Curso 2009/10
CIENCIAS Y TECNOLOGÍAS/9
I.S.B.N.: 978-84-7756-941-1

JESÚS MIGUEL TORRES JORGE

Reconocimiento gestual mediante técnicas avanzadas de visión por computador

Director
JOSÉ DEMETRIO PIÑEIRO VEGA

SERVICIO DE PUBLICACIONES
UNIVERSIDAD DE LA LAGUNA

SOPORTES AUDIOVISUALES E INFORMÁTICOS
Serie Tesis Doctorales

No existe la emoción, sólo existe la paz.
No existe la ignorancia, sólo existe el conocimiento.
No existe la pasión, sólo existe la serenidad.
No existe el caos, sólo existe la armonía.
No existe la muerte, sólo existe La Fuerza.

(Código Jedi)

A Rocío, mi compañera.
A Luna, Quillo, Niña,
Ami y Pitu.

A mis padres y a mi hermano,
Miguel, Carmen Marina
y Ángel

Agradecimientos

En primer lugar quiero agradecer al Dr. D. José Demetrio Piñeiro Vera su apoyo y su guía durante la realización de este trabajo. Es indudable que sin su inestimable ayuda y su implicación, mucho más allá de lo común, esta tesis nunca hubiera sido terminada.

En segundo lugar quiero agradecer al Dr. D. Lorenzo Moreno Ruiz el abrirme las puertas de la carrera universitaria como docente y al Dr. D. Leopoldo Acosta Sánchez su apoyo, sus ánimos y sus buenos consejos durante estos últimos años.

También quiero agradecer al Dr. D. Evelio José González González su paciencia y su implicación, que ha sido casi providencial, y sin la cual el tiempo nos hubiera ganado la partida.

Y por supuesto, también quiero agradecer a todos los miembros del Departamento de Ingeniería de Sistemas y Automática y Arquitectura y Tecnología de Computadores su ayuda. En especial a: D. Iván Castilla Rodríguez, D. Pedro Antonio Toledo Delgado, D. Jonatan Felipe García y D. Yeray Callero de León; por haber compartido con humor el pasado verano y haber asumido de buen grado algunas de mis responsabilidades para que yo pudiera terminar este trabajo. Además de al resto de mis compañeros: Dr. D. José Luis Sánchez de la Rosa, Dr. D. Alberto Francisco Hamilton Castro, Dr. D. Juan Albino Méndez Pérez, D. Juan Julián Merino Rubio, Dra. Dña. Rosa María Aguilar Chinea, Dr. D. Graciliano Nicolás Marichal Plasencia, Dr. D. José Ignacio Estévez Damas, Dr. D. José Francisco Sigut Saavedra, Dr. D. Roberto Luis Marichal Plasencia, Dra. Dña. Marta Sigut Saavedra, Dr. D. Santiago Torres Álvarez, Dra. Dña. Carina Soledad González González, Dr. D. Evelio José González González, Dra. Dña. Silvia Alayón Miranda, Dra. Dña. Vanesa Muñoz Cruz, Dr. D. Jonay Tomás Toledo Carrillo, D. Germán Carlos González Rodríguez, D. Carlos Alberto Martín Galán, D. Hector Javier Reboso Morales, D. Sid Ahmed Ould Sidha, D. Ginés Coll Barbuzano, D. Eladio Hernández Díaz, D. Néstor Morales Hernández, D. Rafael Arnay del Arco, D. Jesús Javier Espelosin Ortega, D. Daniel Perea Ström y Dña. Ángela Hernández López porque todos, en mayor o menor medida, han

contribuido en algo a esta tesis.

No puedo concluir mis agradecimientos sin antes dedicar unas palabras a mis seres más queridos.

Agradezco a Rocío, mi compañera y amiga, su apoyo durante todo este tiempo. La elaboración de una tesis no es fácil, ni para quién la hace ni para sus seres queridos, y eso es algo que merece ser reconocido. Ella ha sabido mantenerse firme, animándome a seguir adelante en los momentos más difíciles y por eso quiero agradecerle su apoyo de todo corazón.

Y por último quiero agradecer a mis padres las oportunidades y la ayuda que me han brindado durante todos los años de mi vida. No dudo que si ellos no hubieran sido como fueron ni hubieran hecho lo que hicieron, yo no hubiera tenido la oportunidad de escribir estos agradecimientos. A ambos, muchas gracias por haber sabido tensar el arco...

Índice

Índice de tablas

Índice de figuras

Índice de algoritmos

Índice de abreviaturas

Articulaciones de la mano

Símbolos y notación

a	escalar
$\lvert a \rvert$	valor absoluto de a
$\boldsymbol{a}$	vector columna
a_i	i-ésimo elemento del vector $\boldsymbol{a}$ o del conjunto de escalares $\{a_n\}$
$\mathbf{1}$	vector columna $[1, 1, \ldots, 1]^T$
$\lVert \boldsymbol{a} \rVert$	norma euclídea de $\boldsymbol{a}$
$\boldsymbol{a}^T$	traspuesta de $\boldsymbol{a}$
$\mathbf{A}$	matriz
$\boldsymbol{a}_i$	i-ésimo elemento de conjunto de vectores $\{\boldsymbol{a}_n\}$ o i-ésima columna de la matriz $\mathbf{A}$
a_{ij}	elemento en la fila i-ésima y columna j-ésima de la matriz $\mathbf{A}$
$\mathbf{I}$	matriz identidad
$\mathbf{A}^{-1}$	inversa de la matriz $\mathbf{A}$
$\lVert \mathbf{A} \rVert_F$	norma de Frobenius de la matriz $\mathbf{A}$
$\text{diag}(\boldsymbol{a})$	matriz diagonal cuyo i-ésimo elemento de la diagonal es a_i
$\det(\mathbf{A})$	determinante de la matriz $\mathbf{A}$
$\text{tr}(\mathbf{A})$	traza de la matriz $\mathbf{A}$
$[\ldots]$	vector o matriz
$\{\ldots\}$	conjunto o lista de elementos

X, Y, Z	ejes de coordenadas en $\mathbb{R}^3$
x, y	ejes de coordenadas en $\mathbb{R}^2$
$\mathbb{F}$	espacio de características
$f(x)$	función f en x
$f(x; p)$	función f en x con parámetro p
$\hat{a}$	estimación de a
$\langle a \rangle$	media del conjunto $\{a_n\}$
$\tilde{a}_i$	i-ésimo elemento del conjunto $\{a_n\}$ al que se le ha restado la media de dicho conjunto
$a^{(t)}$	valor de a en la iteración t
$\mathrm{var}(a)$	varianza de a
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	distribución normal multivariable de media $\boldsymbol{\mu}$ y varianza $\boldsymbol{\Sigma}$
$D_{KL}(\mathcal{P} \Vert \mathcal{Q})$	divergencia de Kullback-Liebler entre las distribuciones de probabilidad $\mathcal{P}$ y $\mathcal{Q}$

Introducción

El presente trabajo se encuadra en una línea de investigación en torno a la Visión Artificial y Aprendizaje Automático que siguen varios miembros del Departamento de Ingeniería de Sistemas y Automática y Arquitectura y Tecnología de Computadores desde hace algunos años. Dicha línea es la continuación lógica de otra con más tradición en el grupo —ya que data de la propia creación del mismo— en la que se tratan temas en torno al reconocimiento de patrones, el aprendizaje automático y varias técnicas agrupables bajo el nombre genérico de *soft-computing*.

El objetivo principal de este trabajo es contribuir con técnicas y estrategias al problema del reconocimiento gestual mediante visión por computador. Éste es un problema extremadamente complejo de resolver en toda su generalidad, representando esta memoria un paso más hacia ese objetivo final, en la citada línea de investigación. Por ello, hay que entenderla como una foto que refleja el estado del trabajo realizado en un momento puntual de un camino mucho más largo y difícil.

El estudio se ha centrado particularmente en la identificación de la postura de la mano. Este problema ha sido elegido por su complejidad y por la importancia y aplicabilidad inmediata de su solución para el desarrollo de interfaces hombre-máquina. A pesar de ello, hay que destacar que las técnicas desarrolladas en este trabajo son totalmente generales —con la excepción de una parte dedicada específicamente al modelado de la mano— siendo aplicables a la reconstrucción 3D a partir de imágenes de cualquier tipo de objetos, tanto articulados como

deformables.

Como primera aproximación, se analizó la familia de técnicas de reconstrucción 3D más prometedora, los métodos de factorización. Estos métodos recuperan la forma y el movimiento a partir de características identificadas a lo largo de una secuencia de imágenes. Entre ellos existe un gran número de versiones adaptadas para funcionar con diferentes tipos de objetos, siendo un campo de investigación en expansión en los últimos años. Para su análisis, se elaboró un sistema de seguimiento de características en tiempo real con el objetivo de proporcionar los datos que necesitan estos algoritmos.

La creación de un modelo foto-realista cinemático de la mano fue otro de los objetivos. Las restricciones en las posturas de la mano humana fueron incorporadas al modelo, reflejando la realidad anatómica. Ese modelo simplifica el montaje experimental necesario y posibilita incorporar la dependencia entre los parámetros de la postura y la apariencia de la mano, sin necesidad de engorrosos y caros sistemas de sensores por hardware.

Los métodos basados en kernel proyectan los datos de un espacio de entrada en otro espacio de mayor dimensión, a través de una transformación no lineal. La idea es que esta transformación haga que las relaciones entre las observaciones sean lineales en el nuevo espacio, aunque en el espacio de entrada no lo fuesen. Estos métodos se proponen como posible solución para el problema del reconocimiento gestual debido a las siguientes propiedades: mapean los datos en nuevos espacios donde se destacan las características deseables para la discriminación de diferentes posturas, permiten —en algunos casos— reducir la dimensión del problema original y son robustos a perturbaciones como oclusiones y ruido. La forma de aplicar estos métodos al reconocimiento gestual es o bien buscando identificar configuraciones de puntos similares en el nuevo espacio o comparando directamente los subespacios generados por cada conjunto de ejemplos similares. Para ello, se considera la totalidad del conjunto de puntos de una mano en una imagen o se caracteriza independientemente cada segmento rígido que la compone.

En esta memoria se encuentran, tanto contribuciones desde un punto de vista

teórico, como resultados prácticos experimentales. No se han descuidado los aspectos computacionales, ya que en el desarrollo y prueba de las técnicas se han presentado tareas de carga computacional muy elevada y, además, también se ha tratado algún problema con restricciones de tiempo real. Al final ha sido necesario plantear el estudio de los mejores algoritmos para resolver cada una de estas tareas, para poder progresar hacia la conclusión de las mismas.

La estructura de la memoria por capítulos es la siguiente:

En el capítulo 1 se describe el reconocimiento gestual desde el punto de vista de su utilidad como interfaz hombre-máquina y se pasa a tratarlo como un problema de visión por computador. Se analizan los orígenes de la dificultad de este tipo de problemas y se hace una revisión somera de las técnicas generales más importantes que se han propuesto en la literatura, en función de su solución y sus características.

El capítulo 2 introduce la técnica clásica del método de factorización, junto con el algoritmo de seguimiento de características que se usa como punto de partida de esta técnica. Se exponen los resultados de las pruebas realizadas con el sistema de seguimiento de características, en tiempo real, realizado al efecto.

El capítulo 3 está dedicado al modelado de la mano, imitando las posibilidades y restricciones de las posturas de una mano articulada que resulte anatómicamente realista y, al mismo tiempo, que también ofrezca un aspecto externo cercano a la realidad, modelando adecuadamente la interacción entre la piel y la luz.

En el capítulo 4 se introducen las ideas principales de los métodos basados en kernel. Se repasan contribuciones recientes como la posibilidad de invertir los resultados obtenidos en el nuevo espacio, volviendo a los espacios originales con el fin de poder interpretar visualmente sus prestaciones. Se elaboran extensiones de estos métodos para tratar con elementos rígidos diferentes y poder caracterizarlos individualmente. Se desarrolla la idea de poder comparar directamente los subespacios asignados a categorías diferentes, en lugar de hacerlo mediante puntos de muestra pertenecientes a dichos espacios.

En el capítulo 5 se aplican los desarrollos del capítulo anterior al problema

del establecimiento de correspondencias entre imágenes, mediante la transformación de una selección de sus puntos al nuevo espacio inducido por la función de kernel y asignando allí las correspondencias entre unos y otros. Esto se realiza para objetos rígidos y objetos articulados. En este último caso, introduciendo las técnicas de relajación en la adjudicación de etiquetas a cada parte rígida del objeto articulado. Se exponen los resultados de la aplicación de esas técnicas en posturas de manos sintéticas, para diversas funciones de kernel y con modelos tanto rígidos como articulados.

En el capítulo 6 se exponen algunos detalles de implementación que se han considerado de especial interés. En concreto, se trata una forma eficiente de realizar búsquedas de vecinos más próximos mediante el algoritmo Cover-Tree y la forma de dirigir mediante *scripts* el programa `Blender` de modelado 3D, para sintetizar las manos en distintas posturas. Ambos desarrollos se han realizado de forma que conecten con el programa `MATLAB`, que ha sido el utilizado para la mayoría de las pruebas.

Finalmente se termina la memoria con las conclusiones, que incluyen las aportaciones específicas realizadas en este trabajo y las posibilidades de continuación previstas.

Capítulo

1

Reconocimiento Gestual

1.1. Motivación y definición del problema

En estos tiempos cabe preguntarse si, ante el surgimiento de nuevas tendencias en la informática, como la realidad virtual, la realidad aumentada y la computación móvil, entre muchas otras, siguen siendo suficientes los medios tradicionales de interacción con el ordenador, como el teclado y el ratón, o es necesario avanzar un paso más para mejorar la calidad de dicha interacción.

El desarrollo del ratón, por ejemplo, constituyó un elemento esencial para lograr la facilidad de uso que ofrecen las interfaces gráficas de usuario que se conocen hoy en día. Quizás un cambio igual de importante en la interfaces de usuario esté a la espera de tecnologías que puedan sustentarlo. Una de dichas técnicas o tecnologías, que podrían revolucionar la forma de interactuar con la máquina, es el reconocimiento gestual. Fundamentalmente consiste en, como su propio nombre indica, que los gestos hechos por el usuario puedan ser reconocidos por el sistema con el que desea interactuar. El gesto en sí es un movimiento corporal con significado, que puede ser realizado con los dedos, las manos, los brazos, la cabeza, con todo el tronco; o incluso con una variación de la expresión de la cara o mediante el movimiento de ojos. Por sí mismo constituye una parte importante de la comunicación natural cotidiana, en muchas ocasiones como refuerzo al lenguaje hablado y, cuando éste no es posible, como la primera alternativa de comunicación. Por esta razón, el reconocimiento de los gestos puede convertir-

se en una novedosa forma de interacción, tanto en el ordenador de escritorio tradicional, como en las otras formas de computación emergentes —«vestible», «pervasiva» o «ubicua», etc.—.

En general, los gestos pueden ser de tipo estático —donde el usuario asume cierta postura, o configuración de su cuerpo o de la cara, fija— de tipo dinámico —en los que la información está codificada en el movimiento, a través de la evolución temporal de las posturas— o bien una mezcla de ambos, como, por ejemplo, es el caso de la lengua de signos. Los tipos de gestos más complejos, como ocurre en el ejemplo anterior, conectan varios símbolos a lo largo de una secuencia. Por tanto, su reconocimiento requiere de la segmentación en el tiempo de las configuraciones de postura que representan a cada símbolo. Pueden producirse entonces fenómenos como la coarticulación —que también se da en el habla verbal— en el que signos diferentes al ser «emitidos» en una secuencia alteran o mezclan sus representaciones. Esto se debe fundamentalmente a que resulta más sencillo o más rápido ejecutar de ese modo la secuencia.

Es posible categorizar los gestos más comúnmente utilizados en las distintas culturas:

- Gestos con el brazo y con la mano. Implican el reconocimiento de la configuración de la mano o conjunto brazo-mano y su evolución. Estos gestos son el objeto principal de este trabajo.

- Gestos con la cabeza y con la cara. Por ejemplo, asentir o negar, dirección de la mirada, movimiento de cejas, parpadeo, expresiones faciales complejas, etc.

- Gestos con el cuerpo. Implican el movimiento del cuerpo completo, incluyendo la posición, la configuración de su postura y la evolución de ambas.

Las aplicaciones posibles son numerosas y sólo están limitadas por la imaginación. Éstas podrían influir, no sólo específicamente en la industria de la informática, sino también cambiando la forma en la que las personas se comunican, se

transportan o son atendidas por sistemas automáticos, pues dichos sistemas serían más conscientes de la situación física de sus usuarios. Algunos ejemplos de aplicación más o menos inmediata del reconocimiento gestual son los siguientes:

- Reconocimiento de la lengua de signos.
- Navegar o realizar manipulaciones en entornos de realidad virtual o realidad aumentada.
- Computación «afectiva». Las expresiones faciales podrían ser usadas por el ordenador para interpretar estados emocionales del usuario.
- Controladores virtuales. Los dispositivos físicos reales podrían ser sustituidos por sistemas simulados que funcionaran a través del gesto en las situaciones que lo así precisaran.
- Nuevas tecnologías de ocio y juegos de tipo inmersivo. El estado de los espectadores —posición o postura— podría ser «captado» e integrado en la acción.
- Monitorización del estado de pacientes o de personas realizando actividades que requieren atención y permanecer en alerta como, por ejemplo, la conducción.
- Comunicaciones: videoconferencia, telecontrol, enseñanza a distancia, etc.

La forma más simple de capturar las características de un gesto cualquiera es mediante el uso de elementos sensores. Estos dispositivos deben de captar la posición del cuerpo, su configuración —ángulos de rotación de las articulaciones— y las velocidades correspondientes; e incluso pueden medir aceleraciones y fuerzas. Es posible caracterizar estos elementos sensores en base a diferentes criterios: precisión, resolución, latencia, rango de movimientos válidos, ergonomía y coste. En el caso más común, los sensores se colocan en guantes o trajes especiales para permitir el seguimiento del movimiento. Estos equipos suelen ser muy caros y molestos de usar; y además tienen un problema relacionado con

la transmisión de la información capturada al ordenador, pues si se realiza por medio de cables puede restringir mucho el movimiento y añadir peso adicional. Estos inconvenientes se superan mediante el uso de cámaras y tecnología de visión por computador, que ofrece un coste mucho más reducido y la facilidad de no obstruir en absoluto el movimiento ni alterar el comportamiento del usuario; además de ser la única posibilidad para, por ejemplo, captar expresiones faciales. A cambio, se introduce un problema que en toda su generalidad es enormemente complejo y que es el objeto central de este trabajo.

1.2. Descripción del problema y su complejidad

El problema que se pretende estudiar consiste en determinar las posiciones y orientaciones del cuerpo mediante un sistema de visión por computador. En el caso particular de la mano, un posible punto de partida es intentar caracterizar la variación de su forma considerando que se trata de un sistema compuesto por varios segmentos rígidos unidos por articulaciones. De esta manera se crea un modelo ajustable con varios grados de libertad, que condensa la información necesaria sobre la postura. Obtener la información completa estática de la configuración de la mano consistirá, por tanto, en determinar primero un modelo para la misma —en términos de la forma y tamaño de cada segmento junto con la especificación de cómo puede rotar cada articulación concreta– para a continuación intentar determinar los valores de todos los parámetros del modelo, así como calcular la posición y orientación de la mano, respecto a un sistema referencia en el mundo real, a partir de las imágenes obtenidas. Las dificultades para realizar el paso de imagen de la mano a parámetros de la postura son, en general, enormes. En primer lugar, se trata de un problema mal condicionado, debido que el paso del mundo 3D a la imagen 2D conlleva, obviamente, pérdida de información. Una postura observada en una imagen se puede corresponder con varias configuraciones posibles de parámetros de la mano, por motivos similares al hecho de que la proyección de un punto en la imagen se corresponde, en

general, con infinitos puntos situados en una recta que converja en el punto proyectado. Otro escollo común, también ocasionado por la naturaleza de la visión, es que falte información sobre la postura en la imagen, dado que puede haber oclusión de una parte del objeto por otra parte más cercana a la cámara. Otros problemas se relacionan más con las características del proceso de adquisición de imágenes. Por ejemplo, su resolución puede ser insuficiente para obtener el detalle necesario o, si se pretende obtener información del movimiento a través de una secuencia de imágenes, puede ocurrir que la resolución temporal tampoco sea la adecuada para la velocidad de los gestos a reconocer, obteniendo pocas imágenes y posiblemente borrosas por el movimiento, que no permitan reconstruir trayectorias fiables. Además de todo lo anterior, hay que señalar que el espacio de los parámetros o configuraciones que se pretenden determinar, suele poseer una dimensión muy elevada, si no se hacen grandes simplificaciones o se imponen restricciones al movimiento. En el caso del cuerpo humano, fácilmente se obtienen 50 grados de libertad asociados a las posiciones y orientaciones de todos sus miembros. Para el caso de la mano, son alrededor de 20 grados de libertad los necesarios para determinar completamente una postura. Todas ellas se deben inferir a partir de imágenes, que conceptualmente no son más que puntos de un espacio de dimensión mucho mayor —en una imagen de niveles de gris, cada píxel representa una dimensión diferente—. En el caso de un gesto dinámico, el problema es todavía más complejo, pues se trataría de encontrar una trayectoria en el espacio de las configuraciones, representada ésta por un vector de parámetros a lo largo del tiempo. Por supuesto, este problema comparte otras dificultades con el procesado «de bajo nivel» de la imagen: elección de algoritmos de preprocesamiento que mejorarán sus propiedades, separación de las partes de la imagen que son de interés del «fondo» —que simplemente contribuye con ruido al problema final— y la selección de las características o elementos informativos medibles en la imagen.

Como en todo problema de visión por computador, para superar las dificultades anteriores, se hace necesario modificar las condiciones experimentales de captura para aportar la información que falta. Así, las distintas técnicas se dife-

renciarán en función de una serie de consideraciones o suposiciones en las que se basa cada tipo de algoritmo:

- Número de cámaras usadas. El uso de varias cámaras especialmente sincronizadas permiten la reconstrucción 3D directa de un punto de un objeto, siempre que se pueda identificar a dicho punto en todas las imágenes. A esto último se lo conoce como el problema del encaje o emparejamiento para estereovisión.

- Necesidad o no de realizar el calibrado de las cámaras. El proceso mediante el cual se determina la transformación no invertible que convierte los puntos en 3D del mundo real a posiciones en píxeles de la imagen se llama calibración. Una vez realizada, para las características de la cámara y una orientación fija de la misma respecto a una referencia del mundo real, es posible predecir la posición en la imagen 2D a partir de sus coordenadas 3D.

- Estructura del entorno. Restricciones de movimiento impuestas sobre el gesto, condiciones de iluminación (luz estructurada), restricciones sobre la posición y el estado de movimiento de la cámara respecto los sujetos.

- Requisitos del usuario. En general se puede facilitar el problema haciendo que el usuario posea una imagen distintiva que facilite la cadena de procesos sobre la imagen, por ejemplo, haciendo que lleve marcas especiales sobre el cuerpo o bien un guante marcado con colores.

- Uso de secuencias de imágenes con movimiento. La técnica puede utilizar una secuencia de imágenes para obtener información y desambiguar las características de las mismas

1.3. Técnicas de reconocimiento gestual

Las técnicas de reconocimiento gestual han experimentado un gran auge en la última década —véase (Mitra y Acharya, 2007) para una revisión reciente— motivado fundamentalmente por la importancia de sus posibles aplicaciones: nuevas modalidades de interacción hombre-máquina, reconocimiento del lenguaje de signos, navegación en entornos virtuales, identificación de expresiones faciales, biometría y seguridad, seguimiento o *tracking* del movimiento de personas o grupos; por citar alguna de las más relevantes. Por otro lado, avances recientes en el campo de la visión por computador y el aprendizaje automático han encontrado aplicación directa en este tipo de problemas, surgiendo una gran cantidad de técnicas y aproximaciones diferentes en su resolución. A pesar de ello, y debido a la gran dificultad del problema, éste dista mucho de estar resuelto. Una de las dificultades que obstaculizan el progreso de estas técnicas es que, frecuentemente, sus prestaciones no son directamente comparables, siendo muy difícil el apreciar la ventaja de un algoritmo o técnica particular sobre otro y las condiciones en que se manifiesta la diferencia. Por ello, como se reconoce ampliamente en la literatura, es necesario sistematizar la caracterización de las prestaciones, haciendo explícitas las condiciones experimentales o desarrollando las pruebas en conjuntos de problemas estandarizados y al alcance de toda la comunidad, de manera que se garantice su reproducibilidad. Actualmente, existen diversas bases de datos —tanto de imágenes como de vídeos (Sigal *et al.*, 2009)— y problemas de *benchmark* públicamente disponibles; aunque no son numerosos debido a la complejidad que entraña la captura en condiciones controladas y la medida real de los parámetros que deben ser estimados por los algoritmos —por ejemplo, medida de ángulos y posiciones de las articulaciones, —. Estos parámetros frecuentemente sólo pueden ser medidos indirectamente, recurriendo a guantes con sensores en el caso de las manos, o a marcadores y sistemas de captura en 3D de coste elevado capaces de posicionar en el espacio dichos marcadores y trazar su movimiento. Esto hace que sólo laboratorios de visión adecuadamente dotados puedan obtener esas mediciones.

