Capítulo 3

Clustering

Este capítulo presenta la manera de agrupar los elementos de un plano que son similares en cuanto a las dos características principales que nos interesan en una textura estructurada, que son, la forma y el emplazamiento. Inicialmente realizamos una catalogación general de los diversos algoritmos de agrupación o clustering, para centrarnos en el problema de agrupación de formas similares con vecindades similares. Este problema lo resolvemos mediante un clustering jerárquico. El cálculo del representante de un cluster se realiza mediante el cálculo de la forma y la vecindad medias, utilizando el algoritmo del cálculo de la cadena media explicado en el capítulo 2.

3.1 Presentación del problema y definiciones

Hemos visto en el capítulo 1 que una de las fases de nuestro trabajo consiste en segmentar, dentro de un plano de arquitectura, las texturas estructuradas que aparezcan en él. En ese mismo capítulo se explica la representación de este plano por medio de un grafo de adyacencia de regiones que representa los polígonos que aparecen en él y sus relaciones de vecindad. Además, una textura estructurada viene definida, en este dominio, por una o varias clases de polígonos emplazados siguiendo unas reglas de emplazamiento. Como consecuencia de todo ello, segmentar zonas con una textura estructurada, supone buscar grupos de polígonos dentro del grafo que representa el plano, similares en cuanto a su forma y regla de emplazamiento. Es decir, necesitamos hacer un *clustering* de polígonos utilizando sus características de forma y regla de emplazamiento.

El *clustering* o agrupación es un campo dentro del aprendizaje no supervisado. Supone agrupar un conjunto de objetos que sean similares, de manera que dos objetos dentro de un mismo grupo sean más parecidos entre sí que con otro elemento que pertenezca a otro grupo. Se debe distinguir entre clustering, o clasificación no supervisada, y análisis discriminante, o clasificación supervisada. En el primero se tiene un conjunto de datos que queremos agrupar, según sus características, en grupos homogéneos. Mientras que el segundo consiste en tener un conjunto de patrones,

etiquetados *a priori* para indicar a qué clase pertenecen, que sirven para aprender la descripción de las clases y posteriormente con esa descripción clasificar nuevos patrones desconocidos.

Una aplicación del clustering es la segmentación de imágenes. Pero se ha utilizado en otras disciplinas a parte del reconocimiento de patrones y el procesamiento de imágenes, como en la biología, la psiquiatría, la psicología y la geología entre otras, pero nosotros nos centraremos en el reconocimiento de patrones.

El clustering permite a los científicos obtener, en una primera etapa, información sobre los datos. Los resultados de este análisis se pueden utilizar para iniciar una hipótesis sobre los datos, clasificarlos, hacer un test de su homogeneidad o comprimirlos.

No existe una única definición de cluster, y en cierto modo, la definición de cluster depende de la aplicación en que nos encontremos y del problema concreto que queramos resolver. Así, será el concepto que tenga *a priori* el usuario el que determinará qué significa cluster, y fijará los objetivos del método de cluster. En la literatura podemos encontrar entre otras las siguientes definiciones:

- Un cluster es un conjunto de entidades que son parecidas entre ellas, y que son diferentes a entidades de otros clusters.
- Un cluster es una agrupación de puntos en el espacio testeado tal que la distancia entre dos puntos del cluster es menor que la distancia entre un punto del cluster y otro que no pertenezca al mismo.
- Un cluster es un conjunto de regiones conectadas en un espacio p-dimensional que contienen una densidad relativa de puntos alta, separadas de otras regiones de alta densidad, por una región que contiene una baja densidad relativa de puntos.

Definir un algoritmo de clustering supone una serie de pasos que empiezan con definir la técnica para representar los datos a agrupar, determinar la medida de proximidad o distancia entre ellos, y el tipo de agrupación a realizar. En función de todas estas características obtendremos un tipo de clustering u otro. Así definir un clustering supone la definición de las siguientes etapas:

1. **Representación del patrón.**
   Consiste en escoger el número, tipo y escala de las características de los datos que se utilizarán como base de la representación de los mismos para realizar el clustering. Obtener estas características se puede hacer usando una o ambas técnicas que presentamos: La primera se llama, *escoger las características*, y supone identificar cuál es el subconjunto de características de los datos más efectivo para poder agrupar los mismos. La segunda se llama, *extraer las características*, y supone modificar las características existentes de los datos para obtener nuevas. La representación del patrón puede también incluir la definición del número de clases que se obtendrán con el clustering.
2. **Definición de una medida de proximidad.**
   Consiste en escoger una medida de proximidad entre los patrones que sea apropiada para el dominio. Normalmente es una función de distancia entre patrones.

3. **Definición del tipo de algoritmo de clustering.**
   Consiste en escoger un tipo de algoritmo. Existen diferente tipos. Unos en función del tipo de grupos obtenidos: si son concretos, o *hard*, es decir se producirá una partición de los datos, o si son difusos, o *fuzzy*, es decir cada patrón tendrá asociada una probabilidad de pertenecer a cada uno de los grupos resultado. También se pueden dividir entre *jerárquicos* o *particionales*, además de probabilísticos y de basados en teoría de grafos. Todos ellos serán explicados con mayor detalle en el apartado 3.4.

4. **Abstracción de los datos.**
   Es el proceso de extraer una representación del conjunto de datos de una manera compacta y simple. Normalmente esta representación supone tener una descripción compacta de cada cluster, como un representante o un prototipo.

5. **Validación de los resultados del clustering.**
   Consiste en ver si los clusters obtenidos son correctos. Existe tres tipos de validación: Una externa que compara la estructura obtenida con una realizada a priori. Una interna que valida si la estructura obtenida es apropiada para los datos. Y una tercera relativa que compara dos estructuras y mira cuál es mejor.

Antes de seguir con la explicación de los diferentes puntos que se han de tener en cuenta para definir un algoritmo de clustering vamos a definir algunos conceptos que necesitaremos a los largo de todo el capítulo:

Definición 3.1 Un *patrón* X es un dato que será usado por el algoritmo de clustering. Normalmente este dato consiste en un vector con m mediciones, $X = (x_1, \ldots, x_m)$.

Definición 3.2 Una *característica* es cada uno de los componentes escalares x_i de un patrón X.

Definición 3.3 La *dimensionalidad* de un patrón, $X = (x_1, \ldots, x_m)$, o del espacio de patrones, es su número de características, en este caso m.

Definición 3.4 Un *conjunto de patrones* se denota por $\mathcal{X} = \{X_1, \ldots, X_n\}$. El i-ésimo patrón del conjunto $\mathcal{X}$ se denota por $X_i = (x_{i1}, \ldots, x_{im})$. Muchas veces este conjunto de patrones se ve como una matriz de patrones de $n \times m$

Definición 3.5 Una *clase* es una fuente de patrones que sigue una distribución en el espacio de características con una densidad de probabilidad específica.

Definición 3.6 Una técnica de clustering *concreta o hard* asigna a cada patrón X_i una etiqueta de clase l_i que lo identifica. El conjunto de todas las etiquetas de un conjunto de patrones $\mathcal{X}$, es $\mathcal{L} = \{l_1, \ldots, l_n\}$ con $l_i \in \{1, \ldots, k\}$ donde k es el número de clusters.

Definición 3.7 Una técnica de clustering *difusa o fuzzy* asigna a un patrón de entrada X_i una probabilidad de pertenencia f_{ij} a cada uno de los clusters de salida j.

Definición 3.8 Una medida de *distancia* es una métrica o casi-métrica sobre el espacio de características que cuantifica la similitud de los patrones.

En los siguientes apartados vamos a presentar con mayor detalle los puntos clave del proceso de clustering. En el apartado 3.2 presentamos las características que pueden tener los datos que vamos a representar. El apartado 3.3 examina las diferentes medidas de distancia que pueden definirse sobre ellos. Los tipos de algoritmos de agrupación que pueden utilizarse se ven en el apartado 3.4, y la validación de los mismos en el apartado 3.5. En el apartado 3.7, explicaremos el algoritmo de clustering realizado en este trabajo así como algunos resultados obtenidos con el mismo. Para finalizar, el apartado 3.9, presenta las conclusiones de este apartado de clustering dentro de nuestro entorno de trabajo.

3.2 Características de los datos

Los algoritmos de clustering han de organizar los datos, y estos datos tienen diversas características que se pueden guardar en diferentes formatos. Escoger las características que definen los patrones es un punto clave para el buen funcionamiento del clustering. A continuación veremos las características de los datos, es decir, su tipo y escala, para continuar con su dimensionalidad.

3.2.1 Características de los datos numéricos

Los datos numéricos se caracterizan básicamente por su escala y su tipo:

- **Escala:** La escala de un número es su importancia relativa. Los tipos de escalas se pueden dividir en:
  - **Cualitativas:** Pueden ser:
    - Nominales: Son las escalas de números nominales o sin orden, como por ejemplo el color.
    - Ordinales: Son las escalas de números ordinales, como evaluaciones de temperatura, frío o caliente.
  - **Cuantitativas:** Son las escalas de:
    Valores continuos: como un peso o una longitud.
    Valores discretos: como el número de aristas de un polígono.
    Valores de intervalo: como la duración de un evento.
- **Tipo:** El tipo de los datos hace referencia a su grado de cuantificación. Así puede ser:
  - Binaria o bivaluada.
  - Discreta o multivaluada.
  - Continua.

