L. Hung, M.-G. Lee, *A similarity measure of intuitionistic fuzzy sets based on the Sugeno integral with its application to pattern recognition*, Inf. Sci., Vol. 189, pp. 93-109, Apr. 2012.

[10] K.J. Kirchberg, O. Jesorsky, and R.W. Frischholz, *Genetic model optimization for Hausdorff distance-based face localization*, In Proc. International ECCV 2002 Workshop on Biometric Authentication, Springer, Lecture Notes in Computer Science, Copenhagen, Denmark, June 2002.

[11] Pablo Suau, Francisco Pujol, Ramón Rizo, Mar Pujol, *Detección facial basada en una distancia de Hausdorff normalizada*, Departamento de Ciencia de la Computación e inteligencia artificial, 2005.

[12] Godfried T. Toussaint, *Computing visibility properties of polygons*, Vol 7 (1988), 103-122 1988.

[13] Y. Zhang, Q. Xiao, *An optimized approach for fingerprint binarization*, IEEE 2006.

Otros Recursos

[14] Funciones para el procesamiento de imágenes: https://es.mathworks.com/help/images/functionlist.html. Última visita: 01/07/2018

[15] Referencia para la función fminsearch en MATLAB: http://www.ieap.uni-kiel.de/lehre/vorlesungen/matlab/fit/fminsearch.pdf

[16] Para la resolución de problemas no lineales: https://es.mathworks.com/help/optim/nonlinear-programming.html. Última visita: 30/06/2018

[17] Para la detención de bordes en una imagen: https://es.mathworks.com/help/images/ref/edge.html. Última visita: 30/06/2018

[18] Referencia para el estudio de las imágenes: https://es.mathworks.com/help/matlab/ref/colorspec.html. Última visita: 01/07/2018

[19] Referencia para aplicaciones donde el rendimiento sea crítico y se quiera vectorizar los programas: https://mat.caminos.upm.es/wiki/Prog31:_Vectores,_bucles_y_vectorizaci%C3%B3n. Última visita: 01/07/2018

[20] Referencia para la vectorización: https://es.mathworks.com/help/matlab/matlab_prog/vectorization.html. Última visita: 01/07/2018