En la literatura es frecuente encontrar modelos probabilísticos bayesianos que explotan la continuidad del movimiento junto con información del pasado para, utilizando información extraída de la imagen en ese instante —características— actualizar la estimación del estado actual en el espacio de las configuraciones —parámetros—. Aquí surgen diversos modelos y métodos en función de que se considere: un conjunto finito discreto de estados o un espacio de estados continuo, las suposiciones sobre la distribución de probabilidad atribuida al estado, el uso modelos de dinámica lineal o no lineal, etc. Respecto a esto último cabría destacar modelos como: el modelo de estados ocultos de Markov (HMM) (Wilson y Bobick, 1999), el filtro de Kalman y el filtro de Kalman generalizado, el filtro de partículas (Arulampalam *et al.*, 2002; Bretzner *et al.*, 2002), el algoritmo de condensación (Isard y Blake, 1998), etc; cada uno de los cuales con sus correspondientes métodos de entrenamiento, adaptación y actualización. En muchos de los métodos probabilísticos, es necesario disponer de la función de verosimilitud, que vincula la probabilidad de las características observadas con los parámetros desconocidos. Para obtenerla se pueden aplicar diversas técnicas de aprendizaje automático y de *soft-computing* para encontrar la selección de características más efectiva e inferir sus relaciones con los parámetros. En otras contribuciones se usan modelos menos fundamentados teóricamente, como los modelos de redes neuronales dinámicas. Éstas son regresores muy flexibles capaces de aproximar evoluciones temporales indefinidamente complejas como, por ejemplo, la denominada redes neuronales con retardos en el tiempo (TDNN). Como contrapartida, es importante destacar la dificultad de su proceso de entrenamiento.

En general, se pueden encontrar dos grandes familias de métodos en torno al reconocimiento gestual: los basados en modelos y los basados en la apariencia. Los segundos intentan buscar directamente en las imágenes estructuras que cumplan ciertas relaciones, con el objetivo de inferir las posiciones y orientaciones. Los primeros utilizan un modelo de lo que se pretende distinguir en la imagen. Ese modelo puede ser de mayor o menor complejidad, pero depende de una serie de parámetros que pueden ser ajustados, de manera que comparando

la imagen obtenida del modelo con la real es posible, en principio, dirigir su sintonización hasta que ambos coincidan. En cualquiera de los casos, los problemas son muy difíciles de tratar. En particular, las técnicas basadas en modelos poseen requerimientos computacionales elevados durante el proceso de optimización de los parámetros, proceso que además es una búsqueda plagada de mínimos locales.

Las técnicas de aprendizaje automático y *soft-computing* pueden aliviar en ambos casos el diseño del reconocedor. En las técnicas basadas en la apariencia se puede automatizar el diseño haciendo que las características a buscar y sus relaciones sean directamente aprendidas a través de modelos jerárquicos simples (Ramanan *et al.*, 2007). Posteriormente es posible, en principio, inducir la relación entre las características así obtenidas y los parámetros, aunque este problema también posee gran dificultad.

Entre las técnicas basadas en la apariencia cabe destacar las populares contribuciones de Viola *et al.* (2005), como ejemplo de cómo los principios del aprendizaje automático pueden apoyar la selección de características muy simples que junto con el entrenamiento de clasificadores o regresores permiten lograr un gran rendimiento en el reconocimiento facial —algoritmo *AdaBoost* y derivados—. Algunas extensiones que tratan el reconocimiento gestual, son los trabajos de Kolsch y Turk (2004) y Viola *et al.* (2005), incorporándose el movimiento al método en este último. Un ejemplo reciente de esta familia de métodos es la propuesta de Mita *et al.* (2008).

Las técnicas basadas en modelos presentan una gran diversidad en función de la complejidad del mismo. Se pueden encontrar en la literatura desde modelos articulados construidos con formas cilíndricas hasta complejos modelos 3D con apariencia fotorrealista. El modelo puede ser también deformable —sometido a una variabilidad mayor que la que ofrecen los simples cambios en la rotación y posición de los modelos rígidos— en cuyo caso es necesario introducir parámetros que den cuenta de esa deformación, que no necesariamente tiene que ser fácil de modelizar. Una de las ventajas de estos métodos es que pertenecen a la clase de métodos denominados generativos, es decir, que una determinada hi-

pótesis sobre los parámetros de configuración de una postura puede ser usada para generar una imagen —realista o aproximada, en función del modelo— de la misma y ser verificada visualmente. En contraste, los métodos discriminadores sólo apuntan a determinar los parámetros a partir de los datos, no permitiendo dicha síntesis.

Siguiendo la línea de los métodos basados en modelos, es útil considerar la geometría del espacio de las imágenes generadas por la variación del vector de parámetros del modelo. A pesar de que, como se comentó antes, las imágenes representan un espacio de muy alta dimensión, lo cierto es que realmente residen en una hipersuperficie o variedad de dimensión intrínseca determinada por la dimensión del vector de parámetros. Esta dimensión intrínseca es mucho menor que la del espacio ambiente en la que se encuentran inmersas. Esto conduce a una línea de trabajo en la que lo que se pretende es estimar transformaciones de reducción de la dimensión sobre el espacio de las imágenes, de manera que se obtengan nuevos parámetros independientes. Es posible adoptar una postura agnóstica sobre el significado de los parámetros, porque en algunos casos los nuevos parámetros logran capturar toda la variabilidad de los gestos, sin tener relaciones evidentes con los originales. Hay que tener en cuenta también que en el movimiento humano existen grados de libertad que no son independientes, por lo que la reducción a los nuevos parámetros puede llegar a ser inferior que los grados de libertad de los modelos de partida, si es que los gestos o posturas fueron capturados a partir de movimiento real y no sintetizados. Algunas de estas técnicas son clásicas, como el análisis de componentes principales (PCA) (Jolliffe, 2002) que posee aplicaciones populares en visión como, por ejemplo, las famosas *EigenFaces* (Turk y Pentland, 1991). El PCA posee una importante generalización a componentes derivadas no linealmente de las variables, el análisis de componentes principales basado en kernel (KPCA), que se discutirá extensamente en este trabajo. El escalado multidimensional (MDS); y otros desarrollos más recientes como *Isomap* (Tenenbaum *et al.*, 2000), *Locally Linear Embedding* (LLE) (Roweis y Saul, 2000) o *Laplacian Eigenmaps* (Belkin y Niyogi, 2001); son técnicas con idéntico propósito que recurren a criterios diferentes en

la búsqueda de la transformación de reducción de dimensión. Por ello no todos estos métodos permiten ir de los nuevos parámetros a la reconstrucción de la imagen. Por su similaridad, (Yan *et al.*, 2007) hicieron una propuesta que trataba de unificar en un mismo marco teórico todos estas técnicas de reducción de dimensión. Estrechamente relacionados con ellas, existe otra familia de métodos de marco probabilístico en la que se interpretan los nuevos parámetros como variables ocultas o latentes que varían dando lugar a las observaciones, que en este caso son las imágenes capturadas (Jaeggli *et al.*, 2008).

Capítulo

2

Métodos de factorización y alineamiento de imágenes

Los métodos de factorización componen una familia de algoritmos capaces de recuperar forma y movimiento de objetos a partir de una secuencia de imágenes obtenidas de una sola cámara. Estos métodos, desarrollados a lo largo de las dos últimas décadas por varios autores, se diferencian en las suposiciones sobre el tipo de objeto de interés, el tipo de proyección —modelo matemático asumido para el paso del mundo 3D al 2D— y el movimiento de la cámara y del propio objeto. Éste puede ser un sólido rígido, un sólido articulado —con varias partes rígidas que se mueven en base a ciertas restricciones— o deformable.

El punto de partida de todos estos algoritmos consiste en obtener un número de características o puntos de interés sobre el objeto a lo largo de la secuencia de imágenes. El objetivo es identificar dichos puntos de interés, obteniendo sus coordenadas 2D, en cada una de las imágenes de una secuencia. Como se verá, esto es un problema importante que tiene mucho en común con otros problemas típicos de la visión por computador[1].

Puesto que el producto de los métodos de factorización serán descripciones de formas y movimientos de los objetos, esto los hace sumamente interesantes en el reconocimiento gestual. En este capítulo se comentará, como ejemplo ilustrativo,

[1] A estos problemas se los conoce como de: seguimiento de características, registro de imágenes, alineamiento de imágenes, correspondencia, etc. en función del contexto.

el método de factorización de Tomasi y Kanade (1992), que permite recuperar la forma y el movimiento relativo entre la cámara y el objeto. En segundo lugar se expondrá uno de los algoritmos de seguimiento de características más populares en su uso con los métodos de factorización, el denominado seguidor de características Kanade-Lucas-Tomasi (KLT) (Lucas y Kanade, 1981; Shi y Tomasi, 1994). Posteriormente se comentarán los resultados de pruebas experimentales realizadas con imágenes reales de reconocimiento gestual de manos, con cámaras de vídeo comunes y sin iluminación estructurada. Finalmente se analizarán las causas del bajo rendimiento de los seguidores de características que originan que esta aproximación al problema, en una situación común, no sea efectiva.

2.1. Método de Tomasi y Kanade

El método de factorización de Tomasi y Kanade (1992) permite recuperar la estructura 3D y el movimiento de la cámara en una secuencia de imágenes bajo la suposición de que el modelo de proyección es ortográfico. Esta proyección no siempre es una buena aproximación a la proyección en perspectiva, pero varios autores han extendido el desarrollo a modelos de proyección más reales. En lo que sigue se supondrá una cámara afín, que sin ser una proyección en perspectiva, incluye como casos particulares la ortográfica y la perspectiva «débil», entre otros modelos.

Tal y como exponen Hartley y Zisserman (2004), el efecto de una cámara afín puede describirse como la siguiente transformación entre las coordenadas del mundo 3D —en mayúsculas por convenio— y las coordenadas 2D en la imagen —en minúsculas—.

$$\begin{bmatrix} x \\ y \end{bmatrix} = \mathbf{M} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \boldsymbol{t} \tag{2.1}$$

donde $\mathbf{M}$ es una matriz 2×3 y $\boldsymbol{t}$ es un vector de dimensión 2. Ambas coordenadas son convencionales cartesianas —no homogéneas— y pueden ser repre-

sentadas en notación vectorial como: $\boldsymbol{x} = (x, y)^T$ y $\boldsymbol{X} = (X, Y, Z)^T$. Suponiendo que se dispone de un conjunto de n puntos de interés, o características, que se pueden identificar en una serie de m vistas, o imágenes, en las que la posición de la cámara ha ido cambiando —es decir, se dispone de las posiciones $\boldsymbol{x}$ medidas sobre la serie de vistas para cada una de las características—. El objetivo es minimizar el error geométrico entre las posiciones medidas en la imagen y las predichas por la expresión anterior teniendo en cuenta que las posiciones $\boldsymbol{X}$ —3D— de cada característica son desconocidas, aunque constantes a lo largo de la serie de vistas, y que cada vista está caracterizada por parámetros $\mathbf{M}$ y $\boldsymbol{t}$ desconocidos y diferentes, debido al movimiento de la cámara. Esto es, se pretende estimar las m «cámaras» $\{\mathbf{M}^i, \boldsymbol{t}^i\}$ que caracterizan el movimiento en la secuencia y las n posiciones de las características en 3D $\{\boldsymbol{X}_j\}$ que dan la geometría del objeto, de tal manera que se minimice la diferencia entre las posiciones de las características estimadas en la imagen $\hat{\boldsymbol{x}}_j^i = \mathbf{M}^i \boldsymbol{X}_j + \boldsymbol{t}^i$ y las posiciones medidas $\boldsymbol{x}_j^i$

$$\underset{\mathbf{M}^i, \boldsymbol{t}^i, \boldsymbol{X}_j}{\arg\min} \sum_{i,j} \|\hat{\mathbf{X}}_j^i - \boldsymbol{x}_j^i\|^2 \tag{2.2}$$

Es posible eliminar de antemano los vectores de traslación $\boldsymbol{t}^i$ de la minimización, eligiendo como origen de cada imagen el centroide del conjunto de posiciones en dicha imagen. Eso se puede ver analíticamente minimizando respecto a $\boldsymbol{t}^i$

$$\frac{\partial}{\partial \boldsymbol{t}^i} \sum_{k,j} \|\boldsymbol{x}_j^k - (\mathbf{M}^k \boldsymbol{X}_j + \boldsymbol{t}^k)\|^2 = \mathbf{0} \tag{2.3}$$

y realizando algunas simplificaciones, de manera que se obtiene como solución $\boldsymbol{t}^i = \langle \boldsymbol{x}^i \rangle - \mathbf{M}^i \langle \boldsymbol{X} \rangle$, donde $\langle \boldsymbol{x}^i \rangle = \frac{1}{n} \sum_j \boldsymbol{x}_j^i$ y $\langle \boldsymbol{X} \rangle = \frac{1}{n} \sum_j \boldsymbol{X}_j$ son los centroides de las posiciones de las características en la i-ésima imagen y de las características reales en 3D, respectivamente. Dado que el origen de los ejes de coordenadas del mundo 3D es arbitrario se puede ubicar en el centroide $\langle \boldsymbol{X} \rangle$, de manera que $\langle \boldsymbol{X} \rangle = \mathbf{0}$ por lo que $\boldsymbol{t}^i = \langle \boldsymbol{x}^i \rangle$. Entonces, si en cada imagen se miden las posiciones tomando como origen el centroide $\langle \boldsymbol{x}^i \rangle$ resulta evidente que $\boldsymbol{t}^i = \mathbf{0}$. Por

simplicidad se asume que se ha realizado este proceso, de manera que se reemplaza x_j^i por $x_j^i - \langle x^i \rangle$, lo que permite que la expresión a minimizar en las nuevas posiciones centradas sea

$$\min_{\mathbf{M}^i, t^i, \boldsymbol{X}_j} \sum_{i,j} \|x_j^i - \mathbf{M}^i \boldsymbol{X}_j\|^2 \tag{2.4}$$

Esta expresión tiene una forma sencilla en términos de matrices. Se denomina matriz de medidas $\mathbf{W}$ a la matriz $2m \times n$ que contiene las x_j^i centradas

$$\mathbf{W} = \begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_n^1 \\ x_1^2 & x_2^2 & \cdots & x_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^m & x_2^m & \cdots & x_n^m \end{bmatrix} \tag{2.5}$$

mientras que las posiciones estimadas se pueden escribir como la matriz de parámetros $\hat{\mathbf{W}}$

$$\hat{\mathbf{W}} = \begin{bmatrix} \mathbf{M}^1 \\ \mathbf{M}^2 \\ \vdots \\ \mathbf{M}^m \end{bmatrix} \begin{bmatrix} \boldsymbol{X}^1 & \boldsymbol{X}^2 & \cdots & \boldsymbol{X}^n \end{bmatrix} = \hat{\mathbf{M}}\hat{\boldsymbol{X}} \tag{2.6}$$

Se puede demostrar que encontrar una matriz $\hat{\mathbf{W}}$ que sea lo más parecida posible a $\mathbf{W}$ en términos de la norma de Frobenius es equivalente a encontrar la solución a la minimización de la ecuación (2.4). La matriz $\hat{\mathbf{W}}$ es el producto de la *matriz de movimiento* $\hat{\mathbf{M}}$ de dimensiones $2m \times 3$, que da cuenta del movimiento de la cámara, y la *matriz de estructura* $\hat{\boldsymbol{X}}$ de dimensiones $3 \times n$, que determina las posiciones de las características en el espacio 3D.

Dado que el rango de un producto de matrices es menor o igual que el menor de los rangos de los factores, se deduce que el rango de $\hat{\mathbf{W}}$ es 3. En definitiva, se busca hallar una matriz $\hat{\mathbf{W}}$ que minimice la norma de Frobenius de la diferencia

con la de las medidas $\mathbf{W}$

$$\arg\min_{\hat{\mathbf{W}}} \|\mathbf{W} - \hat{\mathbf{W}}\|_F^2 \tag{2.7}$$

y que además tenga rango 3. Tal matriz puede ser hallada utilizando la descomposición en valores singulares (SVD) de $\mathbf{W}$, truncada posteriormente a rango 3. Es decir, si la descomposición en valores singulares de $\mathbf{W}$ fuera

$$\mathbf{W} = \mathbf{U}\mathbf{D}\mathbf{V}^T \tag{2.8}$$

entonces

$$\hat{\mathbf{W}} = \mathbf{U}_{2m\times 3}\mathbf{D}_{3\times 3}\mathbf{V}_{3\times n}^T \tag{2.9}$$

donde $\mathbf{U}_{2m\times 3}$ son las tres primeras columnas de $\mathbf{U}$, $\mathbf{V}_{3\times n}^T$ son las tres primeras filas de $\mathbf{V}^T$ y $\mathbf{D}_{3\times 3}$ es la submatriz de la matriz diagonal $\mathbf{D}$ con los tres primeros valores singulares. Sin embargo, no hay una forma única de asignar estos factores a las matrices $\hat{\mathbf{M}}$ y $\hat{X}$

$$\hat{\mathbf{W}} = \mathbf{U}_{2m\times 3}\mathbf{D}_{3\times 3}\mathbf{V}_{3\times n}^T = \hat{\mathbf{M}}\hat{\mathbf{X}} \tag{2.10}$$

ya que podemos podemos elegir

$$\hat{\mathbf{M}} = \mathbf{U}_{2m\times 3}\mathbf{D}_{3\times 3} \tag{2.11}$$

$$\hat{X} = \mathbf{V}_{3\times n}^T \tag{2.12}$$

o bien hacer

$$\hat{\mathbf{M}} = \mathbf{U}_{2m\times 3} \tag{2.13}$$

$$\hat{X} = \mathbf{D}_{3\times 3}\mathbf{V}_{3\times n}^T \tag{2.14}$$

También hay que destacar que existe otro tipo de ambigüedad, ya que se puede insertar en la descomposición una matriz $\mathbf{A}$ de dimensión 3×3 de rango com-

pleto —para que sea invertible— y producir la misma $\hat{\mathbf{W}}$

$$\hat{\mathbf{W}} = \hat{\mathbf{M}}\hat{\mathbf{X}} = \hat{\mathbf{M}}\mathbf{A}\mathbf{A}^{-1}\hat{\mathbf{X}} = (\hat{\mathbf{M}}\mathbf{A})(\mathbf{A}^{-1}\hat{\mathbf{X}}) \tag{2.15}$$

Este tipo de reconstrucción, determinada hasta una transformación lineal $\mathbf{A}$, se denomina reconstrucción afín. Para obtener una reconstrucción completa o *métrica* se puede acudir a diversas consideraciones sobre la forma de las matrices e inyectar información de la escena o bien efectuar un calibrado. En cualquier caso es posible recuperar la información completa sobre la escena 3D y el movimiento de la cámara —o equivalentemente, del objeto si se supone que la cámara está fija y es el objeto el que se mueve—.

2.2. Seguidor de características Kanade-Lucas-Tomasi

Como se ha comentado, los algoritmos de factorización necesitan puntos de interés que se repitan de imagen a imagen a lo largo de una secuencia. El algoritmo seguidor de características Kanade-Lucas-Tomasi (KLT) da una respuesta eficiente al problema y es la solución más popular al mismo.

Tomasi y Kanade (1992) plantean dos problemas fundamentales:

1. Cómo seleccionar las características.

2. Cómo realizar su seguimiento de una imagen a la siguiente.

Su método se basa en solucionar el segundo problema. Se lleva a cabo el seguimiento mediante un procedimiento propuesto anteriormente por Lucas y Kanade (1981) para el encaje de dos imágenes en estereovisión. En ese método se determina la diferencia entre los niveles de intensidad de una subimagen en el instante anterior y el actual. Si hay poco movimiento de cámara entre la imagen anterior y la actual, la subimagen actual puede ser aproximada por una traslación de la anterior. Las intensidades actuales pueden ser escritas como las anteriores mas un término residual, que depende casi linealmente del vector de

traslación. Con esta aproximación, se puede escribir un sistema lineal de dos ecuaciones y dos incógnitas que son las traslaciones en cada eje. Debido a las aproximaciones utilizadas, tal sistema se soluciona de forma iterativa, aunque converge rápidamente a la solución.

El primer problema, el de la selección de las características —y por tanto de la subimagen o ventana donde reside esa característica de la imagen— ha sido tratado por varios autores. Existen múltiples definiciones de lo que constituye una característica «buena»: ventanas con una desviación estándar más alta en intensidades, cruces por cero de la laplaciana de la intensidad, características que son de tipo «esquina» debido a las propiedades de las derivadas primera y segunda de la intensidad, etc. La solución propuesta en el algoritmo KLT es simple: una característica es buena si puede ser seguida de una imagen a otra correctamente, y por ello proceden a definir un criterio de elección que optimiza el seguimiento en su algoritmo. Es decir, no se trata independientemente el seguimiento y la selección de características.

Aunque Tomasi y Kanade (1992) abogan porque las traslaciones sean las únicas transformaciones válidas, claramente esto sólo es una aproximación. La suposición de poder encontrar una traslación entre las dos imágenes se viola en varias situaciones. Por ejemplo, los puntos que han entrado o que han salido de la imagen no tienen equivalencia, al igual que los puntos en fronteras de oclusión de un objeto por otro o cuando el punto de vista cambia, lo que puede influir en la intensidad recogida por la cámara si la reflectividad de las superficies depende de él. En principio, los puntos en superficies con alguna textura o marca que estén lejos de fronteras de oclusión son buenos candidatos al seguimiento cuando únicamente se utilizan traslaciones.

El seguimiento no se realiza en cada píxel de la imagen, debido al probable parecido con sus vecinos además de a la posible aleatoriedad de su valor, sino que se hace en ventanas de píxeles. La elección de la ventana —o característica— debe seguir el criterio general anterior. El problema surge cuando dentro de una ventana se observan comportamientos diferentes. Por ejemplo, si la superficie enfocada presenta mucho sesgo, la imagen actual puede ser muy diferente de

la anterior a causa de la perspectiva. Otro problema surge si la ventana incluye puntos de objetos a diferentes profundidades, pues se observaría como si los puntos de la ventana se movieran a velocidades distintas, obteniéndose varios vectores de traslación para puntos diferentes de la misma ventana. Para estas dificultades Tomasi y Kanade (1992) proponen dos soluciones. La primera consiste en monitorizar las diferencias entre las dos ventanas como un error residual y si se encuentra que es demasiado alto, se descarta la ventana. La otra solución pasa por proponer un modelo de transformación más complejo que la simple traslación, como por ejemplo una transformación afín, mucho más adecuada para modelizar los cambios provocados por la perspectiva. En todo caso se debe pesar la mejora con la necesidad de estimar muchos más parámetros y los inconvenientes que eso comporta. Es necesario usar ventanas más grandes para determinar los parámetros fiablemente y, a su vez, las ventanas mayores son mucho más susceptibles de causar los problemas apuntados. Por ello concluyen en estimar únicamente traslaciones y además hacerlo sobre ventanas pequeñas.

A continuación se verá el procedimiento de seguimiento en un caso genérico y a partir de él se derivará el criterio para elegir las características. Las imágenes se modelizan como funciones $I(\boldsymbol{x})$ que tienen como argumento la posición de un píxel $\boldsymbol{x} = (x, y)^T$ —una pareja de coordenadas correspondientes a su posición discreta en la imagen— y devuelven, en el caso más simple, el nivel de gris correspondiente a dicho píxel.

El objetivo es alinear una imagen plantilla $T(\boldsymbol{x})$ con una imagen de entrada $I(\boldsymbol{x})$ mediante cierta transformación sobre las posiciones de los píxeles —no sobre los niveles de intensidad—. Si se usase el algoritmo para hacer seguimiento de características —y no, por ejemplo, encaje de escenas en estereovisión— $T(\boldsymbol{x})$ podría ser una subimagen de la imagen anterior en la secuencia e $I(\boldsymbol{x})$ la imagen actual. Sea $\mathbf{W}(\boldsymbol{x}; \boldsymbol{p})$ el conjunto de transformaciones permitido, caracterizado por un vector de parámetros $\boldsymbol{p} = (p_1, \ldots, p_n)^T$. La función $\mathbf{W}(\boldsymbol{x}; \boldsymbol{p})$ toma el píxel $\boldsymbol{x}$ de la imagen T y lo mapea en su posición correspondiente en la imagen I —con precisión subpíxel, ya que devuelve un par de coordenadas reales—. Como ejemplo,

$\mathbf{W}(\boldsymbol{x};\boldsymbol{p})$ puede ser una simple traslación

$$\mathbf{W}(\boldsymbol{x};\boldsymbol{p}) = \begin{bmatrix} x + p_1 \\ y + p_2 \end{bmatrix} \tag{2.16}$$

o bien una función más compleja, como por ejemplo una transformación afín

$$\mathbf{W}(\boldsymbol{x};\boldsymbol{p}) = \begin{bmatrix} (1+p_1)\cdot x + p_3 \cdot y + p_5 \\ p_2 \cdot x + (1+p_4)\cdot y + p_6 \end{bmatrix} = \begin{bmatrix} 1+p_1 & p_3 & p_5 \\ p_2 & 1+p_4 & p_6 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.17}$$

El objetivo del algoritmo consiste en averiguar los parámetros $\boldsymbol{p}$ de la transformación, de forma que minimicen la suma del error cuadrático entre los niveles de gris de la plantilla T y los niveles de gris de la imagen I sometida a la transformación de las posiciones de sus píxeles. La suma se lleva a cabo para todos los píxeles de la imagen plantilla T

$$\sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p})) - T(\boldsymbol{x}) \right]^2 \tag{2.18}$$

El cálculo de $I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}))$ implica encontrar el valor de la imagen I en una posición que no tiene porqué coincidir exactamente con la de un píxel y, por tanto, será necesario interpolar para determinar el valor de la imagen en ese punto. La minimización respecto a los parámetros $\boldsymbol{p}$ es, en general, una optimización no lineal; incluso aunque la función $\mathbf{W}(\boldsymbol{x};\boldsymbol{p})$ dependa linealmente de ellos. Esto es debido a la relación compleja que se establece a través de I entre la posición de un píxel y su nivel de intensidad. Para hacer factible la minimización, en el algoritmo se asume que se dispone de una estimación actual de $\boldsymbol{p}$ y se resuelve iterativamente buscando variaciones $\Delta\boldsymbol{p}$ que reduzcan el error. Es decir, se minimiza

$$\sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x};\boldsymbol{p}+\Delta\boldsymbol{p})) - T(\boldsymbol{x}) \right]^2 \tag{2.19}$$

con respecto a $\Delta \boldsymbol{p}$ y posteriormente se actualiza el valor de la estimación de $\boldsymbol{p}$

$$\boldsymbol{p} \leftarrow \boldsymbol{p} + \Delta \boldsymbol{p} \tag{2.20}$$

Estos pasos se repiten iterativamente hasta que la estimación converge. Típicamente se prosigue hasta que las variaciones $\Delta \boldsymbol{p}$ sean pequeñas, por debajo de un umbral prefijado.