3.2.2 Dimensionalidad

La dimensionalidad de los datos es el número de parámetros independientes que se necesitan para describirlos. Normalmente, para cada patrón, se cogen estos parámetros como valor de característica en los d ejes de coordenadas, siendo d el número de parámetros. Pero, muchas veces los datos se pueden proyectar en menos de d ejes, como se describe en [35]. Normalmente las características se escogen en función de la aplicación y de la experiencia previa que se tenga, pero es importante que su número sea lo más pequeño posible para facilitar su cálculo y la interpretación de los resultados. Además se ha de tener en cuenta que sean características lo más descriptivas y discriminatorias posible para poder diferenciar las diferentes clases de patrones. Así, tanto la selección de características como la extracción de las mismas tiene como objetivo mejorar el comportamiento de la clasificación así como su eficiencia computacional.

3.3 Medidas de distancia entre datos

Dado que escoger una buena medida de similitud entre patrones es fundamental en el proceso de clustering existen muchos tipos de distancia. Aquí vamos a comentar dos de las más utilizadas, pero una explicación más detallada y exhaustiva se puede ver en [36, 45].

- **Coeficiente de correlación**: Es un índice de similitud que se define como:

$$r(X,Y) = \frac{\sum_{k=1}^{m} x_k y_k - n m_X m_Y}{n S_X S_Y}$$

  Donde n es el número de patrones, m_X y m_Y son medias y S_X y S_Y son desviaciones estándares. Esta medida puede distorsionar la similitud cuando las características no tienen la misma escala. Para solucionarlo se pueden aplicar pesos.

- **Distancia de Minkowski**: Es una familia de medidas de disimilitud. Se define por:

$$d_p(X,Y) = \left(\sum_{k=1}^{m} |x_k - y_k|^p\right)^{\frac{1}{p}}$$

  en función del valor de p tenemos diversas distancias llamadas:

  - Distancia Euclídea ($p = 2$): Es una distancia intuitiva para objetos con 2 o 3 dimensiones. Funciona bien cuando el conjunto de datos tiene clusters compactos y aislados.
  - Distancia de Manhattan ($p = 1$)
  - Distancia "sup"($p \to \infty$)

  En general todas estas distancias de Minkowski tienen como problema la tendencia que tiene la característica con una escala mayor a dominar a las otras. Las soluciones que se aplican a este problema son la normalización de los datos con valores continuos, o la aplicación de pesos a las características.

3.3.1 Formatos para recoger los datos

Existen dos tipos de estructuras básicas para recoger los valores de los datos para poder ser agrupados en función de ellos, estos formatos deben poder representar la escala y el tipo de los datos y son:

Matriz de patrones: es una matriz de $n \times p$, ver Fig. 3.1, donde cada fila representa un patrón y cada columna una característica. Las p características se ven como un conjunto de ejes ortogonales, y cada patrón se representa como un punto en este espacio p-dimensional llamado espacio de patrones.

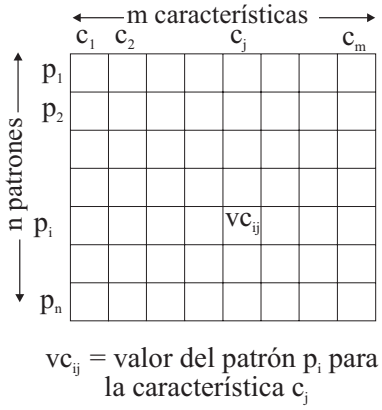

Figura 3.1: Matriz de patrones.

Matriz de proximidades: es una matriz de $n \times n$, ver Fig. 3.2, donde tanto las filas como las columnas representan patrones. Las entradas de la matriz dan una medida de proximidad entre todos los pares de patrones. Esta medida puede ser de similitud (matriz de similitud) o de disimilitud (matriz de disimilitud).

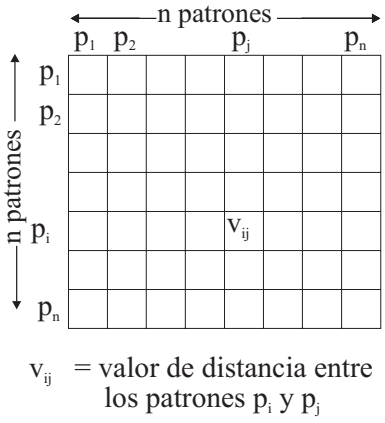

Figura 3.2: Matriz de proximidad.

3.3.2 Normalización y estandarización de los datos

Como hemos indicado en el apartado 3.3, a veces las diferentes características no tienen la misma escala y esto puede hacer que, en función del tipo de distancia que se escoja, algunas características tengan más influencia que otras en esta distancia. A veces esto ya es adecuado según la aplicación pero otras es necesario que todas las características tengan el mismo peso en esta función de distancia, para ello la matriz de patrones se normaliza para equilibrar las contribuciones de las diferentes características a la hora de hacer el cluster. En este apartado indicaremos con el superíndice $*$ que nos referiremos al valor no normalizado.

Existen diversos métodos de normalización:

- **Mover el origen del espacio de características al vector medio.** Esta normalización no afecta a la distancia Euclídea, y sólo simplifica las ecuaciones.

$$\forall i \in \{1, 2.., n\} \ \forall j \in \{1, 2, .., m\}$$

$$x_{ij} = x^*_{ij} - m_j$$

  Así la media será:

$$m_j = \frac{1}{n} \sum_{i=1}^{n} x^*_{ij}$$

  y la varianza

$$s_j^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x^*_{ij} - m_j)^2$$

- **La normalización z-score:** Divide cada característica por su desviación estándar s_j. Esta normalización extiende o comprime cada uno de los ejes de coordenadas para equilibrar las extensiones de todos ellos.

$$x_{ij} = \frac{x^*_{ij} - m_j}{s_j}$$

- **El método rango:** Reduce todas las características al rango [0 1], restando el valor más pequeño a cada columna, y dividiéndolo por el rango de cada columna. El mínimo y máximo se calculan para cada columna.

$$x_{ij} = \frac{x^*_{ij} - min\{x_{ij}\}}{max\{x_{ij}\} - min\{x_{ij}\}}$$

3.4 Métodos de clustering

La gran cantidad de características que se han de escoger para definir un algoritmo de clustering hace que una clasificación de los mismos sea difícil. A continuación vamos a presentar algunas clasificaciones de algoritmos de clustering en función de diversas características de los mismos, pero algunos de estos tipos se pueden dar a la vez. Así en función de qué característica del algoritmo estemos mirando podemos tener algoritmos de clustering:

1. Según si se sabe el número de clusters totales o no los podemos dividir en:

   - **Clustering jerárquico:** No sabemos el número de clusters final. Se crean diferentes niveles de clustering, o dicho de otra manera, generan una serie de particiones, estructuradas en una jerarquía. Dentro de este tipo de clustering podemos clasificarlos en función de si juntan elementos en grupos, o separan grupos:
     - **Aglomerativos:** Parten de que cada patrón forma un cluster y sucesivamente los va agrupando hasta que se cumple un criterio de parada.
     - **Divisorios:** Parten de tener todos los patrones en un mismo cluster y los van separando hasta que se cumple un criterio de parada.
   - **Cluster particional:** Parten de la base de que saben el número de clusters final que se producirá. a diferencia de los jerárquicos generan sólo una partición de los datos y no una jerarquía de particiones. El problema consiste en distribuir los patrones en los diferentes clusters maximizando una medida de similitud entre patrones. Dentro de este tipo de clustering podemos encontrar según un patrón pueda pertenecer a más de un cluster o sólo a uno:
     - **Con agrupación solapada:** Un patrón puede pertenecer a más de un cluster.
     - **Con agrupación exclusiva:** Un patrón sólo puede pertenecer a un cluster.

2. Según el método sea fijo o vaya aprendiendo:

   - **Adaptativos:** El método aprende mientras se va ejecutando, de manera que va modificando su criterio de similitud.
   - **No-adaptativos:** El método es fijo y está predeterminado.

3. Según utilicen todas las características a la vez para agrupar los datos o una a una:

   - **Algoritmos monotéticos:** Consideran las características de los datos una a una para ir particionando el espacio. Es decir, cogen una primera característica y dividen con ella el espacio, luego cada una de las particiones que ha obtenido la vuelven a dividir en función de una segunda característica, después las resultantes las divide con una tercera y así sucesivamente. El problema es que para un número elevado de características puede dar una fragmentación excesiva del espacio.
   - **Algoritmos politéticos:** Son los mas habituales, y consideran todas las características de forma simultánea para ir particionando el espacio.

4. Según el tipo de partición que realicen:

   - **Algoritmos hard:** Los algoritmos *hard* asignan cada patrón a un grupo o cluster determinado.

- **Algoritmos fuzzy:** Asignan, para cada patrón, un grado de pertenencia a cada uno de los clusters. Se podría convertir un resultado de clustering fuzzy en un resultado hard si se le asigna a cada patrón el cluster que tenga con mayor grado de pertenencia.