A continuación se exponen los detalles del algoritmo para minimizar (2.19), que consiste básicamente en realizar una optimización Gauss-Newton de gradiente descendente. En la expresión (2.19) se desarrolla en serie de Taylor el término $I(\mathbf{W}(\boldsymbol{x}; \boldsymbol{p} + \Delta \boldsymbol{p}))$ respecto a $\Delta \boldsymbol{p}$, en torno a $\boldsymbol{p}$ y hasta el término lineal, con lo que se linealiza la expresión y se obtiene

$$\sum_{\boldsymbol{x}} \left[I(\mathbf{W}(\boldsymbol{x}; \boldsymbol{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \Delta \boldsymbol{p} - T(\boldsymbol{x}) \right]^2 \tag{2.21}$$

En esta expresión $\nabla I = (\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y})$ es el gradiente de la imagen I evaluado en $\mathbf{W}(\boldsymbol{x}; \boldsymbol{p})$ para lo cual se usa la estimación actual de $\boldsymbol{p}$. El término $\frac{\partial \mathbf{W}}{\partial \boldsymbol{p}}$ es el jacobiano de la transformación. Si $\mathbf{W}(\boldsymbol{x}; \boldsymbol{p}) = (W_x(\boldsymbol{x}; \boldsymbol{p}), W_y(\boldsymbol{x}; \boldsymbol{p}))^T$ entonces

$$\mathbf{W}(\boldsymbol{x}; \boldsymbol{p}) = \begin{bmatrix} \frac{\partial W_x}{\partial p_1} & \frac{\partial W_x}{\partial p_2} & \cdots & \frac{\partial W_x}{\partial p_n} \\ \frac{\partial W_y}{\partial p_1} & \frac{\partial W_y}{\partial p_2} & \cdots & \frac{\partial W_y}{\partial p_n} \end{bmatrix} \tag{2.22}$$

La expresión (2.21) puede ser minimizada obteniéndose una solución explícita. Para ello se calcula su derivada parcial respecto a $\Delta \boldsymbol{p}$, obteniéndose

$$2 \sum_{\boldsymbol{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \right]^T \left[I(\mathbf{W}(\boldsymbol{x}; \boldsymbol{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \Delta \boldsymbol{p} - T(\boldsymbol{x}) \right] \tag{2.23}$$

igualando a cero y resolviendo para $\Delta \boldsymbol{p}$ se obtiene la solución

$$\Delta \boldsymbol{p} = \mathbf{H}^{-1} \sum \boldsymbol{x} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \right]^T [T(\boldsymbol{x}) - I(\mathbf{W}(\boldsymbol{x}; \boldsymbol{p}))] \tag{2.24}$$

donde $\mathbf{H}$ es una matriz $n \times n$ que es la aproximación Gauss-Newton a la matriz Hessiana

$$H = \sum_{x} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \right]^T \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \right] \tag{2.25}$$

El algoritmo finalmente consiste en aplicar iterativamente las ecuaciones (2.24) para obtener la variación y actualizar la estimación con (2.20).

En el caso particular de una traslación, el jacobiano $\frac{\partial \mathbf{W}}{\partial \boldsymbol{p}}$ es la identidad, con lo que el sistema de ecuaciones lineal a resolver es

$$\left(\sum_{x} \nabla I^T \nabla I \right) \Delta \boldsymbol{p} = \left(\sum_{x} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \right]^T [T(\boldsymbol{x}) - I(\mathbf{W}(\boldsymbol{x}; \boldsymbol{p}))] \right) \tag{2.26}$$

donde se puede definir

$$\mathbf{G} = \sum_{x} \nabla I^T \nabla I \tag{2.27}$$

$$\boldsymbol{e} = \sum_{x} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \boldsymbol{p}} \right]^T [T(\boldsymbol{x}) - I(\mathbf{W}(\boldsymbol{x}; \boldsymbol{p}))] \tag{2.28}$$

Se puede observar que la matriz de los coeficientes $\mathbf{G}$ es de 2×2 y simétrica, mientras que $\boldsymbol{e}$ es un vector columna de dimensión 2. Para que el seguimiento sea correcto es necesario que el sistema (2.26) represente medidas fiables y que sea resoluble de forma robusta. Ello implica, por un lado, que la matriz $\mathbf{G}$ tenga valores considerablemente por encima del nivel de ruido en la imagen, y por otro, que esté bien condicionada. En términos de sus dos autovalores, estas condiciones señalan que ambos deben tener valores grandes y que además no deben diferir en varios órdenes de magnitud. Por ejemplo, dos autovalores pequeños significan un perfil de intensidades aproximadamente constante en la ventana, que no aporta información relevante para el seguimiento; un autovalor grande y otro pequeño implica un patrón unidireccional, que sólo da información fiable sobre el movimiento en dirección perpendicular a dicho patrón; dos autovalores grandes pueden representar esquinas, texturas o en general, un patrón informa-

tivo sobre el movimiento en el plano. En la práctica, si el menor autovalor es lo suficientemente grande para superar el criterio del ruido, la matriz está bien condicionada, ya que la variación del otro autovalor está acotada por el valor máximo de intensidad de un píxel. Por ello, si λ_1 y λ_2 son la pareja de autovalores de G, el criterio para aceptar el seguimiento de una ventana, y por tanto una característica como buena, es

$$\min(\lambda_1, \lambda_2) > \lambda_{umbral} \tag{2.29}$$

siendo λ_{umbral} una constante a determinar. Tomasi y Kanade (1992) proponen elegir el valor medio entre el autovalor calculado en una zona sin características —representativa del ruido en la imagen— y el valor de los autovalores en la saturación, indicando que experimentalmente esta elección funciona adecuadamente.

2.3. Resultados experimentales

El objetivo de estos experimentos es comprobar hasta que punto las técnicas anteriores son usables para el reconocimiento gestual de la mano. Para ello, se asume que el reconocimiento se efectúa en un entorno convencional de trabajo. Es decir, la mano no se va a situar en ninguna postura compleja que resulte impracticable o resulte cansada de mantener, sino que se apoya el antebrazo sobre la mesa de trabajo. La mesa, junto con cualquier objeto de uso cotidiano en una oficina que pueda encontrarse sobre ella, actuará de fondo.

Las condiciones experimentales que hay que fijar son las siguientes:

- Posición de la cámara relativa a la mano y a la mesa.
- Condiciones de iluminación.
- Tipo de la cámara y sus parámetros.

Como cámara se utilizaron varios modelos, tanto analógicas como digitales. Se pretendía utilizar una cámara comercial convencional, de coste asequible. Se hicieron pruebas con varias cámaras y se eligió usar como referencia una cámara digital de laboratorio —Basler A300— de resolución intermedia, por la facilidad de ajuste y cantidad de parámetros sintonizables, para permitir investigar la influencia de éstos en los resultados de la adquisición. Respecto a las condiciones de iluminación, se pretendía que éstas fueran también las de cualquier entorno de oficina. Es decir, no se utilizó iluminación especial, sino únicamente se aseguró que los parámetros de la cámara respecto la iluminación permitieran una captura adecuada, sin excesiva saturación ni oscuridad de la escena. La iluminación fluorescente posee un perfil de intensidad sinusoidal no apreciable por la mayoría de las personas pero que afecta considerablemente a la adquisición. Esto impuso la restricción de que la tasa de fotogramas por segundo de la captura se eligiera de acuerdo a la frecuencia del fluorescente —100 Hz—. Por ello se determinó la frecuencia de adquisición de imágenes a 25 Hz en formato progresivo —no entrelazado—. Esta elección tiene implicaciones sobre la continuidad de movimientos rápidos, que podrían acarrear grandes diferencias entre dos imágenes consecutivas. La posición de la cámara se fijó mediante un trípode con brazo, de forma que no obstaculizara el uso normal del puesto de trabajo. Se fijó el enfoque de forma que la superficie de la mesa permaneciera en foco. En la figura 2.1 se puede observar un fotograma capturado desde la cámara donde se aprecia la posición de ésta respecto a la mesa y la mano, así como el contenido de la mesa y el tipo de iluminación.

2.3.1. Detección de la mano y del fondo

Para poder aplicar la primera parte de la estrategia esbozada anteriormente, los algoritmos de seguimiento de características deben de funcionar únicamente con características pertenecientes a la mano y no a ningún otro objeto que se recoja en la captura. A todo lo que se aprecie en la imagen, con la excepción de la mano, se le denominará fondo de la imagen y no aporta información útil al pro-

Figura 2.1. Vista desde la cámara de las condiciones experimentales.

blema. En la situación experimental descrita, el fondo puede tener una estructura compleja que se asume desconocida a priori. Por ello, una parte del desarrollo debe ir orientado a separar mano y fondo. Es decir, se puede intentar caracterizar o modelizar uno y otro y diseñar un clasificador que permita discriminarlos. En el caso de la mano, asumiendo que la mano no está cubierta por guantes o similares, existen numerosos trabajos dedicados a caracterizar la apariencia de la piel humana, como se verá más adelante.

En este punto nos centramos en establecer un modelo simple para el fondo, de manera que puntos alejados de ese modelo se puedan adjudicar a la mano. Se eligió un modelo gaussiano de color en espacio RGB, tal que cada píxel posee una distribución gaussiana de tres dimensiones, estimada a lo largo de una secuencia de imágenes sin presencia de la mano. Dado que la posición de la cámara y el fondo se asumen estacionarios, la fuente de variación aleatoria proviene de las fluctuaciones de la iluminación, del ruido del sensor de la cámara y de alguna posible vibración del sistema de sujeción de la misma. Para cada píxel se

estima su media y su matriz de covarianza, ambos en el espacio RGB. Aquí hay que destacar posibles dificultades en la estimación de la matriz de covarianza. Si algún píxel que enfoca a algún objeto muy claro se satura —situación bastante probable si no se ajusta el balance de blancos con cuidado— las demás fuentes de aleatoriedad no podrán sacarlo de esa situación y habrá un claro problema de estimación en la matriz de covarianza —bastaría con que una de las componentes de color no tuviera variación para hacer que la matriz sea singular—. En el caso contrario, de ausencia total de luz, el ruido de temperatura propio del sensor hace que no sea tan probable la singularidad, aunque sigue siendo posible. Una vez establecido el modelo de fondo, durante la captura con presencia de la mano, basta con calcular la probabilidad de que un píxel pertenezca al fondo usando la gaussiana determinada en esa posición. Si la probabilidad no supera un umbral prefijado, se concluirá que el píxel pertenece a la mano. De manera que al finalizar el proceso se tendrá una máscara binaria que se usará para eliminar características en el exterior de la zona considerada como mano. Este modelo se puede simplificar en varios sentidos. Por ejemplo, restringiendo la forma de la gaussiana o usando una representación en niveles de gris, en función de la carga computacional. En las pruebas realizadas, se constató que simplemente el error cuadrático de un píxel respecto a la media de la posición ya daba un indicador fiable sin los problemas asociados a la estimación de la gaussiana —motivados por la estimación de la matriz de covarianza—.

En la figura 2.2 se puede observar un ejemplo de lo comentado. Concretamente se grabaron dos secuencias de vídeo, una sin la mano y otra con la mano en movimiento. La primera se usó para calcular la media de cada píxel del fondo, con la que se compararon los píxeles de todos los fotogramas de la segunda, con el objetivo de recortar la mano. En la figura 2.2(a) se muestra, en una imagen en escala de grises, la distancia euclídea entre cada píxel de un fotograma de ese segundo vídeo y la media del mismo calculada a partir de la secuencia de vídeo de imágenes del fondo. En la figura 2.2(b) se puede ver el resultado de la extracción de dicho fondo umbralizando las distancias euclídeas de la figura 2.2(a).

(a) Distancia euclídea entre cada píxel y el modelo del fondo.

(b) Extracción del fondo por umbralizado de la distancia euclídea.

Figura 2.2. Ejemplo ilustrativo del proceso de extracción del fondo.

2.3.2. Experimentos con el algoritmo KLT: características y seguimiento

Conviene ser conscientes de las limitaciones del algoritmo KLT para tenerlas en cuenta en una implementación. La primera es que se parte de una situación con una sola cámara. En el mejor de los casos, la reconstrucción 3D del objeto sólo puede hacerse partiendo de las características de las caras que han estado visibles en la secuencia. Por tanto, es el movimiento del objeto el que permite ir revelando su geometría. Obviamente al rotar o desplazarse es inevitable que una parte de las características queden ocluidas, por lo que se perderán en el seguimiento. Al avanzar la secuencia y producirse más movimientos, el número de características decaerá haciendo más imprecisa la forma del objeto. Por ello es necesario introducir durante la captura nuevas características; tarea que no está prevista en el algoritmo KLT simple expuesto antes, ya que altera la elección del origen de los ejes de referencia.

Otro problema potencialmente más grave consiste en la falta de características en la mano con las propiedades exigidas a la matriz **G** para el seguimiento. En la figura 2.3(a) se han señalado en rojo los puntos que cumplen los criterios sobre los autovalores en toda la imagen. Como se aprecia, se detectan características buenas para el seguimiento en las teclas del teclado —debido a que son esquinas— y algunas más en otros puntos de la escena, pero prácticamente ninguna está sobre la mano. El tipo de textura de la mano a las resoluciones y distancia empleados, no constituye características adecuadas. Cambiando el parámetro de umbral para favorecer la aparición de un mayor número de características, se puede lograr que aparezca alguna sobre la mano, como se aprecia en la figura 2.3(b).

En todo caso el número de características es muy bajo y no permite delimitar claramente la geometría de la mano. Además, las características situadas en posiciones útiles aparecen en lugares donde pueden haber problemas, como en los bordes de los dedos. Esta situación es precisamente muy negativa para el seguimiento, ya que están en fronteras de oclusión que además son puntos candidatos a desaparecer en a una rotación de la mano, con lo cual o bien desaparecerán

(a)

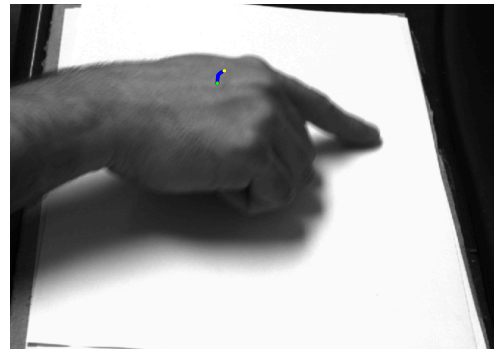
(b)

Figura 2.3. Extracción de características de la mano.

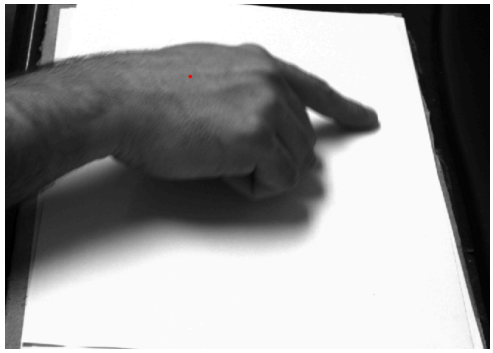
o bien es probable que sean seguidos defectuosamente. Experimentalmente se evaluaron estos problemas mediante el desarrollo de un sistema en tiempo real construido sobre la librería OpenCV (Bradski, 2000), con el que se constataron los defectos anteriores. Concretamente, se desarrolló un software que captura desde la cámara y aplica el KLT fotograma a fotograma para el seguimiento de características. El resultado se muestra por la pantalla del ordenador en tiempo real, sobre el vídeo capturado. Las imágenes de la figura 2.3 fueron realizadas con dicho programa, por lo que son un ejemplo de su funcionamiento. Aparte de mostrar el vídeo, el programa proporciona controles para ajustar diversos parámetros. Entre ellos está el umbral con el que se seleccionan las características. Esto permite ver en tiempo real el efecto que tienen los distintos parámetros en el funcionamiento del algoritmo.

Para probar las prestaciones del seguimiento, se eligieron características manualmente sobre una imagen de un vídeo de niveles de gris de una mano en movimiento y se aplicó el algoritmo para determinar la traslación a la imagen siguiente. Para comprobar su corrección, se realizó una búsqueda exhaustiva o de «fuerza bruta» para encontrar la mejor traslación. En las figuras 2.4 y 2.5 se ilustran estos experimentos con dos ejemplos. Concretamente, en las figuras 2.4(a) y 2.4(c) se puede observar una imagen de la mano donde se ha escogido un punto al que se le quiere hacer el seguimiento. En las figuras 2.4(b) y 2.4(d) la imagen anterior ha sido desplazada, por lo que el punto escogido —en amarillo— ya no está en el mismo lugar de la mano. Para hacer el seguimiento se ejecuta tanto el KLT como una búsqueda exhaustiva. Esta última lo que hace es encontrar la subimagen de la segunda imagen lo más parecida posible, en términos de la suma del error cuadrático, a la subimagen del punto escogido en la imagen original. El resultado del seguimiento mediante KLT se marca en verde, indicando los puntos azules los sucesivos pasos durante la interacción del algoritmo, mientras que el resultado de la búsqueda se indica con un punto rojo. Como se puede apreciar en las figuras 2.4(b) y 2.4(d), en ninguno de los casos la búsqueda exhaustiva tiene problemas en encontrar el punto correspondiente. Sin embargo, el KLT da con la solución correcta en el caso de la figura 2.4(b) pero

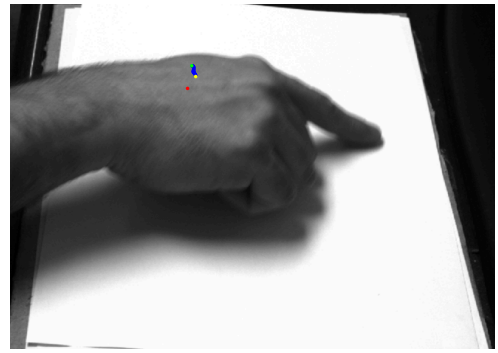
(a) Imagen original con el punto de interés escogido.



(b) Resultado del KLT en 21 iteraciones y de la búsqueda exhaustiva.

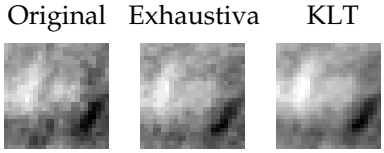


(c) Imagen original con el punto de interés escogido.

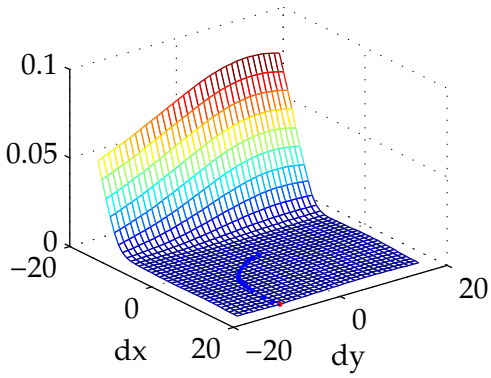


(d) Resultado del KLT en 33 iteraciones y de la búsqueda exhaustiva.

Figura 2.4. Ejemplos de las pruebas con el KLT para dos puntos distintos elegidos manualmente. A la izquierda, las imágenes originales con el punto de interés escogido en rojo. A la derecha la imagen desplazada, marcando en amarillo el punto escogido en la de la izquierda, en verde el punto finalmente asignado mediante el KLT, en azul los puntos de la trayectoria recorrida por el KLT durante la estimación y en rojo la correspondencia encontrada mediante búsqueda exhaustiva.

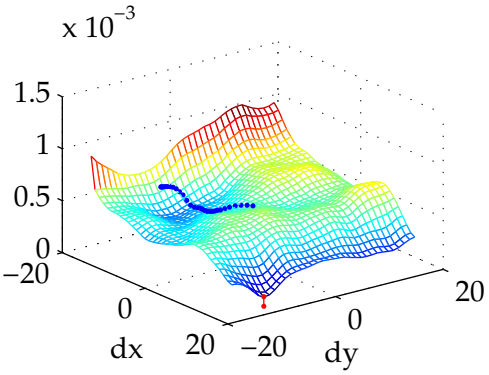


(a) Subimágenes para el caso de la figura 2.4(a).



(b) Superficie del error cuadrático para el caso de la figura 2.4(a). KLT estima la correspondencia en $dx = 17$ y $dy = -9$, mientras que la búsqueda exhaustiva la hace en $dx = 17{,}71$ y dy=$-9{,}71$.

(c) Subimágenes para el caso de la figura 2.4(b).



(d) Superficie del error cuadrático para el caso de la figura 2.4(b). KLT estima la correspondencia en $dx = 17$ y $dy = -11$, mientras que la búsqueda exhaustiva lo hace en $dx = -15{,}28$ y $dy = -5{,}36$.

Figura 2.5. A la izquierda, subimágenes de las ventanas en torno al punto original escogido, al punto encontrado mediante búsqueda exhaustiva y al asignado mediante KLT. A la derecha, superficies del error cuadrático entre la subimagen de la primera imagen, centrada en el punto escogido, y las subimágenes de la segunda, extraídas desplazando la ventana respecto al punto original. Sobre las curvas, en azul la trayectoria del KLT durante la optimización y en rojo la correspondencia encontrada mediante búsqueda exhaustiva.

no en el de la figura 2.4(d). De esto se desprende, que mientras que la primera de las características debe ser buena para el seguimiento, porque la textura de la subimagen tiene información suficiente, la segunda no lo es. En las figuras 2.5(a) y 2.5(c) se pueden observar las subimágenes tanto para el punto original como para las correspondencias asignadas finalmente, donde se puede apreciar lo que para el KLT es una textura adecuada —en la figura 2.5(a)— y lo que no lo es —la figura 2.5(c)—. Finalmente, en las figuras 2.5(b) y 2.5(c) se han trazado las superficies del error cuadrático calculado por la búsqueda exhaustiva para cada caso. En dichas figuras se puede observar el mínimo global encontrado por la búsqueda —en rojo— y los sucesivos pasos del KLT —en azul— hasta terminar asignando la correspondencia.

Tras realizar diversas pruebas se determinó que para alcanzar prestaciones similares a la búsqueda exhaustiva, se debían emplear ventanas relativamente grandes. Las características situadas en posiciones cercanas a bordes de oclusión hacen que el algoritmo no converja a una solución válida y que el criterio del error residual elevado termine con las iteraciones del proceso KLT.

Capítulo

3

Modelado de la mano

Tanto para poder entrenar como para poder probar la distintas técnicas de reconocimiento gestual es necesario disponer de diferentes manos en distintas posturas o haciendo diferentes gestos. Conseguir esto puede resultar extremadamente complejo. La forma más sencilla de obtener información acerca de la postura de una mano es mediante el uso de guantes de datos. Sin embargo, se puede considerar que su utilización es invasiva en el sentido de que el guante cubre la mano, por lo que difícilmente se pueden obtener imágenes al desnudo de la misma al tiempo que se capturan los datos de la postura. Además no parece que sea viable grabar las imágenes y capturar los datos de la postura por separado, puesto que quién realice el gesto difícilmente va a hacerlo de la misma manera en distintas ocasiones. En general se puede decir que no existe actualmente una técnica que permita capturar al mismo tiempo ambas fuentes de información, excepto quizás mediante el uso de una *cámara Z* (Gvili, 2003).

Una de las posibles soluciones a este problema consiste en desarrollar una mano sintética, que pueda ser fácilmente parametrizada, para obtener imágenes foto-realistas de la misma en distintas posturas. Los dos aspectos fundamentales para el desarrollo de dicha mano son la creación de un modelo cinemático, que se ajuste en lo posible al de las manos humanas, y un modelo de piel, que proporcione a la mano sintética el aspecto de una mano real.

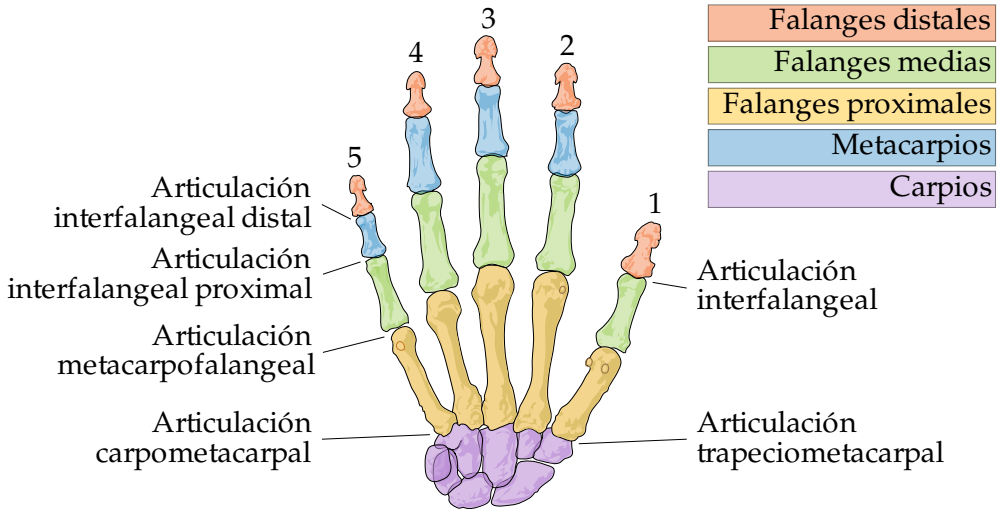


Figura 3.1. Ilustración de los huesos y las articulaciones de la mano.

3.1. Modelo cinemático

La mano humana está formada por diversas extremidades conectadas que componen distintas cadenas cinemáticas. En total hay 17 articulaciones activas[1] en una mano, lo que proporciona 23 grados de libertad. En la figura 3.1 se puede observar un esquema anatómico de los diferentes huesos y articulaciones de una mano humana y en la tabla 3.1 los grados de libertad de cada unión. El brazo se une a la mano a través de la articulación de la muñeca o radiocarpal (RC). En el lado de dicha articulación correspondiente a la mano hay un conjunto de huesos denominados *carpianos*. Estos huesos están unidos a los 5 *metacarpianos* —uno por dedo— que forman la palma de la mano. El metacarpiano del pulgar (1) se une a los carpianos mediante la articulación trapeciometacarpal (TM),

[1]Se entiende por articulaciones activas a aquellas que pueden moverse por la acción de los tendones y músculos de la mano, sin interacciones externas, mientras que las pasivas sólo se mueven por la acción de fuerzas externas. Generalmente las articulaciones presentan un mayor rango de movimiento cuando el origen del mismo es externo.

que proporciona tres grados de libertad; siendo responsable de los movimientos de flexión-extensión, abducción-aducción, circunducción y oposición. Mientras que cada metacarpo de los otros dedos se une a los carpianos mediante su articulación carpometacarpal (CM) —donde sólo las de los dedos meñique (5) y anular (4) son uniones activas— proporcionando un grado de libertad cada una. Cada hueso metacarpiano se une a la falange de su dedo correspondiente mediante la articulación metacarpofalangeal (MF). Estas articulaciones proporcionan un grado de libertad en el caso del pulgar y dos en el del resto de los dedos. Uno de dichos grados de libertad es el que permite el movimiento de flexión-extensión —para abrir y cerrar la mano— mientras que el otro es el responsable del de abducción-aducción —utilizado para separar y acercar los dedos unos a los otros. Cada dedo, con la excepción del pulgar[2] está formado por tres falanges: la proximal, la media y la distal; unidas por las articulaciones interfalangeal proximal (IFP) e interfalangeal distal (IFD). Cada una de dichas articulaciones proporciona un grado de libertad en cada dedo. En el caso del pulgar no hay falange medía, por lo que sólo existe una articulación interfalangeal (IF) que también proporciona un grado de libertad. A esos 23 grados de libertad hay que sumar los de la muñeca, que ofrece tres grados de libertad adicionales por ser una articulación esférica. El primero de los mismos es el que permite el movimiento de flexión-extensión —para plegar la mano sobre el brazo— mientras que el segundo es el responsable del de abducción-aducción —con el que se puede separar la mano lateralmente del eje del brazo— y el tercero permite la circunducción —es decir, girar la mano en redondo—.

Debido al número tan alto de grados de libertad, el problema del reconocimiento gestual es considerado como muy complejo. Además existen muchas restricciones entre los dedos y las articulaciones, lo que hace que el movimiento de la mano sea, si cabe, incluso más difícil de modelar. Varios autores han hecho diferentes propuestas sobre modelos de la mano humana. Por ejemplo, obviando los grados de libertad de la muñeca y la posición global de la mano, Vardy

[2] Al pulgar, en realidad, no se le considera un dedo puesto que sólo tiene dos falanges.

Tabla 3.1. Articulaciones de la mano y grados de libertad.

Articulación	Grados de libertad	Observaciones
radiocarpal (RC)	3	articulación de la muñeca.
trapeciometacarpal (TM)	3	sólo en el pulgar.
carpometacarpal (CM)	1	sólo es activa en el dedo anular y en el meñique.
metacarpofalangeal (MF)	1	la del pulgar.
metacarpofalangeal (MF)	2	la de los otros cuatro dedos.
interfalangeal (IF)	1	articulación entre falanges en todos los dedos.

(1998) propone un modelo de 26 grados de libertad basado en la convención de Denavit-Hartenberg. Respecto al esquema comentado anteriormente, Vardy (1998) establece un grado de libertad en la articulación carpometacarpal de todos los dedos. Además pone a la articulación metacarpofalangeal del pulgar dos grados de libertad, en lugar de uno. Esta propuesta resulta interesante desde el punto de vista de que modela la palma de la mano como si se tratase de una articulación de siete grados de libertad, lo que es importante cuando se pretende simular la forma en la que la mano sujeta los objetos. En todo caso el modelo no se puede considerar completo, puesto que las restricciones propuestas no son más que simples aproximaciones de las de una mano real. Un modelo muy similar es el de Yasumuro *et al.* (1997), cuya estructura es la misma excepto porque la unión trapeciometacarpal sólo tiene dos grados de libertad. Esto tiene mucho sentido puesto que, tal y como está hecha la articulación, el movimiento de oposición se consigue como una combinación de los de flexión-extensión y abducción-aducción, y no como un movimiento independiente (véase Gray, 1918, cap. III ap. 6.h). Este modelo cinemático se utiliza para crear a partir de superficies un modelo 3D de la mano humana que pueda ser animado, simulando el movimiento real en base a un modelo dinámico. Otra propuesta muy similar a éstas es la de Albrecht *et al.* (2003). La estructura es exactamente la

misma que la de Vardy (1998), excepto por los grados de libertad de las articulaciones carpometacarpales y trapeciometacarpal. En concreto la del pulgar tiene tres grados de libertad, las de los dedos anular y meñique tienen dos, y el medio y el índice ninguna. Al igual que con la propuesta de Vardy (1998), el objetivo de tener un modelo así es estudiar la forma en la que la mano sujeta los objetos. Por otro lado Wu y Huang (1999a) tratan la mano como un conjunto de objetos, cada uno de los cuales es modelado por separado. Estos objetos se relacionan cinemáticamente a través de una abstracción del esqueleto de la mano, donde la dimensión de cada uno se reduce a una línea a lo largo de su longitud. Cada dedo es modelado como una cadena cinemática donde la palma es el sistema de referencia, mientras que las puntas de los dedos son los efectores de su cadena cinemática correspondiente.

3.1.1. Restricciones al movimiento

Wu y Huang (1999a) introducen algunas restricciones adicionales al movimiento de la mano. Por ejemplo, establecen los ángulos máximos y mínimos para cada grado de libertad e incorporan el que las articulaciones interfalangeal distal e interfalangeal proximal realmente no son independientes. Esto es importante si se quiere reducir el espacio de búsqueda al aplicar cinemática inversa para inferir la postura de la mano a partir de la posición de los dedos. En general, algunas restricciones se pueden expresar de forma analítica, por lo que son comúnmente utilizadas en la animación y la captura del movimiento (Kuch y Huang, 1995; Kunii, 1995; Wu y Huang, 1999b), mientras que muchas otras son muy difíciles de expresar de dicha manera. En su momento Lin *et al.* (2000) investigaron los diferentes tipos de restricciones que existen y como se pueden utilizar para representar los movimientos de la mano, con menos grados de libertad de los vistos hasta el momento. Además propusieron una forma de aprender dichas restricciones para modelar el espacio de posturas de la mano, independientemente de que éstas se puedan expresar de forma analítica o no.

Las restricciones de la mano pueden ser divididas en tres tipos. Las restriccio-

nes de *tipo 1* —o restricciones estáticas— son los límites en el movimiento de los dedos debidos a la anatomía de la mano. Las del *tipo 2* —o restricciones dinámicas— son los límites de las articulaciones durante el movimiento. Finalmente, las restricciones del *tipo 3* se aplican durante el movimiento natural de la mano y no se deben a limitaciones de la anatomía, sino que se definen por la «naturalidad» de los movimientos.

Restricciones de tipo 1

Como ya se ha comentado, las restricciones de tipo 1 son aquellas que limitan el rango posible de los ángulos de las articulaciones debido a la anatomía de la mano. Este tipo de restricciones suelen indicarse mediante inecuaciones que fijen dichos rangos, por ejemplo

$$0^\circ \leq \theta_{MF\_F} \leq 90^\circ$$
$$-15^\circ \leq \theta_{MF\_AA} \leq 15^\circ$$

donde el subíndice F indica que se trata de una restricción sobre el movimiento de flexión-extensión y el AA sobre el de abducción-aducción, de la articulación metacarpofalangeal (MF). Otras restricciones comunes son las que dan cuenta del movimiento de abducción-aducción limitado del dedo medio $\theta_{MF(3)\_AA} = 0^\circ$ y del pulgar $\theta_{TM(1)\_AA} = 0^\circ$. En estas últimas se ha utilizado en el subíndice la numeración empleada en la figura 3.1 para indicar el dedo al que afecta la restricción. También es muy común imponer que todos los dedos, excepto el pulgar, son manipuladores planares. Es decir, que las articulaciones metacarpofalangeal, interfalangeal proximal y interfalangeal distal se flexionan en la misma dirección, ocupando las tres un mismo plano.

Restricciones de tipo 2

Este tipo hace referencia a aquellas restricciones impuestas sobre el movimiento de los dedos. Las restricciones de tipo 2 se pueden dividir en restricciones

intra-dedo e *inter-dedo*.

Las restricciones intra-dedo establecen límites entre las articulaciones de un mismo dedo. Por ejemplo, se suele asumir que existe una relación entre el ángulo en la articulaciones interfalangeal proximal e interfalangeal distal tal que

$$\theta_{IFP\_F} = \frac{2}{3}\theta_{IFD\_F} \tag{3.1}$$

para todos los dedos excepto el pulgar. Esto permite reducir notablemente los grados de libertad de la mano. Otra restricción inter-dedo es la que establece que el movimiento de abducción-aducción de todos los dedos está limitado por el ángulo de flexión. Es decir, los dedos se puede separar o aproximar libremente cuando están completamente extendidos, pero según se van flexionando se limita el rango de movimiento de abducción-aducción posible (Kunii, 1995). Esta restricción puede definirse mediante la expresión

$$\theta^{dmax}_{MF\_AA} = \left(1 - \frac{1}{\theta^{smax}_{MF\_F}}\right)\theta_{MF\_F}\,\theta^{smax}_{MF\_AA} \tag{3.2}$$

donde los superíndices $dmax$ y $smax$ hacen referencia a los ángulos máximos dinámicos —tipo 2— y estáticos —tipo 1— respectivamente, para la articulación correspondiente.

Las restricciones inter-dedo establecen límites entre las articulaciones de dedos adyacentes. Por ejemplo, si un dedo se flexiona, los otros se verán obligados a flexionarse. Kunii (1995) obtuvo experimentalmente un conjunto de inecuaciones que definen esta restricción

$$\begin{aligned}
\theta^{dmax}_{MF(2)_F} &= \min(\theta_{MF(3)_F} + 25^\circ, \theta^{smax}_{MF(2)_F}) \\
\theta^{dmin}_{MF(2)_F} &= \max(\theta_{MF(3)_F} - 54^\circ, \theta^{smin}_{MF(2)_F}) \\
\theta^{dmax}_{MF(3)_F} &= \min(\theta_{MF(2)_F} + 54^\circ, \theta^{smax}_{MF(3)_F}, \theta_{MF(4)_F} + 20^\circ) \\
\theta^{dmin}_{MF(3)_F} &= \max(\theta_{MF(2)_F} - 25^\circ, \theta^{smin}_{MF(3)_F}, \theta_{MF(4)_F} - 45^\circ)
\end{aligned}$$

$$\begin{aligned}
\theta^{dmax}_{MF(4)_F} &= \min(\theta_{MF(3)_F} + 45^\circ, \theta^{smax}_{MF(4)_F}, \theta_{MF(5)_F} + 48^\circ) \\
\theta^{dmin}_{MF(4)_F} &= \max(\theta_{MF(3)_F} - 20^\circ, \theta^{smin}_{MF(4)_F}, \theta_{MF(5)_F} - 44^\circ) \\
\theta^{dmax}_{MF(5)_F} &= \min(\theta_{MF(4)_F} + 44^\circ, \theta^{smax}_{MF(5)_F}) \\
\theta^{dmin}_{MF(5)_F} &= \max(\theta_{MF(4)_F} - 48^\circ, \theta^{smin}_{MF(5)_F})
\end{aligned} \tag{3.3}$$

En cualquier caso, existen muchas otras restricciones de tipo 2 que no pueden ser expresadas analíticamente, a diferencia de las aquí expuestas.

Restricciones de tipo 3

Las restricciones de tipo 3 vienen impuestas por la naturalidad del movimiento. No son restricciones debidas a posturas prohibidas para una mano humana, sino a que las posturas deben ser «naturales» desde el punto de vista de las que habitualmente se pueden observar en una mano real. Este tipo de restricciones son muy difíciles de detectar y de cuantificar, en tanto en cuanto en su definición entra en juego el concepto subjetivo de la naturalidad. Difícilmente pueden ser expresadas analíticamente y generalmente hay que detectarlas mediante el análisis del movimiento de manos reales de diferentes sujetos.

3.1.2. Modelo cinemático de la mano sintética

El modelo cinemático de la mano sintética con la que se generaron las diferentes imágenes utilizadas en los experimentos, se basa en el utilizado en el programa `MakeHuman` (Bastioni y Flerackers, 2009). Dicho software permite generar modelos 3D de humanoides en diversas posturas que posteriormente pueden ser importados en programas de modelado. Debido a la dificultad de modelar una mano humana real desde cero, se optó por tomar el `MakeHuman` para generar y exportar un modelo, importarlo posteriormente en el software de modelado `Blender` (Blender, 2009) y suprimir del mismo todas sus partes; excepto una mano, su brazo y su antebrazo. El modelo exportado desde `MakeHuman` incluye

tanto una envoltura que hace las veces de piel como un esqueleto. En el caso concreto de la mano, el modelo cinemático utilizado en dicha aplicación se basa en lo descrito por Dragulescu *et al.* (2007) y por (Sturman, 1992, cap. 2).

El modelo cinemático del `MakeHuman` fue modificado para adaptarlo a las necesidades del problema. En la figura 3.2 se puede observar un esquema del modelo cinemático utilizado finalmente para crear la mano sintética. Éste no sólo incluye diferentes elementos de la mano, sino también el codo y el hombro, aunque por el momento no han sido utilizados. Las principales características del modelo son:

- A diferencia de otros modelos que no la contemplan (Albrecht *et al.*, 2003; Vardy, 1998; Yasumuro *et al.*, 1997) se incluye la muñeca como una articulación esférica de 3 grados de libertad.

- Las articulaciones carpometacarpales no se incluyen en el modelo. En lugar de los cuatro metacarpos hay un único hueso *W* que hace las veces de metacarpo para toda la palma de la mano y que se mueve solidario con la articulación de la muñeca. El motivo de hacerlo así es que sólo es necesario incluir las articulaciones carpometacarpales si se busca un modelo preciso de la mano, especialmente para cosas tales como estudiar la forma en la que la mano sujeta distintos objetos. Como ese no es el caso, esos elementos no se han incluido, simplificando notablemente el modelo de forma similar a lo hecho por Dragulescu *et al.* (2007). Por tanto, el sistema de referencia tanto para las falanges proximales de los dedos como para el metacarpo del pulgar es el que está en la muñeca, a través del hueso *W*.

- La articulación trapeciometacarpal del pulgar contempla dos grados de libertad, tal y como como proponen Yasumuro *et al.* (1997). Estos grados se han escogido de forma que puedan modelar tanto el movimiento de flexión-extensión como el de abducción-aducción, puesto que el de oposición se puede conseguir como una combinación de ambos.

- Se modelaron las articulaciones metacarpofalangeales e interfalangeales

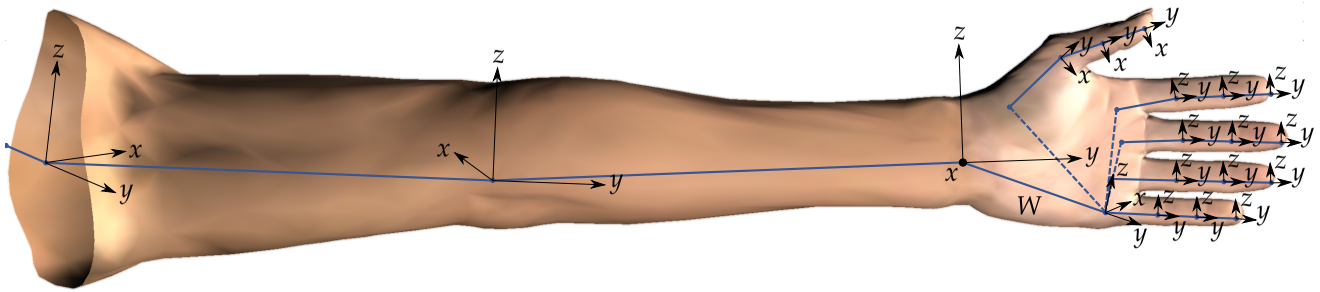


Figura 3.2. Esquema del modelo cinemático de la mano sintética; donde se pueden observar los diferentes segmentos rígidos, sus articulaciones y los sistemas de referencia en cada una. La coordenada Y de los sistemas de referencia siempre se fija a lo largo de los huesos, por imposición del programa de modelado `Blender`. Las rotaciones en el eje Z se utilizan para los movimientos de flexión-extensión, mientras que las rotaciones en el eje X se emplean para los movimientos de abducción-aducción. En el apéndice B en la página 183 se describen con mayor detalle las ecuaciones del modelo cinemático.

Tabla 3.2. Restricciones estáticas del modelo cinemático utilizado.

Grado de libertad	Ángulo mínimo	Ángulo máximo
θ_{RC_F}	$-90°$	$90°$
θ_{RC_AA}	$-25°$	$10°$
θ_{RC_C}	$-60°$	$90°$
θ_{TM_F}	$-42°$	$0°$
θ_{TM_AA}	$-35°$	$20°$
$\theta_{MF(1)\_F}$	$-30°$	$15°$
$\theta_{MF(2-4)\_F}$	$-70°$	$35°$
$\theta_{MF(2)\_AA}$	$-10°$	$15°$
$\theta_{MF(3)\_AA}$	$-10°$	$10°$
$\theta_{MF(4)\_AA}$	$-10°$	$10°$
$\theta_{MF(5)\_AA}$	$-20°$	$10°$
θ_{IF_F}	$-90°$	$0°$
$\theta_{IFP(2-4)\_F}$	$-90°$	$0°$
$\theta_{IFD(2-4)\_F}$	$-90°$	$0°$

tal y como comentó al principio del capítulo (véase el apartado 3.1 en la página 36), utilizando los mismos grados de libertad.

- Aunque Dragulescu *et al.* (2007) describieron el modelo cinemático de la mano utilizando la convención de Denavit-Hartenberg, esto se modificó para adaptarlo a los requerimientos del programa `Blender` de modelado 3D. En dicho programa siempre se fija el eje de coordenadas Y a lo largo del hueso que une ambas articulaciones, en lugar de utilizar el eje X como ocurre con Denavit-Hartenberg. Además se intentó que las rotaciones en el eje Z fueran para los movimientos de flexión-extensión, mientras que las rotaciones en el eje X fueran para los movimientos de abducción-aducción. En el apéndice B en la página 183 se describen las ecuaciones del modelo cinemático utilizado.

- Por construcción el modelo incluye la restricción acerca del movimiento

planar de los dedos (véase el apartado 3.1.1). Además en el programa de modelado `Blender` se incluyeron las restricciones de tipo 1 sobre los ángulos mínimos y máximos indicadas en la tabla 3.2.

- Puesto que el programa de modelado no lo permite, las restricciones de tipo 2 se incluyeron en un programa externo a éste encargado de generar los distintos parámetros de las posturas y verificarlos (véase el apartado 6.2 en la página 158). Las restricciones de tipo 2 incluidas fueron las de Kunii (1995), que fueron descritas en el apartado 3.1.1. Sin embargo la propuesta de Kunii (1995) y la de Dragulescu *et al.* (2007) difieren en los rangos de las restricciones estáticas y en la que consideran que es la postura de reposo[3]. Por consiguiente tuvieron que ser ajustadas al modelo utilizado tanto (3.2)

$$\theta^{dmax}_{MF\_AA} = \left(\frac{\theta_{MF\_F} - \theta^{smax}_{MF\_F}}{\theta^{smax}_{MF\_F} - \theta^{smin}_{MF\_F}} \right) \theta^{smax}_{MF\_AA} \tag{3.4}$$

como (3.3)

$$\begin{aligned}
\theta^{dmax}_{MF(2)_F} &= \min(\theta_{MF(3)_F} + 9^\circ, \theta^{smax}_{MF(2)_F}) \\
\theta^{dmin}_{MF(2)_F} &= \max(\theta_{MF(3)_F} + 76^\circ, \theta^{smin}_{MF(2)_F}) \\
\theta^{dmax}_{MF(3)_F} &= \min(\theta_{MF(2)_F} - 16^\circ, \theta^{smax}_{MF(3)_F}, \theta_{MF(4)_F} + 13^\circ) \\
\theta^{dmin}_{MF(3)_F} &= \max(\theta_{MF(2)_F} - 51^\circ, \theta^{smin}_{MF(3)_F}, \theta_{MF(4)_F} - 69^\circ) \\
\theta^{dmax}_{MF(4)_F} &= \min(\theta_{MF(3)_F} - 9^\circ, \theta^{smax}_{MF(4)_F}, \theta_{MF(5)_F} - 11^\circ) \\
\theta^{dmin}_{MF(4)_F} &= \max(\theta_{MF(3)_F} + 47^\circ, \theta^{smin}_{MF(4)_F}, \theta_{MF(5)_F} + 68^\circ) \\
\theta^{dmax}_{MF(5)_F} &= \min(\theta_{MF(4)_F} - 8^\circ, \theta^{smax}_{MF(5)_F}) \\
\theta^{dmin}_{MF(5)_F} &= \max(\theta_{MF(4)_F} + 71^\circ, \theta^{smin}_{MF(5)_F})
\end{aligned} \tag{3.5}$$

[3] En este contexto el término «postura de reposo» hace referencia a aquella postura donde todos los parámetros de la misma están a 0. Mientras que Kunii (1995), Wu y Huang (1999a) y otros la fijan con los dedos extendidos al máximo, Dragulescu *et al.* (2007) escogen una postura más parecida a la de reposo de una mano real.

También se incluyó en el programa externo la restricción de tipo 1 (3.1) entre la articulación interfalangeal proximal y la interfalangeal distal, debido a las limitaciones del programa `Blender` de modelado 3D.

- No se incluyeron restricciones de tipo 3 al carecer de datos sobre manos humanas en movimiento que utilizar para el aprendizaje.

3.2. Modelo de la piel

El modelo exportado desde el programa `MakeHuman` incluye tanto un esqueleto articulado como un modelo poligonal de la piel. Tras importarlos en el programa `Blender` de modelado 3D, ambos deben vincularse para que los movimientos de los huesos deformen la piel de forma realista. Aunque existen diversas técnicas, el procedimiento más sencillo consisten en asignar un peso a cada polígono para cada hueso, en función de cuanto debe afectar el movimiento del hueso al polígono. Un peso de 1,0 significa que el polígono debe moverse solidariamente con el hueso, mientras que un peso de 0,0 implica que el polígono permanecerá en su posición original aunque el hueso se mueva. Por tanto, la mayor parte de los polígonos de la piel tienen un peso de 1,0 para el hueso de la extremidad a la que pertenecen y un 0,0 para todos los demás. El único aspecto complejo de esta técnica es la asignación de los pesos sobre las articulaciones. En ese caso puede ser necesario que a un mismo polígono se le asignen distintos pesos de diferentes huesos, con el objetivo de simular la manera en la que la piel real se estira y se pliega. Esta técnica es demasiado sencilla para generar resultados realistas en todas las posturas —para lo que sería necesario utilizar técnicas más sofisticadas, como las curvas de interpolación (IPO) de la deformación— por lo que en ocasiones se puede observar que la piel se deforma de manera un tanto artificiosa. Sin embargo funciona muy bien en la mayor parte de los caso, siendo lo suficientemente buena como para generar manos sintéticas que puedan ser utilizadas en la experimentación con técnicas de reconocimiento gestual.

Tras vincular el modelo poligonal de la piel al esqueleto articulado es nece-

sario establecer la apariencia de la mano sintética. Para obtener unos resultados realistas no basta con elegir un color adecuado. Diferentes materiales tienen distintas propiedades físicas que les confieren una apariencia particular. En este sentido la piel humana no es diferente a cualquier otro material, ya sea natural o sintético. En concreto, para obtener un buen resultado es necesario elegir un modelo reflexivo y una textura que sean lo más parecidos posible a los de la piel humana real.