5. Según la forma de optimizar la función de error. Esta clasificación normalmente se da dentro de los algoritmos particionales, que pretenden optimizar una función de error.
   - **Algoritmos deterministas:** Optimizan la función de error usando una técnica tradicional, o exhaustiva.
   - **Algoritmos estocásticos:** Optimizan la función de error mediante una búsqueda aleatoria.
6. Según los datos a agrupar estén todos en memoria o no.
   - **Algoritmos incrementales:** Trabajan normalmente con conjuntos de datos muy numerosos, por ello la memoria que utilizan es un punto clave, así que utilizan una memoria secundaria para guardar los datos a agrupar y sólo pasan estos datos item a item a la memoria principal cuando necesitan examinarlos. De esta manera utilizan la memoria principal básicamente para guardar la información del clustering.
   - **Algoritmos no incrementales:** Tienen todos los datos a agrupar en memoria.
7. Según si tienen en consideración conocimiento básico del entorno de clustering.
   - **Algoritmos tradicionales:** No tienen en cuenta conocimiento básico del entorno sino solamente las distancias numéricas entre las características de los elementos. Obtienen clasificaciones que muchas veces son difíciles de interpretar de forma conceptual por un humano.
   - **Algoritmos conceptuales:** Llamados también, dirigidos por la semántica, o simbólicos. Se basan en que los elementos han de agruparse en clases que representen conceptos y no sólo agruparse en función de un criterio de similitud.
8. Según como utilicen la función de distancia para agrupar clusters.
   - **Single-link:** o de un sólo link, calcula la distancia entre dos clusters como la distancia mínima entre todos los pares de patrones de los dos clusters, cogiendo uno de cada cluster. Dos clusters se unen si cumplen un criterio de distancia mínima.
   - **Complete-link:** o de link completo, calcula la distancia entre dos clusters como la máxima entre todos los pares de patrones cogiendo un patrón de cada cluster. Dos clusters se unen si cumplen un criterio de distancia mínima.

A continuación vamos a ver, en los siguientes apartados, las clases de clustering que creemos más representativas con mayor detalle, así como ejemplos de algunos algoritmos concretos de clustering.

3.4.1 Clustering jerárquico

El objetivo de este proceso de clustering es agrupar los datos haciendo una secuencia anidada de agrupaciones de los mismos. Este método de cluster es un proceso matemático para crear una secuencia de particiones anidadas a partir de una matriz de proximidad.

Así definiremos el clustering jerárquico como:

Definición 3.9 Sea:
$\mathcal{X}$ un conjunto de n patrones.
C=$\{c_1, c_2, ..., c_t\}$ una partición de $\mathcal{X}$
tales que:

$$c_i \cap c_j = \emptyset, \quad \forall i, j \in [1..m]$$

$$c_1 \cup c_2 \cup ... \cup c_t = X$$

Un *clustering* es una partición C de $\mathcal{X}$, y sus componentes c_i se llaman *clusters*.

Un algoritmo de *clustering jerárquico* hace una secuencia de particiones anidadas que empieza con el clustering trivial, en el que cada patrón es un único cluster, y acaba con el clustering también trivial en el que todos los patrones forman un único cluster. Para definirlo de manera más formal:

Definición 3.10 Consideramos que una partición C_i está *anidada* dentro de una partición C_j si cada componente de C_i es un subconjunto de un componente de C_j, es decir C_j está formada juntando componentes de C_i.

Definición 3.11 Un *clustering jerárquico* de un conjunto de datos $\mathcal{X} = \{X_1, \ldots, X_n$ es una secuencia de particiones $C_1, \ldots, C_r$ tal que:
$C_1 = \{c_{11}, \ldots, c_{1n}\}$ donde $c_{1i} = \{X_i\}, i = 1, \ldots, n$.
$C_r = \{c_{r1}\}$, donde $c_{r1} = \{X_1, \ldots, X_n\}$.
Dadas dos particiones consecutivas C_i y C_{i+1}, C_i está anidada dentro de C_{i+1}.

Definición 3.12 Un *dendrograma*, ver Fig. 3.3, es un árbol binario que representa un clustering jerárquico. Cada nodo del árbol representa un cluster. Cada nivel del árbol representa un clustering. El Dendrograma representa n clusterings de los n patrones incluyendo los dos triviales.

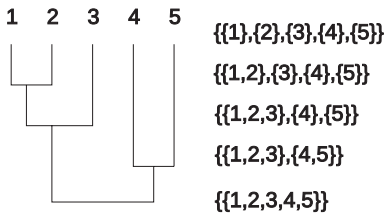

Figura 3.3: Dendrograma.

Proceso de clustering

Un método de clustering jerárquico general trata de crear un dendrograma a partir de una matriz de disimilitud dada. El dendrograma impone una medida de disimilitud, sobre los patrones, llamada **disimilitud de Cophenetic**. Esta medida se calcula de la siguiente forma:

Definición 3.13 Dados:

- Una secuencia de particiones del dendrograma: $\{C_0, C_1, ..., C_{n-1}\}$. Donde C_0 es el clustering donde cada patrón forma un cluster, y C_{n-1} es el clustering donde todos los patrones forman un único cluster.
- Los clusters en el *m-ésimo* clustering: $\{C_{m1}, C_{m2}, ..., C_{mn-m}\}$
- La función de nivel: L(m). Se define sobre las particiones como el nivel de disimilitud en el cual el clustering m se forma primero. Esta función depende del método de clustering.

La *Disimilitud de cophenetic* ($L_C(X, Y)$) entre dos patrones X e Y es:

$$d_C(X, Y) = L(k_{XY})$$

Donde

$$k_{XY} = min\{m : (X, Y) \in C_{mt},\ para\ algún\ t\}$$

Satisface la inecuación ultramétrica:

$$d_C(X, Y) \leq max\{d_C(X, t), d_C(t, Y)\}\ \ \forall(X, Y, t)$$

Algoritmos de clustering jerárquico

Este tipo de algoritmos pueden aplicarse de forma *aglomerativa* o *divisoria*:

En el caso de la forma *aglomerativa* el algoritmo comienza asignando cada patrón a un cluster diferente. Luego calcula las distancias de todos con todos, y ordena estas distancias de manera ascendente. Después, para cada una de las distancias d_i que se encuentra en la lista, genera un grafo de patrones relacionados que tengan un distancia menor o igual a d_i. De esta forma va creando cada uno de los niveles del dendrograma. El proceso para cuando todos los patrones están conectados en un grafo. Así se obtiene una jerarquía de grafos que puede ser cortada para cualquier valor de distancia d_i obteniéndose de esta forma la partición deseada.

El proceso *divisorio* se realiza en sentido contrario, es decir, comienza teniendo todos los patrones en un sólo cluster, y los va dividiendo en función de un criterio, hasta que obtiene un partición en que cada patrón es un cluster.

Además pueden estar relacionados con la teoría de grafos. Utilizando el *árbol de expansión mínima* de los datos y obteniendo los clusters como los subgrafos del mínimo spanning tree, en el caso del single-link, o como subgrafos maximal y completos en el caso del complete-link. Para más información ver [45].

3.4.2 Clustering particional

Este tipo de algoritmos producen una sola partición de los datos. Un problema que tienen es que deben determinar el número de particiones que han de hacer sobre los datos. Normalmente producen los clusters optimizando una función de criterio de forma local, dentro de un conjunto de patrones, o global, sobre todos ellos. En la práctica es prohibitivo probar todas las posibles combinaciones de agrupación de los datos, por ello se generan diversos resultados con diferentes estados de inicio y se selecciona el mejor resultado.

Existen diferentes tipos pero los más conocidos son los que usan el error cuadrático como función de criterio para generar los clusters. El error cuadrático de un clustering $\mathcal{C}$ de un conjunto de patrones $\mathcal{X}$ que contenga k clusters es:

$$e2(\mathcal{X},\mathcal{C}) = \sum_{j=1}^{k}\sum_{i=1}^{n_j} ||X_i^j - cen_j||^2$$

donde X_i^j es el patrón i-ésimo que pertenece al j-ésimo cluster, y cen_j es el centroide del j-ésimo cluster.

Dentro de esta familia de clusterings el más conocido es el algoritmo *k-means*. Empieza con una partición generada de forma aleatoria y continúa reasignando los patrones a los clusters en función de su distancia a los centros de los mismos hasta que se cumple un criterio de convergencia: por ejemplo hasta que no se reasigna ningún patrón de un cluster a otro o hasta que decrementa el error cuadrático. Es uno de los más populares por su sencillez y por tener una complejidad de $O(n)$. Su problema es que es sensible a la selección inicial de particiones y que puede converger en un mínimo local. Existen diferentes variantes de este algoritmo algunas intentan seleccionar buenas particiones iniciales. Otras permiten dividir clusters cuando su varianza es superior a un umbral dado, y unirlos cuando la distancia entre sus centroides es inferior a otro umbral. Esta variante posibilita que se obtengan buenas particiones partiendo de cualquier partición inicial. Un ejemplo de este tipo de algoritmo es el *ISODATA*, para más información sobre el mismo ver [15]. Otras variantes aplican programación dinámica, o diversos tipos de distancias, para mayor información ver [36, 43, 45].

También hay otros algoritmos de partición que utilizan la teoría de grafos. Es decir, construyen el minimal spanning tree y borran las aristas con longitudes mayores para generar los clusters.

3.4.3 Clustering basado en encontrar una distribución de los datos

Algunos algoritmos asumen que los patrones siguen una distribución y el objetivo es buscar los parámetros de la misma. Por ejemplo suponen que siguen una distribución gausiana y buscan sus parámetros.

3.4.4 Clustering basado en el vecino más cercano

Dado que la función de distancia entre clusters es un punto clave en los algoritmos de clustering, algunos se basan en agrupar un patrón dado con su vecino más cercano siempre y cuando la distancia a éste sea menor que un cierto umbral. Muchas veces este tipo de clustering podría llamarse *region growing* ya que va agrupando sólo los elementos que son vecinos. Nuestro algoritmo de clustering puede considerarse que pertenece a este grupo ya que agrupamos elementos vecinos, aunque esta vecindad a veces considera una cierta área alrededor del elemento.