3.2.1. Modelo reflexivo de la piel

El color no es más que la sensación producida por la luz al impresionar los receptores luminosos de los órganos visuales —los ojos— en función de la distribución de la energía respecto a la longitud de onda. En una escena cualquiera, la luz que proviene de las diferentes fuentes de iluminación tiene una densidad espectral de potencia que se modifica al reflejarse sobre las superficies de los objetos, en función de las propiedades espectrales de la reflectancia de dichas superficies. Puesto que distintos materiales suelen tener diferentes propiedades espectrales, la luz que incide en los receptores luminosos tiene una densidad espectral de potencia distinta, generando por tanto una sensación de color diferente, en función del material de la superficie sobre la que fue reflejada, independientemente de que toda la luz tenga su origen en una misma fuente luminosa. Pese a que rigurosamente hablando la definición del término «color» va unida al uso de los receptores luminosos de los órganos visuales, en general también se habla de color para hacer referencia a la respuesta de cualesquiera otros sensores luminosos, como por ejemplo los instalados dentro de cualquier tipo de cámara. A efectos prácticos, la respuesta de un sensor i frente a un estímulo luminoso viene determinada por

$$c_i = \int r(\lambda) f_i(\lambda)\, d\lambda \tag{3.6}$$

donde $r(\lambda)$ es la densidad espectral de potencia de la luz incidente en el sensor i, mientras que $f_i(\lambda)$ describe la respuesta espectral de dicho sensor respecto a

la longitud de onda. Generalmente en un mismo dispositivo hay varios tipos de sensores con diferentes curvas de respuesta espectral, que comúnmente se diseñan para que cada uno opere como un filtro pasabanda centrado en alguna longitud de onda concreta. El número de bandas o canales del dispositivo depende de la aplicación. Lo más habitual cuando se pretende captar imágenes de una escena en color es el uso de tres canales, centrados en el rojo —canal R—, el verde —canal G— y el azul —canal B— respectivamente. Este es, por ejemplo, el caso de los receptores luminosos presentes en el ojo humano. La elección de estos tres canales es debido a que con ellos, y según los principios de la teoría del color (Hunt, 2004), se puede reconstruir con exactitud el color de cualquier objeto.

Como se ha comentado, la densidad espectral de potencia $r(\lambda)$ depende tanto de la densidad espectral de potencia de la iluminación de la escena, como de las propiedades espectrales de la superficie del objeto sobre la que se refleja la luz. La expresión exacta que relaciona estas magnitudes puede ser bastante compleja, en función de las características de la superficie. Sin embargo es muy común aproximar la solución considerando algún modelo sencillo de reflectancia. Por ejemplo, si se supone que todas las superficies de los objetos de la escena son lambertianas[4], la respuesta del sensor i vendría determinada por

$$c_i = (\boldsymbol{e} \cdot \boldsymbol{n}) \int \rho(\lambda) e(\lambda) f_i(\lambda)\, d\lambda \tag{3.7}$$

donde $e(\lambda)$ es la iluminación incidente —supuesta constante en toda la escena— en la dirección de $\boldsymbol{e}$, $\rho(\lambda)$ es la reflectancia de la superficie en un punto $\boldsymbol{x}$ y $\boldsymbol{n}$ es un vector normal a dicha superficie en el mismo punto $\boldsymbol{x}$.

De (3.7) se puede deducir que la respuesta de cada sensor depende de una serie de factores inherentes a la escena; siendo estos la iluminación, la reflectancia de la superficie y la orientación relativa entre ambas —presente a través del pro-

[4]Las superficies lambertianas son aquellas para las que se cumple la ley de Lambert. Dicha ley dice que en esas superficies la luz incidente se dispersa por igual en todas las direcciones, dependiendo solamente el total de la luz reflejada del ángulo de incidencia de la iluminación. Se suele utilizar para modelar la reflectividad de las superficies de apariencia mate.

ducto escalar $(\boldsymbol{e} \cdot \boldsymbol{n})$—. La ventaja de disponer de un modelo para la composición del color en las imágenes es que hace posible normalizar la respuesta del sensor, para obtener nuevas magnitudes que sean invariantes respecto a aquellas propiedades de la escena que no sean de interés. Por ejemplo, se puede eliminar la dependencia de la respuesta del sensor c_i respecto a la orientación relativa entre la superficie y la iluminación, normalizando la respuesta para cada canal i respecto a la suma de las respuestas para todos los canales

$$\frac{c_i}{\sum_i c_i} = \frac{(\boldsymbol{e} \cdot \boldsymbol{n}) \int \rho(\lambda) e(\lambda) f_i(\lambda)\, d\lambda}{(\boldsymbol{e} \cdot \boldsymbol{n}) \sum_i \int \rho(\lambda) e(\lambda) f_i(\lambda)\, d\lambda} = \frac{\int \rho(\lambda) e(\lambda) f_i(\lambda)\, d\lambda}{\sum_i \int \rho(\lambda) e(\lambda) f_i(\lambda)\, d\lambda} \tag{3.8}$$

Si los canales utilizados fueran R, G y B; se obtendría

$$\left(\frac{c_R}{c_R + c_G + c_B}, \frac{c_G}{c_R + c_G + c_B}, \frac{c_B}{c_R + c_G + c_B} \right) \tag{3.9}$$

lo que se conoce como coordenadas de color normalizadas (NCC). En todo caso, es importante destacar que en general esta normalización no elimina la dependencia de (3.7) respecto a la iluminación, pues $e(\lambda)$ no se puede simplificar en (3.8).

Estas expresiones derivan de la hipótesis inicial de que las superficies de los objetos de la escena son lambertianas. Sin embargo, previamente se comentó que lo que realmente se busca es disponer de un modelo de la piel humana que permita generar imágenes de una mano sintética. Por tanto es necesario analizar las propiedades espectrales de la piel, para saber si se comporta como una superficie lambertiana y determinar así si las expresiones anteriores puden ser útiles para desarrollar dicho modelo.

Propiedades espectrales de la piel humana

La piel tiene una estructura compleja que esencialmente puede ser dividida en dos capas, la *epidermis* y la *dermis* (véase la figura 3.3). Cuando la luz incide sobre la superficie de la epidermis ésta se refleja aproximadamente en un 5 %, in-

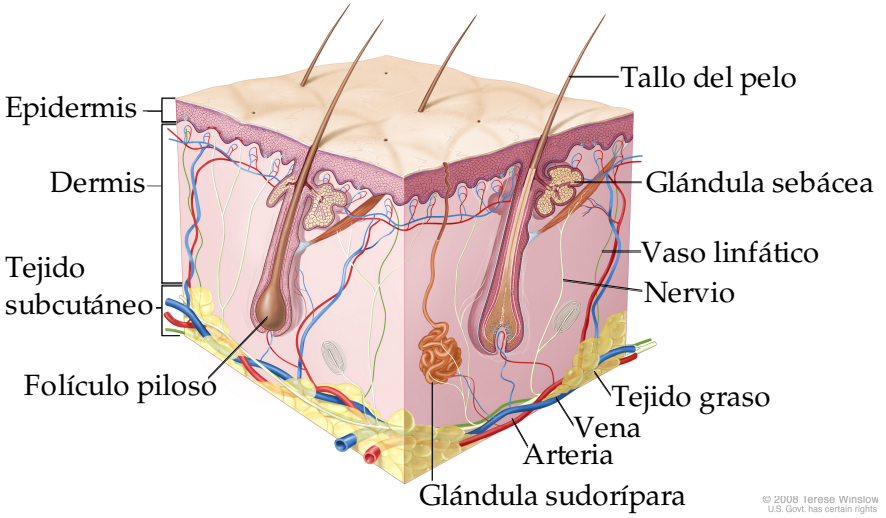


Figura 3.3. Ilustración de la piel y de sus capas.

dependientemente de la longitud de onda y del grupo étnico al que pertenezca el individuo (Anderson *et al.*, 1981). El resto de la luz incidente es en su mayor parte absorbida por la epidermis —que actúa como un filtro óptico— transmitiendo una parte de ella en función de la longitud de onda y de la concentración de melanina en la piel. Al llegar a la dermis la luz es tanto dispersada como absorbida, siendo este último fenómeno debido a componentes de la sangre como la hemoglobina, la bilirubina y el betacaroteno. La dispersión, por el contrario, hace que la luz vuelva a ser transmitida por la epidermis y reflejada hacia el exterior. En general se puede afirmar que las propiedades ópticas de la dermis son las mismas para todos los grupos étnicos. Por tanto, la reflectancia de la piel puede ser modelada como una función de la concentración de melanina en la epidermis y de la cantidad de sangre en la dermis (Ohtsuki y Healey, 1999; Störring, 2001), dependiendo el color de la piel fundamentalmente de la concentración de melanina. Es decir, que la piel de un individuo de raza negra sólo se distingue de la de otro de raza blanca en que tiene una mayor concentración de melanina en la

epidermis.

Si bien la absorción en la epidermis y la dispersión en la dermis, conjuntamente, pueden ser aproximadas por la ley de Lambert, ésta no se cumple para el porcentaje de luz que se refleja en la superficie de la epidermis. La ley de Lambert se suele utilizar para modelar la reflectividad de las superficies de apariencia mate, pero el porcentaje que se refleja en la superficie de la epidermis hace que la piel no tenga dicha apariencia. Por tanto es necesario buscar un modelo que se ajuste mejor al comportamiento real de la luz sobre la piel, como por ejemplo, el modelo dicromático de reflexión.

Modelo dicromático de reflexión

En el modelo dicromático de reflexión (MDR) la densidad espectral de potencia $r(\lambda)$ de la luz reflejada es la suma de la que se refleja en el cuerpo $r_{body}(\lambda)$ y la que se refleja en la superficie $r_{surf}(\lambda)$ del mismo (Klinker *et al.*, 1988; Shafer, 1985)

$$r(\lambda) = r_{surf}(\lambda) + r_{body}(\lambda) \tag{3.10}$$

En general, y de forma similar a lo hecho para obtener (3.7), ambas densidades pueden ser descompuestas en términos de la reflectancia, la densidad espectral de potencia de la luz incidente y un factor de escala que depende de la geometría

$$r(\theta, \lambda) = m_{surf}(\theta)\rho_{surf}(\lambda)e(\lambda) + m_{body}(\theta)\rho_{body}(\lambda)e(\lambda) \tag{3.11}$$

donde $m_{surf}(\theta)$ y $m_{body}(\theta)$ son los factores de escala dependientes de la geometría θ, para la densidad espectral de potencia de la luz reflejada en la superficie $\rho_{surf}(\lambda)$ y en el cuerpo $\rho_{body}(\lambda)$ respectivamente. En general la geometría descrita por θ incluye el ángulo de incidencia de la luz, el ángulo entre la luz incidente y la luz reflejada —o ángulo de fase— y el ángulo de visión. El término $r_{surf}(\lambda)$ se utiliza para modelar los reflejos en la superficie del cuerpo, por lo que $m_{surf}(\theta)$ se debe escoger utilizando algún modelo físico que describa este fenómeno según la geometría de la escena y las propiedades de la superficie (Beckmann y

Spizzichino, 1987; Phong, 1975; Shafer, 1984; Torrance y Sparrow, 1992), mientras que la reflectividad $\rho_{surf}(\lambda)$ se suele tomar como independiente respecto a la longitud de onda, puesto que así ocurre para la mayor parte los materiales. Esto último permite que la densidad espectral de potencia de la luz reflejada $r_{surf}(\lambda)$ en la superficie tenga la misma distribución espectral de potencia que la luz incidente $e(\lambda)$. La luz que no es reflejada en la superficie penetra en el cuerpo, donde es dispersada y absorbida, de forma que una pequeña parte de la misma $r_{body}(\lambda)$ llega a la superficie y escapa del material. Esto es modelado en el MDR siguiendo la ley de Lambert. Por lo que

$$m_{body}(\theta) = \boldsymbol{e} \cdot \boldsymbol{n} \tag{3.12}$$

puesto que según dicha ley la reflectividad sólo depende, geométricamente hablando, del ángulo entre la luz incidente y la normal a la superficie. La luz que viaja a través del cuerpo es absorbida en las longitudes de onda que son características del material, por lo que es $\rho_{body}(\lambda)$ quién fija el color característico del cuerpo (Kittler e Illingworth, 1986).

Debido a las similitudes entre el comportamiento real de luz sobre la piel y el MDR es factible utilizar éste último para obtener un modelo reflexivo de la piel. Por ejemplo, Störring *et al.* (1999) definen las reflectividades para la piel según las expresiones

$$\rho_{surf}(\lambda) = 0{,}5$$
$$\rho_{body}(\lambda) = \tau_{epi}^2 + \rho_{der}$$

donde τ_{epi} y ρ_{der} son la transmitancia de la epidermis y la reflectancia de la dermis respectivamente. Mientras que otros autores prefieren simplificar la componente especular y quedarse sólo con el modelo lambertiano. En cualquier caso estos modelos no sólo permiten generar imágenes foto-realistas de la mano, sino que también permiten desarrollar modelos estadísticos basados en principios físicos (Ohtsuki y Healey, 1999; Störring, 2001) o modelos invariantes frente a la

variación de los parámetros del entorno (Finlayson y Schaefer, 2001; Störring *et al.*, 1999, 2000) que pueden ser utilizados para la detección de regiones de piel.

3.2.2. Apariencia de la mano sintética

Para modelar la apariencia se escogió el MDR, con algunas modificaciones comúnmente utilizadas a la hora de modelar piel humana:

- El modelo físico escogido para $m_{surf}(\theta)$ fue el propuesto por Cook y Torrance (1981). Este modelo tiene en cuenta el brillo relativo entre los diferentes materiales y luces de la escena. Además permite describir la distribución de la luz reflejada, así como la variación de color de ésta según cambia la reflectancia con el ángulo de la luz incidente. El modelo físico puede ser ajustado modificando el coeficiente de especularidad y el ancho de los destellos especulares. Estos parámetros fueron fijados a 0,05 y 10 respectivamente.

- El modelo físico escogido para $m_{body}(\theta)$ no fue el lambertiano sino el propuesto por Oren y Nayar (1995), que puede ser considerado una generalización del primero. Aunque el modelo lambertiano ha sido ampliamente utilizado en visión por computador, resulta ser inadecuado para modelar muchas superficies reales. El motivo fundamental es que no tiene en cuenta la rugosidad de éstas. El modelo de Oren y Nayar (1995) sí predice adecuadamente la reflectancia de las superficies rugosas, teniendo en cuenta fenómenos físicos complejos como el sombreado y las interreflexiones entre puntos de una misma superficie. El modelo físico puede ser ajustado modificando los coeficientes de reflexión y de rugosidad. Estos parámetros fueron fijados a 0,95 y 0,35 respectivamente.

- El color de la piel se estableció ajustando la densidad espectral de potencia de la luz reflejada en la superficie $\rho_{surf}(\lambda)$ y en el cuerpo $\rho_{body}(\lambda)$. Aunque la primera suele ser independiente respecto a la longitud de onda, se optó por que ambas fueran iguales. En concreto se fijaron los pesos para los tres

Figura 3.4. Muestra de la textura de la piel utilizada en el modelado de la mano sintética.

canales de color —rojo, verde y azul— como 0,94, 0,82 y 0,73 respectivamente

Los parámetros comentados fueron escogidos considerando que se deseaba reproducir la piel de un individuo promedio de raza blanca.

Finalmente se añadió un modelo para simular la textura, que fue proporcionado por el propio programa `MakeHuman`. En la figura 3.4 se puede observar una muestra de la misma.

Capítulo

4

Análisis de componentes principales basado en kernel

El análisis de componentes principales (PCA) (Jolliffe, 2002) es una técnica estándar en el análisis de datos que consiste en transformar un número de variables correladas en un número, generalmente menor, de variables incorreladas. Para ello el PCA genera un subespacio donde la dirección de la mayor varianza del conjunto de datos es capturada en la primera dimensión, la dirección de la segunda mayor varianza en la segunda dimensión, y así sucesivamente. El conjunto de datos originales es proyectado en el nuevo subespacio mediante una transformación ortogonal lineal, de forma que se retienen aquellas características del conjunto de datos que contribuyen más a la varianza. Intuitivamente se entiende que así se puede reducir la dimensionalidad de los datos, al tiempo que se revela la estructura oculta subyacente a los mismos. Es decir, se extraen los factores que explican la variabilidad de los mismos.

Por ejemplo, en la figura 4.1, se puede observar un conjunto de datos $\{\boldsymbol{x}_n\}$ en $\mathbb{R}^2$, donde $n = 1, \ldots, N$, así como el vector $\boldsymbol{v}$ que indica la dirección de mayor varianza de los datos. Este vector $\boldsymbol{v}$ define un subespacio en $\mathbb{R}$ sobre el que el PCA proyectará cada punto $\boldsymbol{x}_n$, para obtener así el conjunto de datos proyectado $\{y_n\}$. Si bien el PCA no puede dar un significado a esta nueva variable y en $\mathbb{R}$, de la que parece que depende la variabilidad del conjunto de datos original, que intuitivamente se puede entender que es un factor subyacente al proceso que

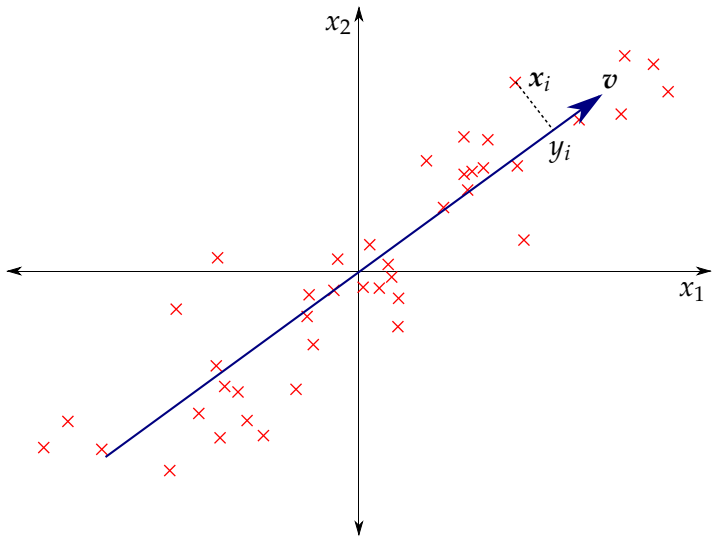


Figura 4.1. Mediante PCA se puede obtener v, la componente principal del conjunto de datos $\{\boldsymbol{x}_n\}$ en $\mathbb{R}^2$. Posteriormente cada punto $\boldsymbol{x}_i$ es proyectado sobre v en y_i.

generó el conjunto de datos $\{\boldsymbol{x}_n\}$.

Como ya se ha comentado, el PCA proyecta el conjunto de datos en el nuevo subespacio mediante una transformación ortogonal lineal. Por lo tanto, podrá extraer la estructura oculta subyacente a los mismos siempre que el proceso que los generó tenga la misma naturaleza. En el caso de que no sea así, eso no será posible.

En la figura 4.2 a la derecha se puede observar otro conjunto de datos $\{\boldsymbol{x}_n\}$ en $\mathbb{R}^2$. Si en ese caso se aplicara el PCA, difícilmente se obtendría como resultado un subespacio donde cada dimensión correspondiera a alguno de los factores ocultos que explican la variabilidad de los datos. Lo que si sería posible es mapear cada punto del conjunto de datos $\{\boldsymbol{x}_n\}$, mediante una transformación no lineal $\Phi(x) : \mathbb{R}^2 \longrightarrow \mathbb{F}$, en el espacio denominado de características $\mathbb{F}$ de dimensión M, de forma que cada punto $\boldsymbol{x}_n$ es proyectado en un punto $\Phi(\boldsymbol{x}_n)$. Si la transformación $\Phi(x)$ es escogida adecuadamente, el espacio de características $\mathbb{F}$ sería un

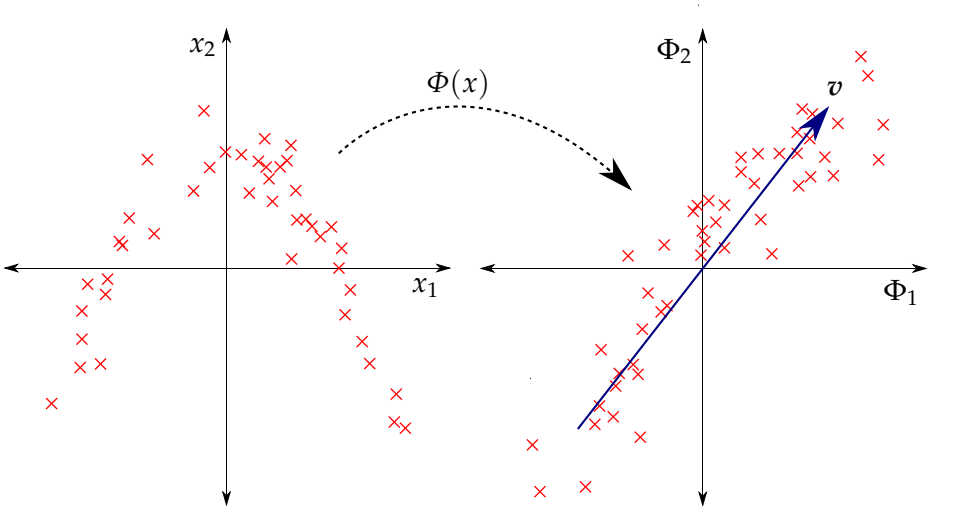


Figura 4.2. En el KPCA un conjunto de datos $\{\boldsymbol{x}_n\}$ en $\mathbb{R}^2$ —izquierda— es proyectado mediante $\Phi(x)$ en el espacio de características —derecha— donde se puede obtener v, la dirección de la componente principal del conjunto de datos $\{\Phi(\boldsymbol{x}_n)\}$.

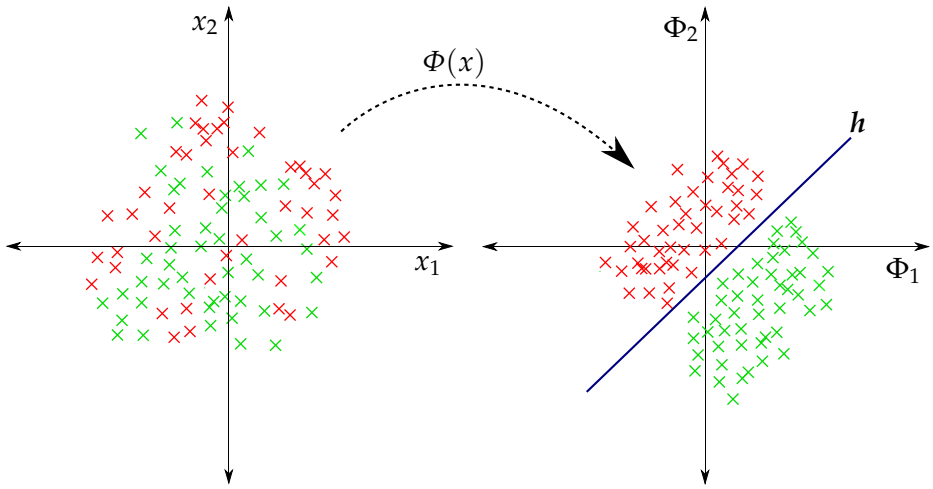


Figura 4.3. En las SVM dos conjuntos de datos difíciles de clasificar en $\mathbb{R}^2$ —izquierda— son proyectados mediante $\Phi(x)$ en el espacio de características —derecha—, donde pueden ser clasificados empleando un hiperplano $\boldsymbol{h}$ como frontera.

espacio de mayor dimensionalidad, donde la relación entre el conjunto de datos proyectado $\{\Phi(\boldsymbol{x}_n)\}$ y los factores subyacentes a la variabilidad del conjunto de datos original $\{\boldsymbol{x}_n\}$ sería lineal. Por tanto, es de suponer que aplicando un PCA estándar en el espacio de características $\mathbb{F}$ se podrían recuperar dichos factores, tal y como se muestra en la figura 4.2. A este tipo de análisis se lo denomina análisis de componentes principales basado en kernel (KPCA) (Schölkopf *et al.*, 1998).

El KPCA, junto con diversas técnicas que siguen el mismo procedimiento de mapear los datos en un espacio características mediante una transformación no lineal, se agrupan en lo que se denomina métodos basados en kernel. Por ejemplo, otra técnica basada en kernel son las máquinas de soporte vectorial (SVM), que se utilizan con vistas a la clasificación. En la figura 4.3 a la derecha se pueden observar dos conjuntos de datos en $\mathbb{R}^2$. En principio sería muy complicado entrenar un clasificador en $\mathbb{R}^2$ que pudiera separar ambos conjuntos. Sin embargo, cada punto puede ser mapeado mediante una transformación no lineal $\Phi(x)$ en

un espacio de características $\mathbb{F}$ de dimensión M. Si la transformación no lineal se escoge adecuadamente, ambos conjuntos de datos serían fácilmente separables utilizando como frontera un hiperplano $\boldsymbol{h}$ en $\mathbb{F}$. Una ilustración esquemática de este procedimiento puede observarse en la figura 4.3.

4.1. Fundamentos del KPCA

Suponiendo que se dispone de un conjunto de datos $\{\boldsymbol{x}_n\}$, donde $n = 1, \ldots, N$, de media $\mathbf{0}$, se puede utilizar el estimador insesgado para calcular la matriz de covarianza $\mathbf{C}$ como

$$\mathbf{C} = \frac{1}{N-1} \sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^T \tag{4.1}$$

En el caso de querer aplicar PCA bastaría con resolver el problema de los autovalores para la matriz de covarianza $\mathbf{C}$

$$\mathbf{C}\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i \qquad \text{para } i = 1, \ldots, D \tag{4.2}$$

donde D es la dimensión de los datos $\boldsymbol{x}_n$. Las componentes principales del PCA vienen definidas por el conjunto de autovectores $\{\boldsymbol{v}_i\}$. Éstos habitualmente se normalizan, de forma que $\boldsymbol{v}_i^T \boldsymbol{v}_i = 1$.

Sin embargo, como se ha comentado anteriormente, en el KPCA cada punto $\boldsymbol{x}_n$ del espacio de entrada de dimensión D es mapeado en un espacio de características $\mathbb{F}$ de dimensión M —donde $M > D$— mediante una transformación no lineal $\Phi(x) : \mathbb{R}^D \longrightarrow \mathbb{F}$. En ese caso se puede definir la matriz de covarianza $\mathbf{C} \in \mathbb{R}^{M \times M}$ como

$$\mathbf{C} = \frac{1}{N-1} \sum_{n=1}^{N} \Phi(\boldsymbol{x}_n) \Phi(\boldsymbol{x}_n)^T \tag{4.3}$$

siempre que la media del conjunto de datos proyectado en el espacio de características sea $\mathbf{0}$. Es decir, que $\sum_{n=1}^{N} \Phi(\boldsymbol{x}_n) = \mathbf{0}$. De la misma manera se puede

definir la descomposición en autovectores de la matriz de covarianza **C** como

$$\mathbf{C}\boldsymbol{v}_i = \lambda_r \boldsymbol{v}_i \qquad \text{para } i = 1, \ldots, M \tag{4.4}$$

El objetivo es resolver este problema de autovalores sin tener que trabajar en el espacio de características, porque éste puede ser de una dimensionalidad tan elevada —incluso infinita— que resulte inmanejable y porque generalmente no se conoce $\Phi(x)$ de forma explícita.

A partir de la definición de la matriz de covarianza (4.3) se puede reescribir el problema de autovalores (4.4)

$$\frac{1}{N-1}\sum_{n=1}^{N} \Phi(\boldsymbol{x}_n)\{\Phi(\boldsymbol{x}_n)^T \boldsymbol{v}_i\} = \lambda_i \boldsymbol{v}_i \qquad \text{para } i = 1, \ldots, M \tag{4.5}$$

Puesto que **C** es por construcción una matriz simétrica semidefinida positiva, el problema sólo tiene que ser resuelto para $\lambda \geq 0$. Además, observando (4.5) detenidamente, resulta obvio que para que haya solución los vectores $\{\boldsymbol{v}_i\}$ deben ser una combinación lineal de los datos proyectados $\{\Phi(\boldsymbol{x}_n)\}$. Es decir, que pueden ser escritos como

$$\boldsymbol{v}_i = \sum_{n=1}^{N} a_{in}\Phi(\boldsymbol{x}_n) \tag{4.6}$$

Sustituyendo (4.6) en (4.5) se obtiene

$$\frac{1}{N-1}\sum_{n=1}^{N} \Phi(\boldsymbol{x}_n)\Phi(\boldsymbol{x}_n)^T \sum_{m=1}^{N} a_{im}\Phi(\boldsymbol{x}_m) = \lambda_i \sum_{l=1}^{N} a_{il}\Phi(\boldsymbol{x}_l) \tag{4.7}$$

Puesto que se busca evitar trabajar en el espacio de características, se multiplica ambos lados de (4.7) por $\Phi(\boldsymbol{x}_j)^T$ para que la expresión quede en términos del producto escalar de los puntos $\Phi(\boldsymbol{x}_n)$

$$\frac{1}{N-1}\sum_{n=1}^{N} \Phi(\boldsymbol{x}_j)^T\Phi(\boldsymbol{x}_n) \sum_{m=1}^{N} a_{im}\Phi(\boldsymbol{x}_n)^T\Phi(\boldsymbol{x}_m) = \lambda_i \sum_{l=1}^{N} a_{il}\Phi(\boldsymbol{x}_j)^T\Phi(\boldsymbol{x}_l) \tag{4.8}$$

o definiendo la función de kernel como $k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \Phi(\boldsymbol{x}_n)^T \Phi(\boldsymbol{x}_m)$

$$\frac{1}{N-1} \sum_{n=1}^{N} k(\boldsymbol{x}_j, \boldsymbol{x}_n) \sum_{m=1}^{N} a_{im} k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \lambda_i \sum_{l=1}^{N} a_{il} k(\boldsymbol{x}_j, \boldsymbol{x}_l) \tag{4.9}$$

Expresión que puede ser escrita en notación matricial como

$$\mathbf{K}^2 \boldsymbol{a}_i = \lambda_i (N-1) \mathbf{K} \boldsymbol{a}_i \tag{4.10}$$

definiendo la matriz $\mathbf{K}$ en términos de la función de kernel como $k_{nm} = k(\boldsymbol{x}_n, \boldsymbol{x}_m)$.

La solución de (4.10) se puede encontrar resolviendo el problema de autovalores

$$\mathbf{K} \boldsymbol{a}_i = \lambda_i (N-1) \boldsymbol{a}_i \qquad \text{para } i = 1, \ldots, M \tag{4.11}$$

Es importante destacar que la soluciones de (4.11) difieren sólo de las soluciones de (4.10) en los autovectores de $\mathbf{K}$ a los que les corresponde al autovalor 0, lo que no afecta a la proyección de componentes principales.

Al igual que en el caso del PCA, generalmente sólo es necesario obtener aquellos $\boldsymbol{a}_i$ a los que les corresponden los autovalores de mayor valor; si bien resolviendo (4.11) se pueden obtener los M vectores de coeficientes $\boldsymbol{a}_i$ necesarios para reconstruir los M autovectores $\boldsymbol{v}_i$ de la matriz de covarianza $\mathbf{C}$. Es decir, como el problema no puede ser tratado fácilmente en un espacio de elevada dimensión —como es el espacio de características $\mathbb{F}$— se busca proyectar los puntos $\Phi(\boldsymbol{x}_n)$ en un subespacio construído a lo largo de las direcciones donde ocurren las mayores variaciones del conjunto de datos $\{\Phi(\boldsymbol{x}_n)\}$. Para ello sólo es necesario calcular los R autovectores $\boldsymbol{v}_i$ de mayor autovalor λ_i de $\mathbf{C}$. Esto a su vez se traduce, según (4.11), en obtener los R —siendo $R \leq M$— autovectores $\boldsymbol{a}_i$ de mayor autovalor $\lambda_i(N-1)$ de $\mathbf{K}$.

Resuelto este problema, se puede obtener el i-ésimo componente principal para un punto $\boldsymbol{x}$ en términos de la función de kernel. Para eso basta con hacer el producto escalar entre dicho punto en el espacio de características $\Phi(\boldsymbol{x})$ y el

autovector v_i correspondiente a dicha componente

$$y_i(\boldsymbol{x}) = \Phi(\boldsymbol{x})^T \boldsymbol{v}_i = \sum_{n=1}^{N} a_{in} \Phi(\boldsymbol{x})^T \Phi(\boldsymbol{x}_n) = \sum_{n=1}^{N} a_{in} k(\boldsymbol{x}, \boldsymbol{x}_n) \tag{4.12}$$

Así, suponiendo que $\mathbf{A}$ sea la matriz de $N \times R$ cuyas columnas son los R autovectores $\boldsymbol{a}_i$ de $\mathbf{K}$ de mayor autovalor $\lambda_i(N-1)$, la proyección de cada punto $\boldsymbol{x}_n$ en el subespacio generado por el KPCA se puede escribir matricialmente como

$$\boldsymbol{y}_n = \mathbf{A}^T \boldsymbol{k}_n \tag{4.13}$$

Es importante destacar que la dimensión del espacio de entrada del conjunto de datos $\{\boldsymbol{x}_n\}$ es D, por lo que mediante PCA se podrían obtener hasta D componentes principales. Sin embargo la dimensionalidad del espacio de características $\mathbb{F}$ puede ser mucho mayor que D, por lo que se puede encontrar un número de componentes principales no lineales mayor que D.

4.2. Condición de normalidad de los autovectores

Generalmente, cuando se resuelve un problema de autovalores no sólo se exige que los autovectores sean ortogonales, sino también que estén normalizados. Es decir, que $\boldsymbol{v}_i^T \boldsymbol{v}_i = 1, \forall i$. Puesto que los autovectores de la matriz de covarianza del conjunto de datos definen directamente el subespacio de componentes principales en el PCA, está garantizado que los vectores que describen dicho subespacio sean normales. Sin embargo en el KPCA, el imponer la normalidad en el problema de autovalores (4.11) en la página anterior no afecta directamente a los autovectores $\boldsymbol{v}_i$ de la matriz de covarianza $\mathbf{C}$. Sólo se estaría imponiendo la normalidad a los vectores de coeficientes $\boldsymbol{a}_i$. Sabiendo esto y observnado (4.6) en la página 62 resulta obvio que, en general, no se puede decir nada acerca de la norma de los autovectores $\boldsymbol{v}_i$.

Lo que sí se puede hacer es forzar que el producto escalar de cada autovector $\boldsymbol{v}_i$ por sí mismo sea la unidad, para posteriormente desarrollar dicha expresión

mediante el uso de (4.6) y (4.11)

$$\boldsymbol{v}_i^T \boldsymbol{v}_i = \sum_{n=1}^{N} \sum_{m=1}^{N} a_{in} a_{im} \Phi(\boldsymbol{x_n})^T \Phi(\boldsymbol{x_m}) = \boldsymbol{a}_i^T \mathbf{K} \boldsymbol{a}_i = \lambda_i (N-1) \boldsymbol{a}_i^T \boldsymbol{a}_i = 1 \tag{4.14}$$

Esto permite fijar una condición sobre la norma de los vectores de coeficientes $\boldsymbol{a}_i$ con el fin de garantizar la normalidad de los autovectores $\boldsymbol{v}_i$.

$$\lambda_i (N-1) \boldsymbol{a}_i^T \boldsymbol{a}_i = 1 \Rightarrow \boldsymbol{a}_i^T \boldsymbol{a}_i = \frac{1}{\lambda_i (N-1)} \Rightarrow \|\boldsymbol{a}_i\| = \frac{1}{\sqrt{\lambda_i (N-1)}} \tag{4.15}$$

4.3. Consideraciones acerca de la función de kernel

El procedimiento descrito para el KPCA funciona sobre la idea de proyectar el conjunto de puntos a analizar en un espacio de características mediante una transformación no lineal $\Phi(x) : \mathbb{R}^D \longrightarrow \mathbb{F}$. Puesto que dicha transformación no se conoce de forma explícita, todas las ecuaciones se manipulan para ponerlas en términos de la función de kernel $k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \Phi(\boldsymbol{x}_n)^T \Phi(\boldsymbol{x}_m)$ o de la matriz de kernel $\mathbf{K}$ cuyos elementos se definen como $k_{nm} = k(\boldsymbol{x}_n, \boldsymbol{x}_m)$. Pero incluso así, sigue siendo necesario conocer $\Phi(x)$ para evaluar la función de kernel.

La solución a este problema se la conoce por el término en inglés de *kernel trick*. Éste usa el hecho de que para cualquier función de kernel semidefinida positiva existe un espacio donde dicha función puede expresarse como un producto escalar (Aizerman *et al.*, 1964; Hofmann *et al.*, 2008). Es decir, que habiendo definido la función $k(\boldsymbol{x}_n, \boldsymbol{x}_m)$ en términos de los puntos en el espacio de entrada para que sea semidefinida positiva, se puede estar seguro de que existe algún espacio en el que dicha función se puede escribir como

$$k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \Phi(\boldsymbol{x}_n)^T \Phi(\boldsymbol{x}_m) \tag{4.16}$$

suponiendo que $\Phi(x)$ sea la transformación que mapea los puntos del espacio de entrada en el nuevo espacio, al que se denomina espacio de características. Por

tanto, escogiendo la función de kernel se está seleccionando implícitamente la transformación $\Phi(x)$ y el espacio de características sobre el que se va a realizar el análisis de los datos. Además, el que la función de kernel se pueda escribir como un producto escalar en el espacio de características implica que **K** es una matriz de Gram, lo que a su vez significa que es una matriz simétrica semidefinida positiva.

En función de lo comentado hasta el momento, cualquier función de similaridad[1] semidefinida positiva es una función de kernel válida. Por lo que la elección de una función concreta depende fundamentalmente de la naturaleza del problema. Aun así existen alguna funciones de kernel que son de uso común, por ejemplo la función de kernel gaussiano

$$k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \exp(-\|\boldsymbol{x}_n - \boldsymbol{x}_m\|^2/c) \tag{4.17}$$

donde c es un parámetro positivo constante. Esta clase de función de kernel es de uso común, puesto que al ser del tipo función de base radial (RBF)[2] está garantizada su invarianza frente a transformaciones de similaridad y el generar siempre una matriz **K** definida positiva (Hofmann *et al.*, 2008).

Otra función de kernel de uso muy común es la polinomial

$$k(\boldsymbol{x}_n, \boldsymbol{x}_m) = (\boldsymbol{x}_n^T \boldsymbol{x}_m + c)^d \tag{4.18}$$

donde c y d son constantes —con $d \in \mathbb{N}$—. Este tipo de funciones capturan la direccionalidad de los datos, lo que puede ser especialmente interesante en el encaje de correspondencias. Sin embargo, el producto escalar no es invariante frente a cambios de escala, asunto que debe ser resuelto en función del tipo de problema en el que se vaya a utilizar.

[1] Una función de similaridad es una función escalar tal que al ser aplicada sobre una pareja de datos devuelve una valor que indica cómo de parecidos son dichos datos.

[2] Una función $k(x, y)$ es una RBF si y sólo si $k(x, y) = k(\|x - y\|)$.

Por otro lado la función de kernel triangular

$$k(\boldsymbol{x}_n, \boldsymbol{x}_m) = -\|\boldsymbol{x}_n - \boldsymbol{x}_m\|^d \qquad \text{para } 0 < d < 2 \tag{4.19}$$

es invariante a transformaciones afines, excepto en los cambios de escala (Sahbi, 2005). Aun así presenta una curiosa propiedad denominada invarianza «en la forma», tal que dado un factor de escala $\gamma > 0$ esta propiedad puede describirse como

$$k(\gamma \boldsymbol{x}_n, \gamma \boldsymbol{x}_m) = \gamma k(\boldsymbol{x}_n, \boldsymbol{x}_m) \tag{4.20}$$

Elegir una función de kernel adecuada es una de los principales problemas de los métodos basados en kernel. Por eso el desarrollo de técnicas que permitan el aprendizaje del mejor kernel para una tarea dada ha atraído mucho la atención de los investigadores en estos últimos años (Daoqiang *et al.*, 2006).

4.4. Análisis de conjuntos de datos no centrados

Hasta el momento se ha supuesto que el conjunto de datos proyectados en el espacio de características $\{\Phi(\boldsymbol{x}_n)\}$ tiene media cero. Sin embargo esto no tiene por qué ser cierto. Si se trabajara con el PCA, bastaría con sustraer la media muestral al conjunto de datos. Sin embargo en KPCA esto no es posible puesto que se evita el trabajar directamente en el espacio de características $\mathbb{F}$. En su lugar se intenta expresar dicha sustracción en términos de la función kernel.

4.4.1. Centrado en el espacio de características

Suponiendo que fuera posible sustraer la media muestral $\frac{1}{N}\sum_{n=1}^{N}\Phi(\boldsymbol{x}_n)$ de los datos en el espacio de características, cada punto centrado y proyectado $\tilde{\Phi}(\boldsymbol{x}_n)$ vendría dado por

$$\tilde{\Phi}(\boldsymbol{x}_n) = \Phi(\boldsymbol{x}_n) - \frac{1}{N}\sum_{s=1}^{N}\Phi(\boldsymbol{x}_s) \tag{4.21}$$

y cada elemento de la matriz $\mathbf{K}$ centrada, denotada por $\tilde{\mathbf{K}}$, vendría dado por

$$\begin{aligned}
\tilde{K}_{nm} &= \tilde{\Phi}(\boldsymbol{x}_n)^T\tilde{\Phi}(\boldsymbol{x}_m) \\
&= \left(\Phi(\boldsymbol{x}_n)^T - \frac{1}{N}\sum_{s=1}^{N}\Phi(\boldsymbol{x}_s)^T\right)\left(\Phi(\boldsymbol{x}_m) - \frac{1}{N}\sum_{t=1}^{N}\Phi(\boldsymbol{x}_t)\right) \\
&= \Phi(\boldsymbol{x}_n)^T\Phi(\boldsymbol{x}_m) - \frac{1}{N}\sum_{s=1}^{N}\Phi(\boldsymbol{x}_n)^T\Phi(\boldsymbol{x}_s) \\
&\quad - \frac{1}{N}\sum_{s=1}^{N}\Phi(\boldsymbol{x}_s)^T\Phi(\boldsymbol{x}_m) + \frac{1}{N^2}\sum_{s=1}^{N}\sum_{t=1}^{N}\Phi(\boldsymbol{x}_s)^T\Phi(\boldsymbol{x}_t) \\
&= k(\boldsymbol{x}_n, \boldsymbol{x}_m) - \frac{1}{N}\sum_{s=1}^{N}k(\boldsymbol{x}_n, \boldsymbol{x}_s) \\
&\quad - \frac{1}{N}\sum_{s=1}^{N}k(\boldsymbol{x}_s, \boldsymbol{x}_m) + \frac{1}{N^2}\sum_{s=1}^{N}\sum_{t=1}^{N}k(\boldsymbol{x}_s, \boldsymbol{x}_t)
\end{aligned} \tag{4.22}$$

que puede ser expresado en notación matricial como

$$\tilde{\mathbf{K}} = \mathbf{K} - \frac{1}{N}\mathbf{1}\mathbf{1}^T\mathbf{K} - \mathbf{K}\frac{1}{N}\mathbf{1}\mathbf{1}^T + \frac{1}{N}\mathbf{1}\mathbf{1}^T\mathbf{K}\frac{1}{N}\mathbf{1}\mathbf{1}^T \Rightarrow \tilde{\mathbf{K}} = \mathbf{HKH} \tag{4.23}$$

donde $\mathbf{H} = \mathbf{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^T$.

De esta manera, el procedimiento completo para el KPCA de un conjunto de datos cualquiera se resume en el algoritmo 4.1.

4.4.2. Centrado por clases en el espacio de características

Hasta el momento se ha visto cómo centrar el conjunto de datos en el espacio de características sustrayendo la media muestral de todos los puntos. Sin embargo, en ocasiones los puntos pueden haber sido clasificados, siendo mucho más interesante centrarlos respecto a los otros miembros de la misma clase. Esto, por ejemplo, puede ser utilizado en el encaje de puntos entre objetos articulados en diferentes posturas (Wang, 2007, cap. 3 o Wang y Hancock, 2006).

Suponiendo que cada punto del conjunto de datos $\{\boldsymbol{x}_n\}$ pertenece a una de

Algoritmo 4.1. KPCA de un conjunto de datos no centrado.

1. Calcular la matriz $\mathbf{K}$ a partir de la función de kernel $k(\boldsymbol{x}_n, \boldsymbol{x}_m)$ seleccionada.
2. Centrar el conjunto de datos en el espacio de características utilizando (4.23)
$$\tilde{\mathbf{K}} = \mathbf{HKH}$$
3. Resolver el problema de autovalores (4.11) en la página 63
$$\tilde{\mathbf{K}}\boldsymbol{a}_i = \lambda_i(N-1)\boldsymbol{a}_i \qquad \text{para } i = 1, \ldots, M$$
para los R autovectores $\boldsymbol{a}_i$ de mayor autovalor λ_i.
4. Reescalar los vectores de coeficientes $\boldsymbol{a}_i$ teniendo en cuenta la condición de normalidad (4.15) en la página 65
$$\|\boldsymbol{a}_i\| = \frac{1}{\sqrt{\lambda_i(N-1)}}$$
5. Proyectar el conjunto de datos en el subespacio de componentes principales mediante la expresión (4.13) en la página 64
$$\boldsymbol{y}_n = \mathbf{A}^T\tilde{\boldsymbol{k}}_n \qquad \text{donde } \mathbf{A} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_R]$$

L clases. Sea $\mathbf{P}$ una matriz de $N \times L$, tal que el elemento p_{nl} contenga la probabilidad de que el n-ésimo punto pertenezca a la clase l y que $\sum_{l=1}^{L} p_{nl} = 1, \forall n$. Entonces la media en el espacio de características de los puntos de la clase l, denotada por $\boldsymbol{\mu}_l$, vendría dada por

$$\boldsymbol{\mu}_l = \frac{1}{\sum_{m=1}^{N} p_{ml}} \sum_{n=1}^{N} \Phi(\boldsymbol{x}_n) p_{nl} \tag{4.24}$$

Puesto que cada punto $\Phi(\boldsymbol{x}_n)$ tiene una probabilidad determinada de pertenecer a cada una de las clases, se puede estimar la media a sustraer a cada uno como el promedio de $\boldsymbol{\mu}_l$ para todos los l, ponderado por la probabilidad de que el punto pertenezca a cada clase concreta

$$\tilde{\Phi}(\boldsymbol{x}_n) = \Phi(\boldsymbol{x}_n) - \sum_{l=1}^{L} \boldsymbol{\mu}_l p_{nl} \tag{4.25}$$

donde $\tilde{\Phi}(\boldsymbol{x}_n)$ denota al punto $\boldsymbol{x}_n$ centrado y proyectado en el espacio de características. Al igual que antes, cada elemento de la matriz $\tilde{\mathbf{K}}$ vendría dado por

$$\begin{aligned}
\tilde{K}_{nm} &= \tilde{\Phi}(\boldsymbol{x}_n)^T \tilde{\Phi}(\boldsymbol{x}_m) \\
&= \left(\Phi(\boldsymbol{x}_n)^T - \sum_{l=1}^{L} \frac{p_{nl}}{\sum_{i=1}^{N} p_{il}} \sum_{s=1}^{N} \Phi(\boldsymbol{x}_s)^T p_{sl} \right) \\
&\quad \left(\Phi(\boldsymbol{x}_m) - \sum_{r=1}^{L} \frac{p_{mr}}{\sum_{j=1}^{N} p_{jr}} \sum_{t=1}^{N} \Phi(\boldsymbol{x}_t) p_{tr} \right) \\
&= \Phi(\boldsymbol{x}_n)^T \Phi(\boldsymbol{x}_m) \\
&\quad - \sum_{r=1}^{L} \frac{p_{mr}}{\sum_{j=1}^{N} p_{jr}} \sum_{t=1}^{N} \Phi(\boldsymbol{x}_n)^T \Phi(\boldsymbol{x}_t) p_{tr} \\
&\quad - \sum_{l=1}^{L} \frac{p_{nl}}{\sum_{i=1}^{N} p_{il}} \sum_{s=1}^{N} \Phi(\boldsymbol{x}_s)^T \Phi(\boldsymbol{x}_m) p_{sl} \\
&\quad + \sum_{l=1}^{L} \sum_{r=1}^{L} \frac{p_{nl} p_{mr}}{\sum_{i=1}^{N} p_{il} \sum_{j=1}^{N} p_{jr}} \sum_{s=1}^{N} \sum_{t=1}^{N} \Phi(\boldsymbol{x}_s)^T \Phi(\boldsymbol{x}_t) p_{sl} p_{tr}
\end{aligned}$$

$$
\begin{aligned}
&= k(\boldsymbol{x}_n, \boldsymbol{x}_m) \\
&\quad - \sum_{r=1}^{L} \frac{p_{mr}}{\sum_{j=1}^{N} p_{jr}} \sum_{t=1}^{N} k(\boldsymbol{x}_n, \boldsymbol{x}_t) p_{tr} \\
&\quad - \sum_{l=1}^{L} \frac{p_{nl}}{\sum_{i=1}^{N} p_{il}} \sum_{s=1}^{N} k(\boldsymbol{x}_s, \boldsymbol{x}_m) p_{sl} \\
&\quad + \sum_{l=1}^{L} \sum_{r=1}^{L} \frac{p_{nl} p_{mr}}{\sum_{i=1}^{N} p_{il} \sum_{j=1}^{N} p_{jr}} \sum_{s=1}^{N} \sum_{t=1}^{N} k(\boldsymbol{x}_s, \boldsymbol{x}_t) p_{sl} p_{tr}
\end{aligned} \tag{4.26}
$$

que puede ser expresado en notación matricial como

$$
\tilde{\mathbf{K}} = \mathbf{K} - \mathbf{KW} - \mathbf{WK} - \mathbf{WKW} \Rightarrow \tilde{\mathbf{K}} = (\mathbf{I} - \mathbf{W})\,\mathbf{K}\,(\mathbf{I} - \mathbf{W}) \tag{4.27}
$$

donde $\mathbf{W} = \mathbf{PDP}^T$ y $\mathbf{D}$ es una matriz diagonal de elementos $d_{ll} = \frac{1}{\sum_{n=1}^{N} p_{nl}}$.

El procedimiento completo para el KPCA centrado por clases de un conjunto de datos cualquiera se resumen en el algoritmo 4.2.

4.5. Estimación de la preimagen

La transformación que hace el PCA a los datos es invertible. Es decir, dado un punto $\boldsymbol{y}$ del subespacio de componentes principales de dimensión R, es posible obtener un punto $\boldsymbol{x}$ en el espacio de dimensión D del conjunto de datos original.

Sin embargo, en general, en el KPCA esto no es posible. Como se esquematiza en la figura 4.4, la transformación no lineal $\Phi(x)$ mapea los puntos $\boldsymbol{x}_n$ del espacio de dimensión D en una variedad, también de dimensión D, en el espacio de características $\mathbb{F}$ de dimensión M —con $M \geq D$—. El punto $\boldsymbol{x}_n$ es conocido como la *preimagen* del correspondiente punto $\Phi(\boldsymbol{x}_n)$. Por el contrario $P\Phi(\boldsymbol{x}_n)$, la proyección de $\Phi(\boldsymbol{x}_n)$ en el subespacio de componentes principales pero en coordenadas del espacio de características[3], no tiene por qué estar dentro de la variedad de

[3]La proyección de $\Phi(\boldsymbol{x}_n)$ en el subespacio de componentes principales se expresa respecto a la base que define dicho subespacio. Sin embargo, para la estimación de la preimagen es más interesante expresarla en coordenadas del espacio de características, lo que se denota por $P\Phi(\boldsymbol{x}_n)$.

Algoritmo 4.2. KPCA de un conjunto de datos con centrado por clases.

1. Calcular la matriz $\mathbf{K}$ a partir de la función de kernel $k(\boldsymbol{x}_n, \boldsymbol{x}_m)$ seleccionada.