3.4.5 Clustering basado en lógica difusa *fuzzy*

Normalmente los algoritmos de clustering asignan cada patrón a una clase concreta, pero a veces es deseable tener una probabilidad de pertenencia de un patrón a cada una de las clases existentes. Los algoritmos de clustering fuzzy dan como resultado un vector de probabilidades, para cada uno de los patrones a clasificar, que representa la probabilidad del patrón de pertenecer a cada una de las clases existentes. Normalmente los algoritmos de este tipo comienzan seleccionando una partición fuzzy de N patrones en K clases o clusters, seleccionando una matriz, F de $N \times K$. Cada elemento f_{ij} de la matriz F representa el grado de pertenencia del patrón X_i al cluster c_j, normalmente $f_{ij} \in [0, 1]$. Luego se calcula la función de criterio del clustering, por ejemplo el error cuadrático ponderado, que sería:

$$e^2(\mathcal{C}, F) = \sum_{i=1}^{N} \sum_{k=1}^{K} f_{ij} ||X_i - cen_k||^2 \tag{3.1}$$

donde

$$cen_k = \sum_{i=1}^{N} f_{ik} X_i$$

es el centro del cluster k-ésimo. Después se reasignan los patrones a los clusters para minimizar esta función y se vuelve a calcular F, para empezar el proceso de nuevo hasta que no haya cambios significativos. El resultado de un clustering fuzzy, es un conjunto de clusters, donde cada cluster es a su vez un conjunto de todos los patrones con una probabilidad de pertenencia al mismo, que puede ser 0.0. Se puede obtener un clustering hard a partir de uno fuzzy haciendo que cada patrón pertenezca al cluster para el que tiene una mayor probabilidad de pertenencia, siempre que ésta sea superior a un cierto umbral. En la Fig. 3.4 vemos un ejemplo de clustering fuzzy. Vemos dos clusters fuzzy F_1 y F_2, cada uno de ellos formados de la siguiente manera:

$$F_1 = \{(a, 0.8), (b, 0.8), (c, 0.9), (g, 0.75), (h, 0.7), (f, 0.2), (d, 0.1), (e, 0), (j, 0), (k, 0), (i, 0)\}$$

$$F_2 = \{(a, 0), (b, 0), (c, 0), (g, 0.15), (h, 0.2), (f, 0.6), (d, 0.7), (e, 1), (j, 0.9), (k, 0.9), (i, 0.8)\}$$

Así podemos obtener los hard clusters $H_1 = \{a, b, c, g, h\}$ y $H_2 = \{f, d, e, j, k, i\}$.

Existen diversos tipos de clustering fuzzy, uno de ellos es la adaptación del algoritmo particional k-means explicado en uno de los párrafos anteriores. Pero existen otros tipos, para ver un estudio comparativo de los mismos ver [12], una explicación más exhaustiva de los algoritmos de fuzzy clustering puede verse en [13, 43].

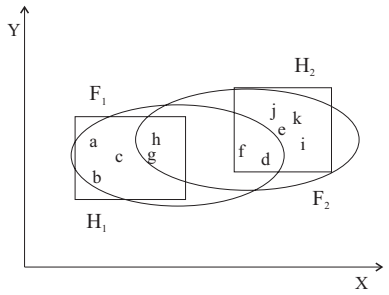

Figura 3.4: Clustering fuzzy.

3.4.6 Clustering basado en redes neuronales

Las redes neuronales se han utilizado tanto para clasificar como para hacer clustering, o agrupar. Normalmente generan un número fijo de nodos de salida y por tanto tienen un número limitado de clusters a generar. Para mayor información sobre este tipo de algoritmos ver [45].

3.4.7 Clustering basado en algoritmos genéticos

Los algoritmos genéticos se basan en generar varias soluciones, en este caso varias particiones de los datos, evaluarlas, y a partir de las mejores generar nuevas soluciones usando unos operadores de *selección*, *cruce* y *mutación*. Las soluciones se llaman *cromosomas*, y normalmente vienen representadas por vectores binarios. Así un algoritmo de este tipo empezaría generando una población de soluciones, o k-particiones, y asociando a cada solución un valor que indica lo acertada que es esa partición. Luego se escogen las mejores particiones para crear la nueva generación de soluciones. Esta generación se realiza con los operadores explicados anteriormente. El de selección, que selecciona dos particiones a partir de las cuales se generarán dos nuevos individuos. El de cruce, que indicará como cruzar esas dos particiones para generar las dos nuevas a partir de ellas. Normalmente este operador de cruce indica a partir de qué punto cortamos los cromosomas, para coger una parte de un padre y la otra del otro y generar dos nuevos hijos para la próxima generación, como se ve en la Fig. 3.5. Y por último acostumbra a utilizarse el operador de mutación que cambia el valor de una de las posiciones de los hijos, escogida de forma aleatoria.

Los principales problemas de los algoritmos genéticos son su sensibilidad a la selección de varios parámetros como la medida de la población o la probabilidad de cruce o mutación entre otros. Para más información ver [45].

3.4.8 Clustering estocástico

Normalmente las técnicas de búsqueda se pueden dividir entre las deterministas y las estocásticas. Las deterministas hacen una búsqueda exhaustiva de la solución pero tienen un coste elevado, mientras que las estocásticas generan una partición cercana a la óptima en un tiempo menor.

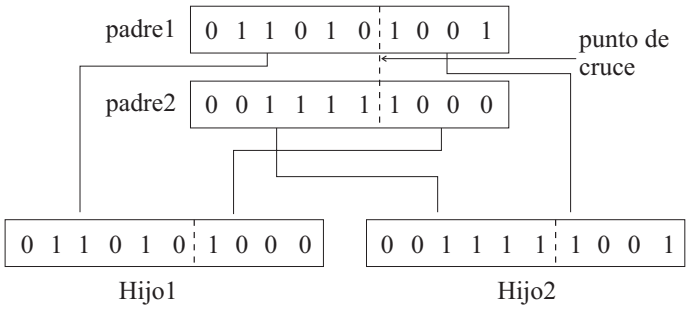

Figura 3.5: Operación de cruce en los algoritmos genéticos de clustering.

Dentro de las técnicas estocásticas secuenciales está el *simulated annealing*. La idea es evitar o recuperarse de soluciones que son mínimos locales, para ello aceptan, con una cierta probabilidad y para una nueva generación, soluciones peores a la que ya tienen. La probabilidad de aceptación de esta nueva generación peor se rige por un parámetro llamado de temperatura, que normalmente se define en términos de su valor inicial y final. Normalmente estos algoritmos funcionan de la siguiente manera: Primero seleccionan de manera aleatoria una partición y un elemento dentro de la misma, P_0, y calculan su error cuadrático, E_{P_0}, y seleccionan sus valores de temperatura inicial T_0 y final T_f, siendo $T_0 > T_f$. Luego seleccionan un vecino de P_0, P_1, y calculan el error cuadrático con este nuevo elemento en la partición, si el error es menor asignan este nuevo elemento a la partición y si no, si este error es inferior a T_0 también lo asignan. Se va repitiendo este proceso un número determinado de veces. Luego se reduce el valor de T_0, si esta reducción hace que $T_0 \leq T_f$ paramos y sino continuamos con el proceso desde la selección de un nuevo P_1.

Existe otro tipo de algoritmos que son también estocásticos como el *Tabu search*, que también está diseñado para evitar mínimos locales, y sistemáticamente impone o quita restricciones para permitir explorar regiones que de otro modo estarían prohibidas.

3.5 Validación e interpretación

El objetivo de esta fase es comprobar que los clusters, en que el algoritmo de clustering ha dividido los datos, son lo suficientemente significativos para hacer hipótesis sobre ellas, [108]. Pero lo que se conoce como el problema fundamental de la validación de clustering, es determinar el número real de clusters que existen en los datos. Y esta es una cuestión muy importante en la segmentación de imágenes.

El principal problema de esta fase es la dificultad para encontrar una definición de cluster y por tanto de validación sobre él. Lo que se intenta con la validación es responder a las siguientes preguntas:

- Es aleatoria la matriz de datos? Es decir se quiere determinar, antes de aplicar un algoritmo de clustering, si los datos están agrupados realmente en clusters o

están distribuidos de forma aleatoria. A esto se le llama la tendencia a cluster que tienen los datos. Algunos de los tests que se hacen para determinar esta tendencia están explicados en [108].

- Se adecúa la matriz de proximidad y la jerarquía establecida? Aquí se comprueba, una vez se ha aplicado un algoritmo de cluster jerárquico, si el dendrograma resultante se adecúa con la matriz de proximidad.
- Se valida una partición? Para responder a esta pregunta se toman medidas de separación entre clusters y de cohesión dentro de un mismo cluster.
- Qué clusters individuales son válidos? Una vez se ha comprobado que la estructura jerárquica se adecúa a los datos lo que se quiere saber es si los clusters formados son reales o no. Para ello se mide su compactación interna y su aislamiento respecto al resto de clusters. Mientras que la primera característica se mide según su medida y su instante de creación, la segunda se mide teniendo en cuenta la duración de su vida en el dendrograma.

Existen diversos tipos de tests de validación [35]:

- **Tests externos:** Comparan la estructura obtenida con un algoritmo de clustering, con una estructura que tiene *a priori*, intenta cuantificar la correspondencia entre las dos.
- **Tests internos:** No utilizan información *a priori*, sino que intentan determinar si la estructura es intrínsecamente apropiada a los datos.
- **Tests Relativos:** Este tipo de test compara dos estructuras para ver cuál es mejor. Por ejemplo, si tenemos una jerarquía de clusterings sobre unos datos, y queremos saber qué nivel escoger.

En general el clustering se ha de enfrentar a muchos problemas como: Encontrar un criterio de clustering que sólo reconozca los clusters que nos interesan, escoger el algoritmo de clustering, y validar los resultados de una manera objetiva. Este último es el problema más difícil de resolver.

3.6 Diferentes métodos de clustering en análisis de patrones

Hasta este punto se ha presentado una revisión del estado del arte respecto a la clasificación automática de datos o clustering. Dado que se trata de un paso clave en el trabajo que se presenta en esta tesis, antes de describir el algoritmo de clustering que se ha desarrollado vamos a presentar algunos de los métodos más relevantes relacionados con el trabajo presentado.

En Seong y otros [88] presentan un método de clustering que trabaja con grafos aleatorios con atributos. Dado un conjunto de grafos aleatorio, crean una jerarquía de conceptos de forma incremental utilizando una función de criterio, y una búsqueda *hill climbing*. El hecho de construir la jerarquía de forma incremental permite que puedan

tener una clasificación cada vez que entran un grafo nuevo a clasificar, y no necesitan tenerlos todos. La función de criterio que utilizan se basa en una minimización de la entropía. El resultado de este algoritmo es una jerarquía de clases conceptuales donde el nodo raíz representa a todos los grafos entrados, los nodos hojas representan a cada uno de los grafos aleatorios entrados y los intermedios representan el conjunto de hojas que está asociado a ellos. Cada clase se representa por medio de un grafo aleatorio.

En Wang [107] presenta un método de clustering de patrones que están representados por un vector. Tiene un conjunto de gramáticas isométricas de vector, ver apartado 4.2, y cada una representa una clase de patrones. Así dado un patrón representado por un vector calcula la distancia, a la que llaman d_a, del patrón a cada una de las gramáticas aumentadas con posibles distorsiones de su clase de patrones. Una vez calculadas todas las distancias asigna el patrón a la clase cuya distancia es menor. Puede darse el caso que varias clases den la misma distancia al patrón con lo cual se pasará a una segunda fase del algoritmo en que se calcula una segunda distancia, a la que llaman d_c, del patrón a cada una de las clases a las que ha sido asignado usando la distancia d_a, y se asigna a la que da la distancia d_c menor. La distancia d_c se calcula del patrón a las gramáticas que representan las clases de patrones sin permitir distorsiones.

En Rizzi [85] presenta un clustering jerárquico de grafos usando algoritmos genéticos. Los grafos a clasificar, representan en sus nodos puntos del espacio Euclídeo. Aplica este proceso a un entorno de autómatas móviles, donde el robot construye un grafo donde los nodos son lugares del mapa, y los arcos son caminos entre lugares. Una vez construido el grafo se agrupan sus lugares a múltiples niveles de abstracción en su representación, para decrementar la complejidad de los problemas de planificación de caminos sobre el mapa. La manera de generar cada uno de los niveles de la jerarquía del clustering es utilizando algoritmos genéticos. Primero definen una codificación de los clusters mediante un vector que indica los clusters y los elementos que los forman, luego evalúan los mejores resultados para ser reproducidos, al final obtienen un clustering para un determinado nivel y siguen el proceso en el que los elementos a agrupar en cada paso son los nodos del nivel anterior. En el nivel final todos los lugares del mapa se agrupan en un sólo cluster.

3.7 Un algoritmo de clustering para la segmentación de texturas estructuradas

Uno de los objetivos de esta tesis es la segmentación de texturas estructuradas. Como hemos explicado al inicio de este capítulo, dentro del trabajo que presentamos necesitamos segmentar zonas de un plano en las que haya este tipo de texturas. La detección de una textura estructurada supone la detección de dos características repetitivas: el patrón o texel, y la estructura. La detección de un patrón repetitivo normalmente se realiza mediante un procedimiento de clustering dentro de un espacio de características, ver [96], mientras que la detección de una estructura regular se ha solucionado en la literatura mediante, información de proyección o una función de distancia, ambas con matrices de coocurrencia (CM), ver [50] y [81] respectivamente, una matriz binaria

CM, ver [91], o un clustering, ver [96]. Nosotros proponemos una formulación común tanto para detectar el patrón que se repite como para la estructura, en términos de un clustering sobre un grafo, en el que las características de los elementos a agrupar se representan por medio de cadenas, como las explicadas en el capítulo 2. Como explicamos en el capítulo 1, dentro del entorno en que nos hallamos, que es el de dibujos de arquitectura, las texturas estructuradas pueden caracterizarse por medio de repeticiones de puntos, líneas, o polígonos, aunque aquí consideraremos polígonos, siendo las otras posibilidades fácilmente adaptables al algoritmo simplificando la función de distancia entre elementos a agrupar. Así dado un conjunto de polígonos se agrupan en función de su forma, y área, así como de su regla de emplazamiento, obteniendo las texturas estructuradas en los clusters en que aparecen un cierto número de elementos. Cada cluster se representa por medio de su área media, así como de su forma y su regla de emplazamiento medias calculadas a partir de todas las que están representando.

Como se explica en el capítulo 1 un plano de arquitectura, tras diversos procesos, lo representamos por medio de un grafo de adyacencia de regiones
$H = (V, E, LV, LE, FV, FV', FE)$ que representa los polígonos que aparecen en el plano así como sus relaciones de vecindad, y al que denotaremos por $\mathcal{H}(\mathcal{V}, \mathcal{E})$. Donde:

- $\mathcal{V}$ es el conjunto de nodos que corresponden a regiones.
- $\mathcal{E}$ es el conjunto de aristas que representan sus relaciones de vecindad. Denotamos por $(v, v') \in \mathcal{E}$ la arista que conecta dos regiones v, $v' \in \mathcal{V}$. Cada vértice del grafo, $v \in \mathcal{V}$, se representa por dos cadenas cíclicas con atributos, ambos atributos basados en cadenas se presentan en la Fig. 3.6, y se asignan según las dos funciones de atributos $\mathcal{LV}$ y $\mathcal{LV}'$.
  $\mathcal{LV}$ es la función que asigna etiquetas a los nodos del grafo. Estas etiquetas son números naturales que nos servirán para determinar en los diferentes procesos diversos estados de estos nodos, como visitados o no visitados, a qué cluster pertenecen etc.
  $\mathcal{LE}$ es la función que asigna etiquetas a las aristas del grafo. Estas etiquetas son números naturales que se utilizarán en los diversos procesos de segmentación y reconocimiento de símbolos para determinar si corresponden a vecindades dentro de un mismo cluster, o si las regiones son adyacentes o están separadas, etc.
- $\mathcal{FV}$ es una función de atributos $\mathcal{FV} : \mathcal{V} \rightarrow E^*$, que asigna a cada nodo la secuencia de líneas que forman el contorno del polígono, a partir del conjunto, E de segmentos del dibujo vectorizado.
- $\mathcal{FV}'$ es otra función de atributos que determina la otra cadena de atributos, $\mathcal{FV}' : \mathcal{V} \rightarrow C'^*$, y que asigna a cada nodo la cadena que representa la vecindad del polígono, es decir, representa la secuencia de líneas que conectan el centro del polígono representado por el nodo v y los respectivos centros de sus vecinos, estas líneas se obtienen del conjunto C' de segmentos que unen todos los centros de las regiones vecinas.
- $\mathcal{FE}$ es la función de atributos que, asigna a cada arista del grafo (v, v') el segmento de línea que conecta los centros de las regiones v y v'. Se define como $\mathcal{FE} : \mathcal{E} \rightarrow C'$, donde C' es el conjunto de segmentos conectando los polígonos

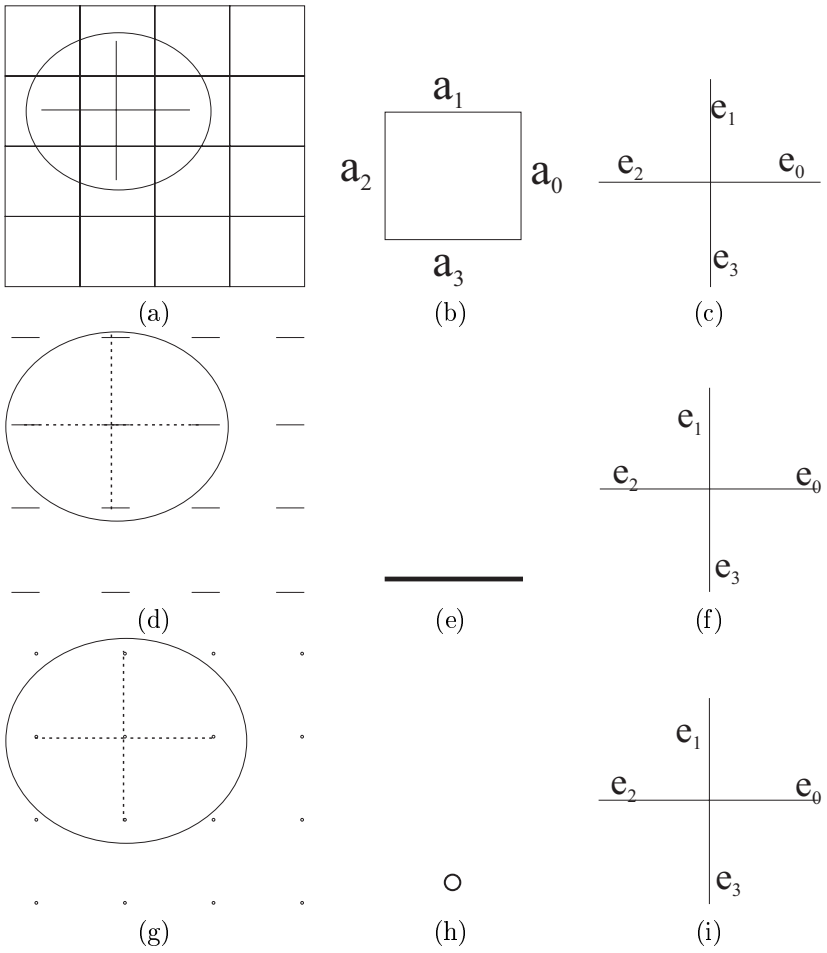

Figura 3.6: Tres casos distintos de textura estructural. (a)Textura. (b)Polígono($P = a_0a_1a_2a_3$). (c)Regla de emplazamiento ($R = e_0e_1e_2e_3$). (d)Textura. (e)Segmento de línea. (f)Regla de emplazamiento ($R = e_0e_1e_2e_3$). (g)Textura. (h)Punto. (i)Regla de emplazamiento ($R = e_0e_1e_2e_3$).