2. Centrar el conjunto de datos en el espacio de características utilizando (4.27)
$$\tilde{\mathbf{K}} = (\mathbf{I} - \mathbf{W})\,\mathbf{K}\,(\mathbf{I} - \mathbf{W})$$
donde $\mathbf{W} = \mathbf{P}\mathbf{D}\mathbf{P}^T$ y $\mathbf{D}$ es una matriz diagonal de elementos $d_{ll} = \frac{1}{\sum_{n=1}^{N} p_{nl}}$.

3. Resolver el problema de autovalores (4.11) en la página 63
$$\tilde{\mathbf{K}}\boldsymbol{a}_i = \lambda_i(N-1)\boldsymbol{a}_i \qquad \text{para } i = 1, \ldots, M$$
para los R autovectores $\boldsymbol{a}_i$ de mayor autovalor λ_i.

4. Reescalar los vectores de coeficientes $\boldsymbol{a}_i$ teniendo en cuenta la condición de normalidad (4.15) en la página 65
$$\|\boldsymbol{a}_i\| = \frac{1}{\sqrt{\lambda_i(N-1)}}$$

5. Proyectar el conjunto de datos en el subespacio de componentes principales mediante la expresión (4.13) en la página 64
$$\boldsymbol{y}_n = \mathbf{A}^T\tilde{\boldsymbol{k}}_n \qquad \text{donde } \mathbf{A} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_R]$$

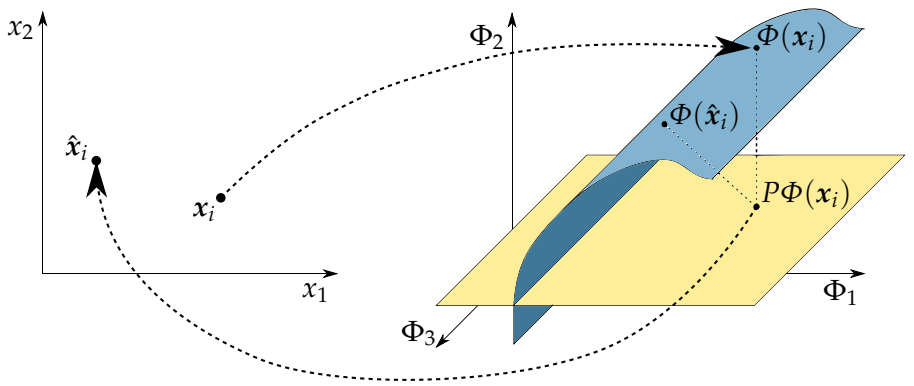


Figura 4.4. El punto $P\Phi(\boldsymbol{x}_i)$, resultante de mapear y proyectar el punto $\boldsymbol{x}_i$, no pertenece a la variedad definida por el espacio de entrada en el espacio de característica $\mathbb{F}$, por lo que no tiene preimagen. Se pretende estimar una hallando un punto $\hat{\boldsymbol{x}}_i$ tal que su proyección $\Phi(\hat{\boldsymbol{x}}_i)$ en $\mathbb{F}$ esté lo más cerca posible de $P\Phi(\boldsymbol{x}_i)$.

dimensión D de los datos. Por tanto, no tiene porqué tener una preimagen en el espacio de entrada. Sin embargo, la obtención de una preimagen, aunque sea aproximada, puede ser de interés para, por ejemplo, obtener algo de información acerca de lo que está ocurriendo en el espacio de características. Por eso se han propuesto diferentes técnicas para intentar encontrar preimágenes aproximadas (Bakir *et al.*, 2004).

El objetivo, por tanto, es estimar una preimagen $\hat{\boldsymbol{x}}$ del punto $\boldsymbol{y}$ del subespacio de componentes principales del espacio de características. Para eso es necesario deshacer la transformación lineal que proyecta los puntos del espacio de características $\mathbb{F}$ en el subespacio de componentes principales. Puesto que dicho subespacio está definido por los autovectores $\{\boldsymbol{v}_i\}$, para cualquier punto $\boldsymbol{y}$ es posible encontrar su proyección en el espacio de características —denotada por $P\Phi(\boldsymbol{x})$— mediante la expresión

$$P\Phi(\boldsymbol{x}) = \sum_{i=1}^{R} y_i \boldsymbol{v}_i \tag{4.28}$$

donde y_i es la i-ésima componente del vector $\boldsymbol{y}$. Puesto que la preimagen no tiene por qué existir, Mika *et al.* (1999) proponen hallar la solución resolviendo el problema de optimización para $\hat{\boldsymbol{x}}$

$$\hat{\boldsymbol{x}} = \arg\min_{\hat{\boldsymbol{x}}} \|\Phi(\hat{\boldsymbol{x}}) - P\Phi(\boldsymbol{x})\|^2 \tag{4.29}$$

reemplazando los términos independientes de $\hat{\boldsymbol{x}}$ por Ω

$$\hat{\boldsymbol{x}} = \arg\min_{\hat{\boldsymbol{x}}} \|\Phi(\hat{\boldsymbol{x}})\|^2 - 2(\Phi(\hat{\boldsymbol{x}}) \cdot P\Phi(\boldsymbol{x})) + \Omega \tag{4.30}$$

Sustituyendo (4.28) y (4.6) en la página 62 en (4.30) se obtiene una expresión en términos de productos escalares, lo que permite introducir la función de kernel y eliminar las referencias a la transformación no lineal $\Phi(x)$

$$\hat{\boldsymbol{x}} = \arg\min_{\hat{\boldsymbol{x}}} k(\hat{\boldsymbol{x}}, \hat{\boldsymbol{x}}) - 2\sum_{i=1}^{R} y_i \sum_{n=1}^{N} a_{in} k(\hat{\boldsymbol{x}}, \boldsymbol{x}_n) + \Omega \tag{4.31}$$

4.5.1. Preimágenes para funciones de kernel gaussianas

En el caso de que la función de kernel sea una RBF el término $k(\hat{\boldsymbol{x}}, \hat{\boldsymbol{x}})$ de (4.31) es constante para cualquier $\hat{\boldsymbol{x}}$, lo que permite que sea englobado en Ω

$$\hat{\boldsymbol{x}} = \arg\max_{\hat{\boldsymbol{x}}} \sum_{n=1}^{N} \gamma_n k(\hat{\boldsymbol{x}}, \boldsymbol{x}_n) + \Omega \tag{4.32}$$

donde $\gamma_n = \sum_{i=1}^{R} y_i a_{in}$. Mika *et al.* (1999) resuelven (4.32) para una función de kernel gaussiano, (4.17) en la página 66, mediante el método del gradiente descendente.

$$\hat{\boldsymbol{x}} = \frac{\sum_{n=1}^{N} \gamma_n \exp(-\|\hat{\boldsymbol{x}} - \boldsymbol{x}_n\|^2/c)\boldsymbol{x}_n}{\sum_{n=1}^{N} \gamma_n \exp(-\|\hat{\boldsymbol{x}} - \boldsymbol{x}_n\|^2/c)} \tag{4.33}$$

Centrado en el espacio de características

La expresión (4.33) es correcta si se supone que los datos quedan centrados al ser mapeados en el espacio de características mediante la transformación no lineal $\Phi(x)$. Si no fuera así, al aplicar el KPCA sería necesario utilizar alguna de las técnicas descritas en 4.4.1 en la página 67. El uso de éstas hace que los autovectores $\boldsymbol{v}_i$, que describen el subespacio de componentes principales, no dependan de los datos mapeados $\Phi(\boldsymbol{x}_n)$ sino de los datos mapeados y centrados $\tilde{\Phi}(\boldsymbol{x}_n)$ en el espacio de características[4]. El resultado es que en (4.29) se comparan la proyección de un punto $\boldsymbol{y}$ del subespacio de componentes principales en el espacio de características —obtenida mediante (4.28)— con la estimación de la preimagen $\hat{\boldsymbol{x}}$ de dicho punto también proyectada en el espacio de características. Sin embargo al primero se le ha restado la media muestral del conjunto de datos proyectados $\{\Phi(\boldsymbol{x}_n)\}$, por la dependencia de los autovectores respecto a los datos mapeados y centrados, mientras que al segundo no. Por tanto (4.28) debe ser reescrita, para el caso de datos mapeados no centrados, como

$$P\Phi(\boldsymbol{x}) = \sum_{i=1}^{R} y_i \boldsymbol{v}_i + \frac{1}{N} \sum_{n=1}^{N} \Phi(\boldsymbol{x}_n) \tag{4.34}$$

Modificando (4.6) en la página 62 para tomar en consideración la dependencia de los autovectores respecto a los datos mapeados y centrados

$$\boldsymbol{v}_i = \sum_{n=1}^{N} a_{in} \tilde{\Phi}(\boldsymbol{x}_n) \tag{4.35}$$

[4]En (4.6) en la página 62 se indica que los autovectores son una combinación lineal de los datos mapeados en el espacio de características $\Phi(\boldsymbol{x}_n)$. Sin embargo, el desarrollo de todas las expresiones del KPCA se basó en la premisa de que dichos datos están centrados, lo permite utilizar la igualdad $\tilde{\Phi}(\boldsymbol{x}_n) = \Phi(\boldsymbol{x}_n)$ según (4.21) en la página 67. Si esto no fuera cierto, todas las expresiones del KPCA deben ser reescritas en función de los datos mapeados y centrados $\tilde{\Phi}(\boldsymbol{x}_n)$ y utilizar lo visto en 4.4.1 en la página 67 para obtener la matriz $\tilde{\mathbf{K}}$ de los datos centrados a partir de la de los datos no centrados $\mathbf{K}$. Para el algoritmo 4.1 ésto ya fue tenido en cuenta

sustituyendo (4.34) y (4.35) en (4.30) y aplicado el método del gradiente descendente —como se ha hecho anteriormente— se obtiene la siguiente expresión para la estimación de la preimagen

$$\hat{\boldsymbol{x}} = \frac{\sum_{n=1}^{N} \tilde{\gamma}_n \exp(-\|\hat{\boldsymbol{x}} - \boldsymbol{x}_n\|^2/c)\boldsymbol{x}_n}{\sum_{n=1}^{N} \tilde{\gamma}_n \exp(-\|\hat{\boldsymbol{x}} - \boldsymbol{x}_n\|^2/c)} \tag{4.36}$$

donde $\tilde{\gamma}_n = \gamma_n + \frac{1}{N}\left(1 - \sum_{m=1}^{N} \gamma_m\right)$ y $\gamma_n = \sum_{i=1}^{R} y_i a_{in}$.

Preimágenes por métodos no iterativos

Si bien la técnica de Mika *et al.* (1999) es la más extendida, no por ello está exenta de problemas. Tanto (4.33) como (4.36) describen métodos iterativos dependientes del valor inicial con el que arranquen. Por ello diversos autores han propuesto técnicas que permiten obtener una aproximación de la preimagen $\hat{\boldsymbol{x}}$ de un punto $\boldsymbol{y}$ en un sólo paso (Kwok y Tsang, 2004; Rathi *et al.*, 2006).

Rathi *et al.* (2006) comienzan observando que para muchas funciones de kernel existe una relación (Schölkopf, 2000) entre la distancia de los puntos en el espacio de entrada $d^2(\boldsymbol{x}_n, \boldsymbol{x}_m)$ y la distancia de los mismos en el espacio de características $D^2(\Phi(\boldsymbol{x}_n), \Phi(\boldsymbol{x}_m))$

$$\begin{aligned} D^2(\Phi(\boldsymbol{x}_n), \Phi(\boldsymbol{x}_m)) &= \|\Phi(\boldsymbol{x}_n) - \Phi(\boldsymbol{x}_m)\|^2 \\ &= k(\boldsymbol{x}_n, \boldsymbol{x}_n) + k(\boldsymbol{x}_m, \boldsymbol{x}_m) - 2k(\boldsymbol{x}_n, \boldsymbol{x}_m) \\ &= k(\boldsymbol{x}_n, \boldsymbol{x}_n) + k(\boldsymbol{x}_m, \boldsymbol{x}_m) - 2k(d^2(\boldsymbol{x}_n, \boldsymbol{x}_m)) \end{aligned} \tag{4.37}$$

donde se ha hecho uso de la igualdad $k(\boldsymbol{x}_n, \boldsymbol{x}_m) = k(d^2(\boldsymbol{x}_n, \boldsymbol{x}_m))$ suponiendo que la función de kernel es una función dependiente de la distancia de los puntos en el espacios de entrada, como ocurre en el caso de la función de kernel gaussiano (4.17) en la página 66 y otras funciones de base radial. Si la función de

kernel fuera invertible

$$\begin{aligned} k(d^2(\boldsymbol{x}_n, \boldsymbol{x}_m)) &= \frac{1}{2}(k(\boldsymbol{x}_n, \boldsymbol{x}_n) + k(\boldsymbol{x}_m, \boldsymbol{x}_m) - D^2(\Phi(\boldsymbol{x}_n), \Phi(\boldsymbol{x}_m))) \Rightarrow \\ d^2(\boldsymbol{x}_n, \boldsymbol{x}_m) &= k^{-1}\left(\frac{1}{2}(k(\boldsymbol{x}_n, \boldsymbol{x}_n) + k(\boldsymbol{x}_m, \boldsymbol{x}_m) - D^2(\Phi(\boldsymbol{x}_n), \Phi(\boldsymbol{x}_m)))\right) \end{aligned} \tag{4.38}$$

Por otro lado, se puede calcular la distancia entre un punto cualquiera del conjunto de datos proyectado en el espacio de características $\Phi(\boldsymbol{x}_n)$ y un punto proyectado desde el subespacio de componentes principales $P\Phi(\boldsymbol{x})$ mediante (4.34)

$$\begin{aligned} D^2(P\Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}_n)) &= \|P\Phi(\boldsymbol{x}) - \Phi(\boldsymbol{x}_n)\|^2 \\ &= \|P\Phi(\boldsymbol{x})\|^2 + \|\Phi(\boldsymbol{x}_n)\|^2 - 2P\Phi(\boldsymbol{x})^T\Phi(\boldsymbol{x}_n) \end{aligned} \tag{4.39}$$

que, después de sustituir (4.34), (4.35), (4.12)[5] y (4.23) en la página 68 en (4.39) y hacer algunas simplificaciones, se puede escribir en términos de la función de kernel

$$\begin{aligned} D^2(P\Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}_n)) = &\left(\boldsymbol{k}_x + \frac{1}{N}\mathbf{K}\mathbf{1} - 2\boldsymbol{k}_n\right)^T \mathbf{H}\mathbf{A}\mathbf{A}^T\mathbf{H}\left(\boldsymbol{k}_x - \frac{1}{N}\mathbf{K}\mathbf{1}\right) \\ &+ \frac{1}{N^2}\mathbf{1}^T\mathbf{K}\mathbf{1} + k_{nn} - \frac{2}{N}\mathbf{1}^T\boldsymbol{k}_n \end{aligned} \tag{4.40}$$

donde $\boldsymbol{k}_x = [k(\boldsymbol{x}, \boldsymbol{x}_1), \ldots, k(\boldsymbol{x}, \boldsymbol{x}_N)]^T$ y $\mathbf{H} = \mathbf{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^T$.

En el caso de la función de kernel gaussiano (4.17) en la página 66 la expresión $\|\boldsymbol{x}_n - \boldsymbol{x}_m\|^2$ es una medida de la distancia en el espacio de entrada, lo que permite reescribir (4.36) como

$$\hat{\boldsymbol{x}} = \frac{\sum_{n=1}^{N} \tilde{\gamma}_n \exp(-d^2(\hat{\boldsymbol{x}}, \boldsymbol{x}_n)/c)\boldsymbol{x}_n}{\sum_{n=1}^{N} \tilde{\gamma}_n \exp(-d^2(\hat{\boldsymbol{x}}, \boldsymbol{x}_n)/c)} \tag{4.41}$$

[5]La expresión (4.12) en la página 64 se obtuvo suponiendo que el conjunto de datos mapeados en el espacio de características $\{\Phi(\boldsymbol{x}_n)\}$ está centrado. Si esto no fuera cierto, la expresión (4.12) debe ser reescrita para depender de la función de kernel de los datos centrados $\tilde{k}(\boldsymbol{x}_n, \boldsymbol{x}_m)$ y no de la de los datos no centrados $k(\boldsymbol{x}_n, \boldsymbol{x}_m)$.

donde resulta sencillo sustituir (4.38)

$$\hat{\boldsymbol{x}} = \frac{\sum_{n=1}^{N} \tilde{\gamma}_n \left(1 - \frac{1}{2} D^2(\Phi(\hat{\boldsymbol{x}}), \Phi(\boldsymbol{x}_n)) \boldsymbol{x}_n\right)}{\sum_{n=1}^{N} \tilde{\gamma}_n \left(1 - \frac{1}{2} D^2(\Phi(\hat{\boldsymbol{x}}), \Phi(\boldsymbol{x}_n))\right)} \tag{4.42}$$

Finalmente se sustituye en (4.42) la aproximación $P\Phi(\boldsymbol{x}) \approx \Phi(\hat{\boldsymbol{x}})$, puesto que esa sería la solución óptima de (4.29)

$$\hat{\boldsymbol{x}} \approx \frac{\sum_{n=1}^{N} \tilde{\gamma}_n \left(1 - \frac{1}{2} D^2(P\Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}_n)) \boldsymbol{x}_n\right)}{\sum_{n=1}^{N} \tilde{\gamma}_n \left(1 - \frac{1}{2} D^2(P\Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}_n))\right)} \tag{4.43}$$

donde $\tilde{\gamma}_n = \gamma_n + \frac{1}{N}\left(1 - \sum_{m=1}^{N} \gamma_m\right)$ y $\gamma_n = \sum_{i=1}^{R} y_i a_{in}$.

El procedimiento para calcular la preimagen de un punto $\boldsymbol{y}$ en el subespacio de componentes principales se resume en el algoritmo 4.3.

Preimágenes por métodos no iterativos y con centrado por clases en el espacio de características

Tanto (4.36) como (4.43) permiten obtener una aproximación de la preimagen $\hat{\boldsymbol{x}}$ de un punto $\boldsymbol{y}$ cuando la transformación no lineal $\Phi(x)$ no deja centrados los datos en el espacio de características. Sin embargo no son adecuadas cuando los puntos han sido clasificados, por lo que interesa centrarlos respecto a los otros miembros de la misma clase (véase 4.4.2 en la página 68).

Para obtener una expresión adecuada para esas circunstancias basta con utilizar las técnicas de Mika *et al.* (1999) y Rathi *et al.* (2006), descritas hasta el momento en este apartado, pero recordando que ahora los puntos proyectados y centrados en el espacio de características $\tilde{\Phi}(\boldsymbol{x}_n)$ y la matriz $\tilde{\mathbf{K}}$ se deben obtener mediante (4.25) en la página 70 y (4.27) en la página 71 respectivamente. Además, también es necesario reescribir (4.28) en la página 73 como

$$P\Phi(\boldsymbol{x}) = \sum_{i=1}^{R} y_i \boldsymbol{v}_i + \sum_{l=1}^{L} \boldsymbol{\mu}_l p_{xl} \tag{4.44}$$

Algoritmo 4.3. Estimación de la preimagen de un punto proyectado en el subespacio de componentes principales.

1. Calcular la proyección del punto $\boldsymbol{y}$ en el subespacio de características utilizando (4.34) en la página 75

$$P\Phi(\boldsymbol{x}) = \sum_{i=1}^{R} y_i \boldsymbol{v}_i + \frac{1}{N}\sum_{n=1}^{N} \Phi(\boldsymbol{x}_n)$$

2. Calcular mediante (4.40)

$$D^2(P\Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}_n)) = \left(\boldsymbol{k}_x + \frac{1}{N}\mathbf{K1} - 2\boldsymbol{k}_n\right)^T \mathbf{HAA}^T\mathbf{H}\left(\boldsymbol{k}_x - \frac{1}{N}\mathbf{K1}\right) + \frac{1}{N^2}\mathbf{1}^T\mathbf{K1} + k_{nn} - \frac{2}{N}\mathbf{1}^T\boldsymbol{k}_n$$

   donde $\boldsymbol{k}_x = [k(\boldsymbol{x}, \boldsymbol{x}_1), \ldots, k(\boldsymbol{x}, \boldsymbol{x}_N)]^T$ y $\mathbf{H} = \mathbf{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^T$, las distancias entre el punto $\boldsymbol{y}$ proyectado $P\Phi(\boldsymbol{x})$ y cada punto del conjunto de datos en el espacio de características $\{\Phi(\boldsymbol{x}_n)\}$

3. Obtener la aproximación de la preimagen utilizando (4.43)

$$\hat{\boldsymbol{x}} \approx \frac{\sum_{n=1}^{N} \tilde{\gamma}_n \left(1 - \frac{1}{2}D^2(P\Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}_n))\boldsymbol{x}_n\right)}{\sum_{n=1}^{N} \tilde{\gamma}_n \left(1 - \frac{1}{2}D^2(P\Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}_n))\right)}$$

   donde $\tilde{\gamma}_n = \gamma_n + \frac{1}{N}\left(1 - \sum_{m=1}^{N}\gamma_m\right)$ y $\gamma_n = \sum_{i=1}^{R} y_i a_{in}$.

donde p_{xl} es la l-ésima componente de un vector $\boldsymbol{p}_x$ de dimensión L, de manera que el l-ésimo elemento de dicho vector contiene la probabilidad de que el punto $P\Phi(\boldsymbol{x})$ pertenezca a la clase l. Hechas las sustituciones y después de las manipulaciones matemáticas comentadas anteriormente, se obtiene la expresión para el cálculo en un solo paso de la preimagen $\hat{\boldsymbol{x}}$ de un punto $P\Phi(\boldsymbol{x})$

$$\hat{\boldsymbol{x}} \approx \frac{\sum_{n=1}^{N} \tilde{\gamma}_n \left(1 - \frac{1}{2} D^2(P\Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}_n)) \boldsymbol{x}_n\right)}{\sum_{n=1}^{N} \tilde{\gamma}_n \left(1 - \frac{1}{2} D^2(P\Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}_n))\right)} \tag{4.45}$$

donde $\tilde{\gamma}_n = \gamma_n + \sum_{l=1}^{L} \frac{p_{nl}}{\sum_{s=1}^{N} p_{sl}} \left(p_{xl} - \sum_{m=1}^{N} \gamma_m p_{ml}\right)$ y $\gamma_n = \sum_{i=1}^{R} y_i a_{in}$. Igualmente se obtiene la expresión de la distancia $D^2(P\Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}_n))$

$$\begin{aligned} D^2(\Phi(\boldsymbol{x}_n), P\Phi(\boldsymbol{x})) = {} & (\boldsymbol{k}_x + \mathbf{K}\boldsymbol{w}_x - 2\boldsymbol{k}_n)^T (\mathbf{I} - \mathbf{W}) \mathbf{A}\mathbf{A}^T (\mathbf{I} - \mathbf{W}) (\boldsymbol{k}_x - \mathbf{K}\boldsymbol{w}_x) \\ & + \boldsymbol{w}_x^T \mathbf{K} \boldsymbol{w}_x + k_{nn} - 2\boldsymbol{w}_x^T \boldsymbol{k}_n \end{aligned} \tag{4.46}$$

donde $\boldsymbol{k}_x = [k(\boldsymbol{x}, \boldsymbol{x}_1), \ldots, k(\boldsymbol{x}, \boldsymbol{x}_N)]^T$, $\mathbf{W} = \mathbf{P}\mathbf{D}\mathbf{P}^T$, $\boldsymbol{w}_x = \mathbf{P}\mathbf{D}\boldsymbol{p}_x$ y $\mathbf{D}$ es una matriz diagonal de elementos $d_{ll} = \frac{1}{\sum_{n=1}^{N} p_{nl}}$.

4.5.2. Preimágenes para funciones de kernel polinomiales

Aunque la función de kernel gaussiano (4.17) en la página 66 se utiliza con mucha frecuencia, en ocasiones puede ser interesante elegir otra. En el caso de escoger una función de kernel polinomial (4.18) en la página 66 es perfectamente posible repetir el procedimiento descrito por Mika *et al.* (1999), para obtener una expresión que permita calcular la preimagen $\hat{\boldsymbol{x}}$ de un punto $\boldsymbol{y}$ de forma iterativa

$$\hat{\boldsymbol{x}} = \sum_{n=1}^{N} \tilde{\gamma}_n \left(\frac{\hat{\boldsymbol{x}}^T \boldsymbol{x}_n}{\hat{\boldsymbol{x}}^T \hat{\boldsymbol{x}}}\right)^{d-1} \tag{4.47}$$

donde

$$\tilde{\gamma}_n = \begin{cases} \gamma_n + \frac{1}{N}\left(1 - \sum_{m=1}^{N} \gamma_m\right) & \text{si centrado uniforme} \\ \gamma_n + \sum_{l=1}^{L} \frac{p_{nl}}{\sum_{s=1}^{N} p_{sl}} \left(p_{xl} - \sum_{m=1}^{N} \gamma_m p_{ml}\right) & \text{si centrado por clases} \end{cases}$$

y $\gamma_n = \sum_{i=1}^{R} y_i a_{in}$. Igualmente es posible seguir el procedimiento descrito por Rathi *et al.* (2006) —incluida la aproximación $P\Phi(\boldsymbol{x}) \approx \Phi(\hat{\boldsymbol{x}})$— para después de algunas simplificaciones obtener una expresión para el cálculo de la preimagen en un sólo paso

$$\hat{\boldsymbol{x}} \approx \sum_{n=1}^{N} \tilde{\gamma}_n \left(\frac{\|P\Phi(\boldsymbol{x})\|^2 + k(\boldsymbol{x}_n, \boldsymbol{x}_n) - D^2(P\Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}_n))}{2\|P\Phi(\boldsymbol{x})\|^2} \right)^{\frac{d-1}{d}} \boldsymbol{x}_n \tag{4.48}$$

donde

$$\|P\Phi(\boldsymbol{x})\|^2 = \left(\boldsymbol{k}_x + \frac{1}{N}\mathbf{K1}\right)^T \mathbf{HAA}^T\mathbf{H}\left(\boldsymbol{k}_x - \frac{1}{N}\mathbf{K1}\right) + \frac{1}{N^2}\mathbf{1}^T\mathbf{K1}$$

si el centrado es uniforme o

$$\|P\Phi(\boldsymbol{x})\|^2 = \left(\boldsymbol{k}_x + \mathbf{K}\boldsymbol{w}_x\right)^T (\mathbf{I} - \mathbf{W})\,\mathbf{AA}^T\,(\mathbf{I} - \mathbf{W})\,(\boldsymbol{k}_x - \mathbf{K}\boldsymbol{w}_x) + \boldsymbol{w}_x^T\mathbf{K}\boldsymbol{w}_x$$

si el centrado es por clases.