vecinos del grafo de adyacencia de regiones $\mathcal{H}$.

El proceso de reconocimiento compara la representación de una textura estructurada dada con cada una de las representaciones de las texturas segmentadas del RAG $\mathcal{H}(\mathcal{V},\mathcal{E})$. Esta segmentación se calcula por medio de un doble clustering jerárquico sobre el grafo $\mathcal{H}(\mathcal{V},\mathcal{E})$. Primero se agrupan polígonos con un área similar, en los clusters resultantes agrupan formas similares, generando nuevos clusters sobre los que se agrupan reglas de emplazamiento similares.

El proceso de clustering que vamos a explicar se enmarca dentro de los clustering jerárquicos, ya que se construye incrementalmente, y aglomerativo, ya que partiremos de un clustering en el que cada polígono será un cluster y los iremos agrupando. Además será monotético ya que utilizaremos primero una característica para agruparlos, es decir su área, luego sobre las particiones resultantes volveremos a clusterizar en función de su forma, para finalizar utilizando el atributo de vecindad. También corresponde a un algoritmo hard y no fuzzy, ya que queremos que nos de una partición del grafo en el que para cada polígono sepamos a qué cluster pertenece. También se basa en el vecino más cercano ya que agruparemos polígonos que sean vecinos.

Una vez determinadas las características principales del algoritmo de clustering que utilizamos vamos a definir algunos conceptos sobre este tipo de clustering jerárquico sobre grafos.

Definición 3.14 Sea $\mathcal{H}(\mathcal{V},\mathcal{E})$ el RAG de entrada. Sea $\mathcal{P}$ una partición de $\mathcal{V}$. Un *clustering* sobre $\mathcal{H}$ es una partición $\mathcal{P}$ donde cada cluster es un grafo conectado. Esta partición define un conjunto de subgrafos $\mathcal{C}_1(\mathcal{V}_1,\mathcal{E}_1),\ldots,\mathcal{C}_m(\mathcal{V}_m,\mathcal{E}_m)$, a los que llamaremos *clusters*.

Definición 3.15 Llamaremos *puente* entre dos clusters $\mathcal{C}_i$ y $\mathcal{C}_j$ al conjunto de aristas que conectan un vértice en $\mathcal{C}_i$ con un vértice en $\mathcal{C}_j$: $(\mathcal{C}_i,\mathcal{C}_j)=\{(v,v')\in\mathcal{E}|v\in\mathcal{V}_i\wedge v'\in\mathcal{V}_j\}$ para $i,j=1,\ldots,m, i\neq j$.

Definición 3.16 La *imagen* de $\mathcal{H}$ a través de $\mathcal{P}$ es el grafo $\mathcal{H}_\mathcal{P}(\mathcal{V}_\mathcal{P},\mathcal{E}_\mathcal{P})$, cuyos vértices y arcos son respectivamente los clusters y los puentes no vacíos inducidos por $\mathcal{P}$, $\mathcal{V}_\mathcal{P}=\{\mathcal{C}_1,\ldots,\mathcal{C}_m\}$, $\mathcal{E}_\mathcal{P}=\{(\mathcal{C}_i,\mathcal{C}_j)|(\mathcal{C}_i,\mathcal{C}_j)\neq,i,j=1,\ldots,m,i\neq j\}$.

Definición 3.17 Un *clustering jerárquico de altura n* es una secuencia de n clusterings, $\mathcal{H}^0(\mathcal{V}^0,\mathcal{E}^0),\ldots,\mathcal{H}^n(\mathcal{V}^n,\mathcal{E}^n)$, cada uno aplicado a la imagen del grafo generado por el clustering precedente, que produce una jerarquía de $n+1$ grafos incluyendo el original. Donde cada $\mathcal{H}^k$ es la imagen de $\mathcal{H}^{k-1}$ a través de la partición $\mathcal{P}^k$, $\forall k=1,\ldots,n$, y $\mathcal{H}^0$ es el grafo original.

La segmentación de texturas estructuradas se resuelve por medio de un proceso de clustering de grafo que puede agrupar formas o reglas de emplazamiento, simplemente redefiniendo la función de similitud entre vértices del grafo. El motor de clustering común a ambos está representado en el Algoritmo 3.7.1, y es como sigue:

1. Empieza dando un peso, $\frac{1}{n}\sum_{i=1}^{n}(dist(v,v_i))$, a cada $v\in\mathcal{V}^0$, donde n es el número de vecinos de v, $(v,v_i)\in\mathcal{E}^0$, y $dist(v,v_i)$ es una distancia entre vértices.

2. Para cada vértice, si no tiene un vecino superviviente se selecciona como tal el que tenga un peso mayor de entre todos ellos, incluido él mismo. Los vértices que no han sido seleccionados como supervivientes, v_j, se asocian al vecino superviviente más cercano, v_s, si $d(v_j, v_s)$ es menor que un umbral dado, y sino se convierten a su vez en supervivientes.

3. Cada superviviente junto con sus vecinos asociados es un cluster $\mathcal{C}_i^j(\mathcal{V}_i^j, \mathcal{E}_i^j)$ en el grafo actual $\mathcal{H}^j$ y un vértice $v_i^{j+1} \in \mathcal{V}^{j+1}$ en el siguiente $\mathcal{H}^{j+1}$.

4. Los atributos de cada v_i^{j+1} se calculan como la media de atributos de $v_i^j \in \mathcal{V}_i^j$.

5. Si $\mathcal{H}^n = \mathcal{H}^{n+1}$ el proceso se detiene y sino continúa en el paso 1.

Ahora vamos a ver como aplicamos este proceso de clustering a agrupar formas similares y luego formas con el mismo tipo de relaciones de emplazamiento.

3.7.1 Agrupación según forma: Clustering de formas 2D

El algoritmo de clustering adaptado para formas sería como sigue:

Las formas similares necesitamos que tengan, además, un área también parecida, por ello inicialmente se realiza un clustering por área que agrupa polígonos de área similar independientemente de su posición en el plano, pero utilizando el mismo algoritmo de clustering explicado. En él los nodos del grafo sólo tendrán dos vecinos, el de la franja de área inmediatamente superior y la inmediatamente inferior. A partir de estos clusters se realiza el primer proceso de clustering por formas. Lo primero que hacemos es recalcular las vecindades entre estos polígonos agrupados por área, considerando un cierto entorno de vecindad, y utilizando los polígonos de *Voronoi* para aquellos que han quedado aislados, como se describe en [89]. Para mayor claridad en la explicación consideraremos que la agrupación por formas parte del RAG original $\mathcal{H}(\mathcal{V}, \mathcal{E})$, aunque se aplicaría el proceso a cada uno de los subgrafos obtenidos de los clusters por área con las nuevas vecindades calculadas.

Así sea $\mathcal{H}(\mathcal{V}, \mathcal{E})$ el RAG de entrada. Una región de RAG $v \in \mathcal{V}$ se representa por medio de la cadena cíclica de atributos $\mathcal{LV}(v)$. Así la función de distancia que compara dos formas descritas por medio de las cadenas $\mathcal{LV}(v)$ y $\mathcal{LV}(v')$,

$$dist(v, v') = w_1 d(\mathcal{LV}(v), \mathcal{LV}(v')) + w_2 d_a(\mathcal{LV}(v), \mathcal{LV}(v')) \tag{3.2}$$

donde d_a es la diferencia aritmética entre las áreas de las cadenas, w_1 y w_2 son dos pesos constantes, y d es la distancia de edición de cadenas explicada en la sección 2.3.3.

Dadas dos cadenas X e Y, $d(X, Y)$ es, hablando de una manera informal, el coste mínimo de transformar X en Y usando ciertas operaciones de edición. Como en nuestro caso las formas pueden tener sus aristas cortadas en diversos fragmentos, necesitamos la *operación de edición de fusión* que permite transformar una secuencia completa de símbolos por otra, entendiendo por símbolos cada uno de los elementos de la cadena. El atributo de un vértice $v_i^{j+1} \in \mathcal{H}^{j+1}$, $v_i^{j+1} = \mathcal{C}_i^j(\mathcal{V}_i^j, \mathcal{E}_i^j)$, es una cadena que representa todas las formas en $\mathcal{V}_i^j$, y se calcula como la forma media del grupo

Algoritmo 3.7.1 Algoritmo de clustering

entrada: RAG $\mathcal{H}(\mathcal{V},\mathcal{E})$
salida: Clustering jerárquico $\mathcal{H}^0(\mathcal{V}^0,\mathcal{E}^0),\ldots,\mathcal{H}^n(\mathcal{V}^n,\mathcal{E}^n)$
método:
$P=0$
$\mathcal{H}^P=\mathcal{H}(\mathcal{V},\mathcal{E})$
Repetir
  Para cada $v\in\mathcal{V}^P$ **hacer**
    $peso(v)=0.0$
    Para $v_i=PrimerVecino(v)$ hasta $UltimoVecino(v)$ **hacer**
      $peso(v)=peso(v)+dist(v,v_i)$
    FinPara
    $peso(v)=\frac{1}{NumVecinos(v)}peso(v)$;
  FinPara
  Para cada $v\in\mathcal{V}^P$ **hacer**
    $VertMayor=v$
    Para $v_j=PrimerVecino(v)$ hasta $UltimoVecino(v)$ **hacer**
      Si $peso(v_j)>peso(VertMayor)$ **Entonces**
        $VertMayor=v_j$
      Fin Si
    FinPara
    $Superviviente(VertMayor)=TRUE$;
  FinPara
  Para cada $v\in\mathcal{V}^P$ **hacer**
    Si $No(Superviviente(v))$ **Entonces**
      $DistanciaVecinoSuperv=INFINITO$
      Para $v_j=PrimerVecino(v)$ hasta $UltimoVecino(v)$ **hacer**
        Si $Superviviente(v_j)$ **Entonces**
          Si $D=d(v_j,v_s)<umbral$ **Entonces**
            $DistanciaVecinoSuperv=D$
            $VecinoSuperviviente=v_j$
          Fin Si
        Fin Si
      FinPara
      Si $DistanciaVecinoSuperv<INFINITO$ **Entonces**
        $Padre(v)=VecinoSuperviviente$
      Sino
        $Superviviente(v)=TRUE$
      Fin Si
    Fin Si
  FinPara
  Para cada $v\in\mathcal{V}^P$ **hacer**
    Si $Superviviente(v)$ **Entonces**
      Creamos un nodo v_i^{P+1} en el nuevo Grafo $\mathcal{H}^{P+1}$
      Atributos(v_i^{j+1})=Media(Atributos($v_i^j\in\mathcal{V}_i^j$)).
    Fin Si
  FinPara
  $P++$
Hasta que: $\mathcal{H}^{P-2}==\mathcal{H}^{P-1}$

de formas $v' \in \mathcal{V}_i^j$. El algoritmo que utilizamos para calcular esta forma media está descrito en el apartado 2.4.

El proceso de clustering explicado anteriormente empieza con $\mathcal{H}^0(\mathcal{V}^0, \mathcal{E}^0) = \mathcal{H}(\mathcal{V}, \mathcal{E})$. Al final los clusters de $\mathcal{H}^n$ definen conjuntos de regiones en $\mathcal{H}^0$ que tienen una forma y área similares.

3.7.2 Agrupación según estructura: Clustering de vecindades

Después del clustering de formas, cada vértice en $\mathcal{H}^n(\mathcal{V}^n, \mathcal{E}^n)$ es un cluster que representa polígonos con una forma y área similares. Ahora el problema es agrupar formas que tengan además la misma regla de emplazamiento. Para ello se calcula un segundo clustering usando el mismo algoritmo.

Se parte de un grafo inicial $\mathcal{H}_i^0(\mathcal{V}_i^0, \mathcal{E}_i^0)$ para cada $v_i \in \mathcal{V}^n$. A cada vértice $v \in \mathcal{V}_i^0$ se le asigna como atributo una cadena que representa sus vecindades $\mathcal{LV}'(v)$, que es la misma que está explicada en el capítulo 1, como atributo de vecindad de un polígono del RAG inicial $\mathcal{H}$. Estas vecindades se consideran de diferentes maneras, primero consideramos sólo las vecindades que están en un entorno dado del polígono, normalizándolas a valores de ángulo de 0 a π radianes, y considerando que dos vecindades con una diferencia de ángulo de π radianes son en realidad la misma. En un segundo paso se consideran las obtenidas a partir de los polígonos de *Voronoi*. Para cada grafo $\mathcal{H}_i^0$, tenemos una cadena $S_i = \mathcal{LV}(v_i)$ que representa su forma media, y por tanto la forma media de todos los nodos que están en el grafo. La distancia entre dos vértices v y v' se calcula como $dist(v, w) = d(\mathcal{LV}'(v), \mathcal{LV}'(v'))$, donde d es la distancia ilustrada en la Fig. 2.2. La operación de fusión explicada para la sección anterior no tiene sentido cuando nos referimos a vecindades, y por ello se utilizan las tres operaciones de edición básicas explicadas en el apartado 2.3.3: *sustitución*, *inserción* y *borrado* de los elementos de la cadena. El atributo de un cluster $\mathcal{C}_i^j(\mathcal{V}_i^j, \mathcal{E}_i^j)$ es una cadena que representa la regla de emplazamiento, y se calcula también como la media de entre todas las reglas de emplazamiento $v \in \mathcal{V}_i^j$, usando el mismo algoritmo del cálculo de la cadena media descrito en el apartado 2.4. Pero ahora el coste asociado a cada una de las operaciones de edición es como sigue. Sean A y B dos cadenas que representan dos vecindades distintas, con una longitud total de los segmentos que las forman de l_A y l_B respectivamente. La sustitución de un segmento que representa una vecindad en A, a, por otro que representa una vecindad en B, b, siendo l_a y l_b sus respectivas longitudes, y ϕ_a y ϕ_b sus respectivos ángulos, tiene el coste asociado siguiente:

$$c(a \to b) = \frac{|\phi_a - \phi_b|}{\pi} + |\frac{l_a}{l_A} - \frac{l_b}{l_B}| \tag{3.3}$$

$$c(a \to \lambda) = w_e \frac{l_a}{l_A} \tag{3.4}$$

$$c(\lambda \to a) = w_i \frac{l_a}{l_A} \tag{3.5}$$

Al final del proceso se obtienen diversos grafos $\mathcal{H}_i^{n_i}(\mathcal{V}_i^{n_i}, \mathcal{E}_i^{n_i})$, y cada nodo $v \in \mathcal{V}_i^{n_i}$ que agrupa un cierto número de polígonos en $\mathcal{H}(\mathcal{V}, \mathcal{E})$ es una textura estructurada. Esta textura estructurada se representa por medio de dos cadenas: el texel medio

S_i que se repite en la textura, y la regla de emplazamiento media representada por medio de la cadena media $\mathcal{LV}'^{n_i}_i(v)$. Un ejemplo de este proceso de clustering global, tanto para formas como para reglas de emplazamiento puede verse en la Fig. 3.8.

A veces, debido a las distorsiones existentes en la imagen, una misma textura puede verse fragmentada en diversos clusters por quedar sus vecindades modificadas, un ejemplo de ello lo tenemos en la Fig. 3.7. Tendremos dos clusters uno formado por los polígonos $\{P1, P2, P3, P4, P6, P7, P8, P9, P11, P12\}$, y otro por $\{P5, P10\}$. De ellos los primeros tendrán como vecindad media calculada la que aparece en la Fig. 3.7(b), mientras que los segundos la que aparece en la Fig. 3.7(a). Vemos que la segunda vecindad incluye a la primera, es decir tiene las aristas existentes en la primera y alguna más. Además también podemos ver que la vecindad que existe de más es causada por una arista cuya longitud es una fracción muy pequeña del perímetro del polígono. Todo ello conduce a pensar que es una distorsión y permite unir estos clusters en uno. Así una vez calculados los clusters se realiza un proceso de fusión de clusters con vecindades, unas incluidas dentro de otras, siempre que sean debidas a una fracción pequeña de su perímetro. Podemos definir formalmente este concepto de la siguiente manera:

Definición 3.18 Dadas dos vecindades definidas por las cadenas $V_1 = \{v_{11} \dots v_{1n}\}$, $V_2\{v_{21} \dots v_{2m}\}$ decimos que V_1 está *incluida* en V_2, $V_1 \subseteq V_2$, si $\exists j$ tal que $v_{1i} \approx v_{2j}$ $i = 1, \dots, n$.

El representante del cluster final será la vecindad media calculada para el mayor número de elementos. Así en el ejemplo presentado en la Fig. 3.7(a), unimos todos los polígonos en un solo cluster y dejamos como vecindad media la presentada en la Fig. 3.7(b).

3.7.3 Texturas formadas por más de un tipo de polígono

En el proceso de clustering explicado tendremos un cluster por cada grupo de polígonos similares emplazados según una regla común. Pero cuando la textura está formada por más de un texel nos producirá un cluster por cada texel. Estos clusters que corresponden a una misma textura estarán solapados en el espacio. Por tanto necesitamos comprobar para cada cluster si se solapa con alguno y si lo hace determinar cuantos están componiendo la textura. Representando a la misma por medio de sus formas medias y sus vecindades medias.

3.8 Resultados

El algoritmo se ha testeado utilizando varios dibujos de arquitectura, la Fig. 3.9 ilustra uno de ellos. La Fig. 3.9(a) presenta la vectorización con las texturas estructuradas reconocidas marcadas con líneas en trazo grueso. Se pueden observar dos texturas que abarcan una zona considerable, ambas correspondientes a tejados, que como explicamos en el capítulo 1 es un símbolo texturado. En la parte de arriba y en la mitad de la imagen los tejados tienen las tejas en una posición horizontal, mientras que en la parte inferior izquierda sus tejas están dispuestas en posición vertical. El texel y la

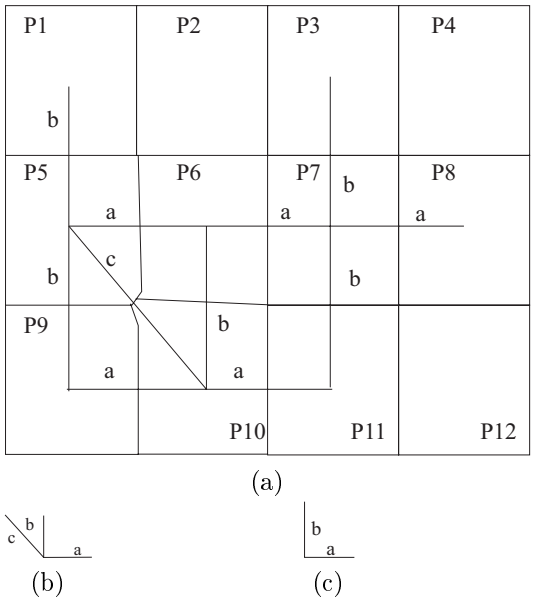

Figura 3.7: (a)Vecindades con distorsión. (b)Vecindades Polígono $P5$, $P10$. (c)Vecindades Polígonos $P1$, $P2$, $P3$, $P4$, $P6$, $P7$, $P8$, $P9$, $P11$, $P12$.

regla de emplazamiento calculadas como cadenas medias que representan los símbolos de tejados con tejas colocadas de forma horizontal se muestran en la Fig. 3.9(b) y la Fig. 3.9(c) respectivamente, mientras la representación, texel y regla de emplazamiento, del tejado con tejas verticales aparece en las Fig. 3.9(d) y Fig. 3.9(e) respectivamente. Podemos observar que en ambos casos son las representaciones que esperamos visualmente aunque vemos que algunas tejas no han sido segmentadas debido a distorsiones. Otro símbolo texturado que encontramos en la imagen es la escalera. La estructura de la textura segmentada que obtenemos se observa en las Fig. 3.9(f) y Fig. 3.9(g). Dado que el símbolo de la escalera aparece distorsionado por algunos agujeros y líneas adicionales, el proceso de clustering no lo puede segmentar completamente. Veremos más resultados de este algoritmo de clustering en el capítulo 5 donde explicamos el proceso general.

3.9 Conclusiones

Uno de los pasos para la segmentación de texturas estructuradas es la agrupación de formas similares dentro de la imagen distribuidas de forma regular. En este capítulo hemos unificado ambos problemas (detección de forma que se repite y regla) bajo un mismo algoritmo de agrupación o clustering. Tras hacer un repaso de los conceptos fundamentales del problema de clustering así como de los métodos más utilizados, hemos presentado la aportación de esta tesis en relación a éstos.

Hemos visto como una textura estructurada puede segmentarse agrupando polí-

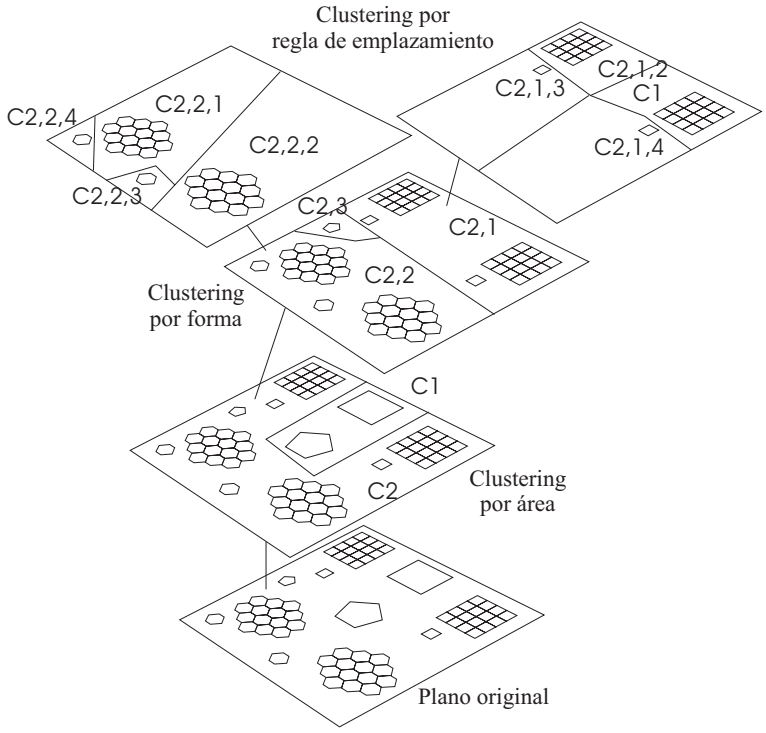

Figura 3.8: Proceso de Clustering.

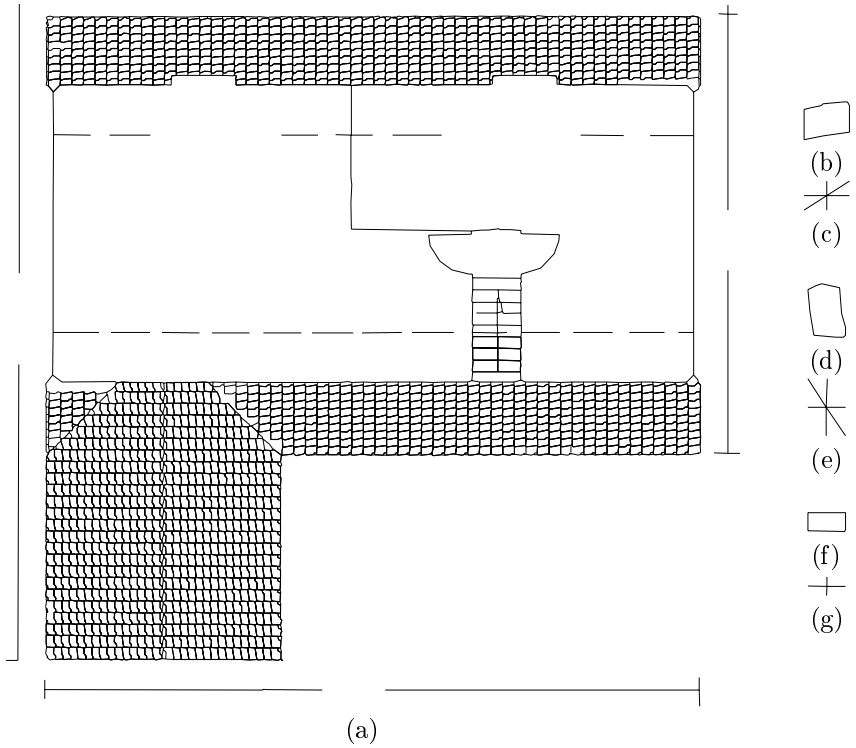

Figura 3.9: (a)Imagen vectorizada (texturas en líneas gruesas). (b)Texel del tejado horizontal. (c)Regla de emplazamiento de (b). (d)Texel del tejado vertical. (e)Regla de emplazamiento de (d). (f)Texel de las escaleras. (g)Regla de emplazamiento de (f).

gonos similares que sigan una regla de emplazamiento similar. Este proceso de agrupación o clustering lo hemos realizado sobre un grafo de adyacencia de regiones sobre el que teníamos representados los polígonos que aparecían en el plano y sus relaciones de vecindad. Tanto la forma como las relaciones de vecindad las hemos representado usando cadenas cíclicas de atributos y su distancia se calcula usando la distancia de edición de cadenas. Para cada cluster se calcula el representante de la textura que representa, calculando la forma media, como una media de cadenas de todas las cadenas que representan los polígonos agrupados en el cluster, y la regla de emplazamiento media, calculada también como la media de cadenas de las relaciones de vecindad de todos los polígonos que están agrupados en el cluster. Estas dos medias de cadenas se realizan mediante el algoritmo de cálculo de la cadena media explicado en el apartado 2.4.

Una vez tenemos la representación de una textura estructurada podemos reconocer si es una de las texturas dadas comparando sus representaciones. Este proceso se puede realizar usando una distancia de edición de cadenas que sea invariante a rotación. En realidad sería como la distancia descrita en el apartado 2.3.3, pero en vez de considerar la orientación de cada una de las aristas de un polígono respecto a la horizontal, la consideraríamos respecto a la arista anterior del mismo polígono, de esta manera tendríamos una representación del polígono invariante a rotación y por tanto su distancia de edición de cadenas también lo sería.

Aunque el algoritmo tolera distorsiones en las formas y reglas de emplazamiento, falla en presencia de agujeros en la textura o líneas adicionales. Esto sugiere algún proceso de relajación para encontrar estos texels, distorsionados, cortados o agrupados, que permita adaptar el texel calculado o la regla de emplazamiento a ellos. Además cuando las texturas están formadas por más de un texel necesitamos saber, no sólo las vecindades entre polígonos de una misma clase, sino las vecindades interclases. La manera que tenemos de aceptar estas distorsiones mayores, y de representar vecindades entre clases, es la utilización de una gramática con corrección de errores como la que explicamos en el capítulo siguiente.