En general, usando esta misma metodología sería posible obtener expresiones para el cálculo de preimágenes para cualquier tipo de función de kernel invertible y cualquier tipo de métrica en el espacio de entrada.

4.6. Métricas entre subespacios de componentes principales

Como ya se ha comentado, el espacio de características $\mathbb{F}$ sobre el que se mapean los datos se define implícitamente por la elección de la función de kernel. Sin embargo, el subespacio de componentes principales se define por la varia-

bilidad de los propios datos en el espacio de características. Así que diferentes conjunto de datos $\{\boldsymbol{x}_n\}$ generarán distintos subespacios que en ocasiones es interesante comparar. Por ejemplo, si se disponen de diversos conjuntos de imágenes de entrenamiento, cada uno con rostros de una misma persona perfectamente identificada, se podría utilizar el KPCA para generar un subespacio para las imágenes de cada individuo. Dado entonces un conjunto de imágenes de una persona concreta, sería posible hallar su subespacio para posteriormente compararlo con los subespacios de entrenamiento y determinar, así, la identidad del individuo. Esta técnica se ha utilizado con éxito, ofreciendo mejor rendimiento que la opción más usual de generar un subespacio común para todas las imágenes de entrenamiento (Wang *et al.*, 2006). También ha sido utilizada en el reconocimiento basado en vídeo, tanto para la detección de caras (Yamaguchi *et al.*, 1998) como para la detección de trayectorias irregulares en el movimiento de un individuo (Wolf y Shashua, 2003a,b). La ventaja indudable de esta técnica es que permite comparar conjuntos de datos entre sí en lugar de datos individuales.

4.6.1. Distancia entre subespacios

Varios autores han enfocado de maneras diferentes el problema de definir una distancia entre subespacios. Por ejemplo, Wang *et al.* (2006) parte de lo que se denomina distancia L_2-Hausdorff entre el vector de la base de un subespacio $\mathbb{U}$ y otro subespacio V, y la extienden para definir una distancia entre los subespacios $\mathbb{U}$ y $\mathbb{V}$ como

$$d(\mathbb{U},\mathbb{V}) = \sqrt{\max(R,S) - \sum_{i=1}^{m}\sum_{j=1}^{n}(\boldsymbol{u}_i^T\boldsymbol{v}_j)^2} \tag{4.49}$$

donde $\boldsymbol{u}_i$ e $\boldsymbol{v}_j$ son el i-ésimo y el j-ésimo vector de la base de los subespacios $\mathbb{U}$ y $\mathbb{V}$, de dimensiones R y S respectivamente. Ésta se generaliza posteriormente al caso no lineal sustituyendo (4.6) en la página 62 en (4.49)

$$d(\mathbb{U},\mathbb{V}) = \sqrt{\max(R,S) - \sum_{i=1}^{R}\sum_{j=1}^{S}(\boldsymbol{u}_i^T\boldsymbol{v}_j)^2}$$

$$= \sqrt{\max(R,S) - \sum_{i=1}^{R}\sum_{j=1}^{S}\left(\sum_{n=1}^{N}\sum_{m=1}^{M} a_{in}b_{jm}\tilde{\Phi}(\boldsymbol{x}_n)^T\tilde{\Phi}(\boldsymbol{x}'_m)\right)^2}$$

$$= \sqrt{\max(R,S) - \sum_{i=1}^{R}\sum_{j=1}^{S}\left(\sum_{n=1}^{N}\sum_{m=1}^{M} a_{in}b_{jm}\tilde{k}(\boldsymbol{x}_n,\boldsymbol{x}'_m)\right)^2} \tag{4.50}$$

donde $\{\boldsymbol{x}_n\}$ y $\{\boldsymbol{x}'_m\}$ son los conjuntos de datos de entrada que generan los subespacios $\mathbb{U}$ y $\mathbb{V}$ respectivamente. Esta definición de la distancia entre subespacios es invariante a la elección de base de los mismos y $d(\mathbb{U},\mathbb{V}) \leq \sqrt{\max(mn,n)}$. Otros detalles acerca de (4.50), así como sus aplicaciones, han sido desarrollados por Wang *et al.* (2006) y Sun *et al.* (2007).

4.6.2. Ángulos principales basados en kernel

Por otro lado Wolf y Shashua (2003a,b) generalizan el concepto de *ángulos principales* al caso no lineal. En general los ángulos principales $\theta_1,\ldots,\theta_Q$ entre dos subespacios $\mathbb{U}$ y $\mathbb{V}$ se definen como

$$\cos(\theta_q) = \max_{\boldsymbol{u}\in\mathbb{U}}\max_{\boldsymbol{v}\in\mathbb{V}} \boldsymbol{u}^T\boldsymbol{v} = \boldsymbol{u}_q^T\boldsymbol{v}_q \tag{4.51}$$

donde

$$\begin{aligned} \boldsymbol{u}^T\boldsymbol{u} = \boldsymbol{v}^T\boldsymbol{v} &= 1 \\ \boldsymbol{u}^T\boldsymbol{u}_i &= 0 \qquad \text{para } i = 1,\ldots,q-1 \\ \boldsymbol{v}^T\boldsymbol{v}_i &= 0 \qquad \text{para } i = 1,\ldots,q-1 \end{aligned}$$

Los vectores $\{\boldsymbol{u}_i\}$ y $\{\boldsymbol{v}_i\}$ son los *vectores principales* entre los subespacios $\mathbb{U}$ y $\mathbb{V}$. En su trabajo, Wolf y Shashua (2003a) comentan diversas técnicas para hallar los ángulos principales entre dos subespacios descritos por dos conjuntos de datos mapeados en el espacio de características, destacando que la dificultad de desarrollar este tipo de técnicas está en que siempre deben de poder ser compuestas en términos de la función de kernel.

Sin embargo, en el caso de haber aplicado un KPCA sobre dos conjuntos de datos, resulta sencillo hallar los ángulos principales entre los subespacios de componentes principales de ambos conjuntos. Suponiendo que las columnas de $\mathbf{U} \in \mathbb{R}^{M\times R}$ y $\mathbf{V} \in \mathbb{R}^{M\times S}$ definen las bases ortogonales de los subespacios $\mathbb{U}$ y $\mathbb{V}$ del espacio de características de dimensión M, respectivamente. Entonces se puede reescribir (4.51) como

$$\max_{\substack{\boldsymbol{u}\in\mathbb{U}\\ \|u\|=1}} \max_{\substack{\boldsymbol{v}\in\mathbb{V}\\ \|v\|=1}} \boldsymbol{u}^T\boldsymbol{v} = \max_{\substack{\boldsymbol{w}\in\mathbb{R}^R\\ \|w\|=1}} \max_{\substack{\boldsymbol{z}\in\mathbb{R}^S\\ \|z\|=1}} \boldsymbol{w}^T(\mathbf{U}^T\mathbf{V})\boldsymbol{z} \tag{4.52}$$

donde, al igual que para obtener (4.50), se puede utilizar (4.6) para escribir $\mathbf{U}^T\mathbf{V}$ en términos de la función de kernel

$$\max_{\substack{\boldsymbol{w}\in\mathbb{R}^R\\ \|w\|=1}} \max_{\substack{\boldsymbol{z}\in\mathbb{R}^S\\ \|z\|=1}} \boldsymbol{w}^T(\mathbf{A}^T\tilde{\mathbf{K}}_\times\mathbf{B})\boldsymbol{z} \tag{4.53}$$

donde $\mathbf{A}$ y $\mathbf{B}$ son matrices de $N \times R$ y $N \times S$ cuyos elementos son los coeficientes a_{in} y b_{jm} de (4.6) para cada conjunto de datos, respectivamente. Mientras que $\tilde{\mathbf{K}}_\times$ es una matriz de elementos $\tilde{k}_{nm} = \tilde{k}(\boldsymbol{x}_n, \boldsymbol{x}'_m)$. Por las propiedades de la SVD, si $\mathbf{W}^T(\mathbf{A}^T\tilde{\mathbf{K}}_\times\mathbf{B})\mathbf{Z} = \mathrm{diag}(\sigma_1, \ldots, \sigma_q)$ fuera la SVD de $\tilde{\mathbf{K}}_\times$, entonces

$$\cos(\theta_q) = \sigma_q \qquad \text{para } q = 1, \ldots, Q \tag{4.54}$$

mientras que los vectores principales no pueden ser calculados puesto que sería necesario conocer la transformación no lineal $\Phi(x)$.

Obtenidos los ángulos principales puede ser interesante calcular a partir de éstos una medida de similaridad que pueda utilizarse en, por ejemplo, tareas de clasificación. Concretamente, Wolf y Shashua (2003a) proponen

$$\prod_{q=1}^{Q} \cos(\theta_q)^2 \tag{4.55}$$

como criterio porque al calcular los ángulos principales directamente a partir

de los datos en el espacio de entrada, es interesante tener una magnitud que se pueda escribir como producto escalar para utilizarla como función de kernel en algoritmos basados en kernel como máquina de soporte vectorial (SVM) o KPCA.

4.6.3. Divergencia de Kullback-Leibler

Una alternativa a lo comentado hasta el momento es utilizar la divergencia de Kullback-Leibler[6] en el espacio de características, suponiendo que los datos en dicho espacio forman una distribución gaussiana.

Sean $\mathcal{N}(\boldsymbol{\mu}, \mathbf{C})$ y $\mathcal{N}'(\boldsymbol{\mu}', \mathbf{C}')$ las distribuciones normales de dos conjuntos de datos $\{\boldsymbol{x}_n\}$ y $\{\boldsymbol{x}'_m\}$ proyectados en su propio subespacio de componentes principales no lineal mediante KPCA. La divergencia de Kullback-Leibler de dichas distribuciones sería

$$D_{KL}(\mathcal{N}\|\mathcal{N}') = \frac{1}{2}\left[\log\left(\frac{\det(\mathbf{C}')}{\det(\mathbf{C})}\right) + \operatorname{tr}(\mathbf{C}'^{-1}\mathbf{C}) + (\boldsymbol{\mu}' - \boldsymbol{\mu})\mathbf{C}'^{-1}(\boldsymbol{\mu}' - \boldsymbol{\mu}) - \Omega\right] \tag{4.56}$$

Debido a las propiedades del determinante, el primer término de (4.56) puede calcularse a partir de los autovalores de la matrices de covarianza $\mathbf{C}$ y $\mathbf{C}'$, calculados al resolver (4.11) en la página 63 para el KPCA de ambos conjuntos de datos

$$\log\left(\frac{\det(\mathbf{C}')}{\det(\mathbf{C})}\right) = \log\left(\frac{\prod_{j=1}^{S}\lambda'_j}{\prod_{i=1}^{R}\lambda_i}\right) \tag{4.57}$$

Mientras que el segundo término de (4.56) puede calcularse sustituyendo las matrices de covarianza por su autodescomposición, para a continuación sustituir por (4.4) en la página 62 con el objetivo de descomponer los autovectores, y finalmente utilizar (4.11) en la página 63 para dejar la expresión en términos de la función de kernel. Después de algunas simplificaciones el segundo término

[6] La divergencia de Kullback-Leibler $D_{KL}(\mathcal{P}\|\mathcal{Q})$ no es una métrica puesto que no es simétrica, por lo que en ocasiones puede ser interesante utilizar la expresión $D_{KL}(\mathcal{P}\|\mathcal{Q}) + D_{KL}(\mathcal{Q}\|\mathcal{P})$ que sí es simétrica además de no negativa.

de la divergencia de Kullback-Leibler (4.56) puede escribirse como

$$\mathrm{tr}(\mathbf{C}'^{-1}\mathbf{C}) = \left(\frac{M-1}{N-1}\right) \|\tilde{\mathbf{K}}_{\times}\tilde{\mathbf{K}}'^{-1}\|_F^2 \tag{4.58}$$

donde N y M son el número de puntos de $\{\boldsymbol{x}_n\}$ y $\{\boldsymbol{x}'_m\}$ respectivamente. Finalmente el tercer término de (4.56) puede ser ignorado puesto que los datos fueron centrados antes de la descomposición en componentes principales, por lo que su media es $\mathbf{0}$. Ω debe valer el número de dimensiones de las distribuciones gaussianas para las que se calcula la divergencia de Kullback-Leibler, que al ser un factor constante puede ser eliminado de la expresión. Sustituyendo (4.57) y (4.58) en (4.56), esta última puede escribirse como

$$D_{KL}(\mathcal{N}\|\mathcal{N}') = \frac{1}{2}\left[\log\left(\frac{\prod_{j=1}^{S}\lambda'_j}{\prod_{i=1}^{R}\lambda_i}\right) + \left(\frac{M-1}{N-1}\right)\|\tilde{\mathbf{K}}_{\times}\tilde{\mathbf{K}}'^{-1}\|_F^2\right] \tag{4.59}$$

4.6.4. Estimación contraída de la matriz de covarianza en el subespacio de componentes principales

El principal inconveniente de (4.59) es que $\tilde{\mathbf{K}}'$ puede ser singular. Al fin y al cabo lo único garantizado por construcción es que dicha matriz es semidefinida positiva, lo que implica que $\lambda'_j \geq 0, \forall j$. Una solución podría ser estimar las matrices de covarianza necesarias para resolver (4.56) a partir de los puntos proyectados en su subespacio de componentes principales. Es decir, que (4.56) compararía las distribuciones de probabilidad de los conjuntos de datos en las coordenadas de sus propios subespacios y no en el espacio de características. Sin embargo esto tampoco garantiza que $\mathbf{C}'$ vaya a ser invertible, requisito indispensable para utilizar (4.56). Una solución a este problema es el uso de un *estimador contraído* de la matriz de covarianza.

Sean $\boldsymbol{\psi} = [\psi_1, \ldots, \psi_p]$ los parámetros de un modelo sin restricciones de elevada dimensionalidad. Por ejemplo, los elementos de la matriz de covarianza de una distribución normal multivariable. Sean $\boldsymbol{\omega} = [\omega_1, \ldots, \omega_q]$ los parámetros de

un modelo restrictivo de baja dimensionalidad. Por ejemplo, el único valor de varianza necesario en un modelo donde se asume que las variables de la distribución normal multivariable no están correladas y que todas tienen la misma varianza. Ajustando ambos modelos a los datos observados se obtiene una estimación $\boldsymbol{u}$ para los parámetros del modelo no restrictivo y otra estimación $\boldsymbol{v}$ para los del restrictivo. Obviamente la estimación $\boldsymbol{u}$ podría presentar una gran varianza debido al gran numero de parámetros que tienen que ser ajustado, mientras que $\boldsymbol{v}$ tendrá una menor varianza pero seguramente un mayor sesgo como estimador del verdadero $\boldsymbol{\psi}$. En lugar de tener que escoger una de estas dos opciones, el estimador contraído permite combinarlas

$$\hat{\boldsymbol{u}} = \tau \boldsymbol{v} + (1 - \tau)\boldsymbol{u} \qquad \text{para } \tau \in [0, 1] \tag{4.60}$$

donde τ indica la intensidad de la contracción. La cuestión entonces es como seleccionar el valor óptimo para dicho coeficiente de contracción. Schäfer y Strimmer (2005) dan una respuesta general a este problema que puede ser fácilmente aplicable a los casos más comunes.

Sea $\mathbf{C}$ la estimación insesgada de la matriz de covarianza para el conjunto de datos proyectados $\{\boldsymbol{y}_n\}$ en su subespacio de componentes principales

$$\mathbf{C} = \frac{1}{N-1} \sum_{n=1}^{N} \boldsymbol{y}_n \boldsymbol{y}_n^T \tag{4.61}$$

y sea $\hat{\mathbf{C}}$ una estimación contraída de la misma matriz de covarianza

$$\hat{\mathbf{C}} = \tau \mathbf{I} + (1 - \tau)\mathbf{C} \qquad \text{para } \tau \in [0, 1] \tag{4.62}$$

donde el modelo restrictivo es el de una matriz de covarianza diagonal de media unidad. Entonces el óptimo de τ será

$$\tau = \frac{\sum_{i \neq j} \widehat{\text{var}}(c_{ij}) + \sum_i \widehat{\text{var}}(c_{ii})}{\sum_{i \neq j} c_{ij}^2 + \sum_i (c_{ii} - 1)^2} \tag{4.63}$$

donde falta obtener una expresión para $\widehat{\text{var}}(c_{ij})$, la estimación de la varianza insesgada de los elementos de $\mathbf{C}$. Suponiendo que el número de puntos $\boldsymbol{y}_n$ es N y que éstos han sido proyectados un subespacio de dimensión R, se puede definir la media muestral para la componente i-ésima del conjunto de datos como $\langle y_i \rangle = \frac{1}{N}\sum_{n=1}^{N} y_{ni}$. Al mismo tiempo se puede definir la covarianza muestral sesgada entre las componentes i-ésima y j-ésima como $\langle w_{ij} \rangle = \frac{1}{N}\sum_{n=1}^{N} w_{nij}$, donde $w_{nij} = (y_{ni} - \langle y_i \rangle)(y_{nj} - \langle y_j \rangle)$. Entonces la covarianza muestral insesgada sería

$$c_{ij} = \frac{N}{N-1}\langle w_{ij} \rangle \tag{4.64}$$

Por lo que la varianza insesgada de los elementos de $\mathbf{C}$ se puede estimar como

$$\begin{aligned} \widehat{\text{var}}(c_{ij}) &= \frac{N^2}{(N-1)^2}\,\widehat{\text{var}}(\langle w_{ij} \rangle) \\ &= \frac{N}{(N-1)^2}\,\widehat{\text{var}}(w_{nij}) \\ &= \frac{N}{(N-1)^3}\sum_{n=1}^{N}(w_{nij} - \langle w_{ij} \rangle)^2 \end{aligned} \tag{4.65}$$

Gracias al estimador contraído (4.62) se pueden calcular estimaciones de las matrices de covarianza para los conjuntos de datos $\{\boldsymbol{y}_n\}$ e $\{\boldsymbol{y}'_m\}$, que posteriormente puedan ser empleadas en (4.56) para la obtención de la divergencia de Kullback-Leibler entre ambos conjuntos de datos.

Por otro lado, si los datos han sido clasificados, es posible utilizar la divergencia de Kullback-Leibler para comparar las distribuciones de probabilidad de los datos en diferentes clases. En teoría esto proporcionaría una herramienta para evaluar la similaridad entre dos clases l_1 y l_2 en dos conjuntos de datos $\{\boldsymbol{y}_n\}$ y $\{\boldsymbol{y}'_m\}$ respectivamente

$$\begin{aligned} D_{KL}(\mathcal{N}_{l_1} \| \mathcal{N}'_{l_2}) = \frac{1}{2}\Bigg[&\log\left(\frac{\det(\mathbf{C}'_{l_2})}{\det(\mathbf{C}_{l_1})}\right) + \text{tr}(\mathbf{C}'^{-1}_{l_2}\mathbf{C}_{l_1}) \\ &+ (\boldsymbol{\mu}'_{l_2} - \boldsymbol{\mu}_{l_1})\mathbf{C}'^{-1}_{l_2}(\boldsymbol{\mu}'_{l_2} - \boldsymbol{\mu}_{l_1}) - \Omega\Bigg] \end{aligned} \tag{4.66}$$

donde el tercer término puede ser ignorado puesto que los datos fueron centrados antes de la descomposición en componentes principales.

Sea $\mathbf{C}_l$ la estimación insesgada de la matriz de covarianza para el conjunto de datos proyectados $\{\boldsymbol{y}_n\}$ condicionados a su pertenencia a la clase l

$$\mathbf{C}_l = \frac{\sum_{m=1}^{N} p_{ml}}{\left(\sum_{m=1}^{N} p_{ml}\right)^2 - \left(\sum_{m=1}^{N} p_{ml}^2\right)} \sum_{n=1}^{N} \boldsymbol{y}_n \boldsymbol{y}_n^T p_{nl} \tag{4.67}$$

de manera que se pretende obtener un estimador contraído (4.62) de la misma matriz de covarianza, siendo (4.63) la expresión para el coeficiente de contracción τ en (4.62). Nuevamente hay que obtener una expresión para $\widehat{\mathrm{var}}(c_{ij})$, pero en esta ocasión hay que tener en cuenta que c_{ij} son los elementos de $\mathbf{C}_l$, la matriz de covarianza condicionada a la pertenencia de los puntos a la clase l. Así se puede definir la media muestral para la componente i-ésima de los puntos del conjunto de datos como

$$\langle y_i \rangle = \frac{1}{\sum_{m=1}^{N} p_{ml}} \sum_{n=1}^{N} y_{ni} p_{nl} \tag{4.68}$$

y la covarianza muestral sesgada entre las componentes i-ésima y j-ésima como

$$\langle w_{ij} \rangle = \frac{1}{\sum_{m=1}^{N} p_{ml}} \sum_{n=1}^{N} w_{nij} p_{nl}, \tag{4.69}$$

donde

$$w_{nij} = (y_{ni} - \langle y_i \rangle)(y_{nj} - \langle y_j \rangle) \tag{4.70}$$

Entonces la covarianza muestral insesgada condicionada a l sería

$$c_{ij} = \frac{\left(\sum_{m=1}^{N} p_{ml}\right)^2}{\left(\sum_{m=1}^{N} p_{ml}\right)^2 - \left(\sum_{m=1}^{N} p_{ml}^2\right)} \langle w_{ij} \rangle \tag{4.71}$$

Por lo que la varianza insesgada de los elementos de **C** condicionada a l se puede estimar como

$$\begin{aligned}
\widehat{\mathrm{var}}(c_{ij}) &= \frac{\left(\sum_{m=1}^{N} p_{ml}\right)^4}{\left[\left(\sum_{m=1}^{N} p_{ml}\right)^2 - \left(\sum_{m=1}^{N} p_{ml}^2\right)\right]^2} \widehat{\mathrm{var}}(\langle w_{ij}\rangle) \\
&= \frac{\left(\sum_{m=1}^{N} p_{ml}\right)^2 \left(\sum_{m=1}^{N} p_{ml}^2\right)}{\left[\left(\sum_{m=1}^{N} p_{ml}\right)^2 - \left(\sum_{m=1}^{N} p_{ml}^2\right)\right]^2} \widehat{\mathrm{var}}(w_{nij}) \\
&= \frac{\left(\sum_{m=1}^{N} p_{ml}\right)^3 \left(\sum_{m=1}^{N} p_{ml}^2\right)}{\left[\left(\sum_{m=1}^{N} p_{ml}\right)^2 - \left(\sum_{m=1}^{N} p_{ml}^2\right)\right]^3} \sum_{n=1}^{N} (w_{nij} - \langle w_{ij}\rangle)^2 p_{nl} \qquad (4.72)
\end{aligned}$$

4.7. Análisis de correlación canónica basado en kernel

Si se cuenta con diversos conjuntos de datos, que de alguna manera deben ser analizados juntos puesto que se supone la existencia de alguna relación, en ocasiones puede ser interesante mapear los datos en el espacio de características pero, en lugar de buscar para cada conjunto el subespacio definido por las direcciones de mayor variabilidad, buscar un subespacio común a todos que maximice la covariabilidad de los mismos. A este tipo de análisis se lo denomina análisis de correlación canónica basado en kernel (KCCA) (Akaho, 2001; Bach y Jordan, 2003; Melzer *et al.*, 2001).

El KCCA resulta particularmente interesante cuando se tienen conjuntos de datos en los que se sabe que existen relaciones subyacentes, aunque desconocidas, que seguramente no podrían ser descritas mediantes transformaciones lineales. En esos casos puede servir para revelar la estructura oculta de dichas relaciones. El KCCA ha sido aplicado con éxito extrayendo relaciones no linea-

les entre variables en genómica (Yamanishi *et al.*, 2003), imagen por resonancia magnética funcional (FMRI) del cerebro (Hardoon *et al.*, 2004), análisis de componentes independientes (ICA) (Bach y Jordan, 2003), series temporales caóticas (Suetani *et al.*, 2006), recuperación y categorización de texto (Li y Shawe-Taylor, 2006), recuperación de imágenes a partir de consultas textuales (Hardoon y Shawe-Taylor, 2003) y recuperación de la postura mediante modelos de apariencia (Melzer *et al.*, 2003).

Sean $\{\boldsymbol{x}_n\}$ y $\{\boldsymbol{x}'_n\}$, donde $n = 1, \ldots, N$ dos conjuntos de datos en el espacio de entrada de dimensión D. Suponiendo que ambos conjuntos son mapeados en un espacio de características mediante dos transformaciones no lineales $\Phi(x)$ y $\Psi(x)$, obteniéndose los conjuntos de datos $\{\Phi(\boldsymbol{x}_n)\}$ y $\{\Psi(\boldsymbol{x}'_n)\}$ respectivamente. Se buscan dos vectores $\boldsymbol{u}$ y $\boldsymbol{v}$ sobre los que proyectar los conjuntos de datos $\{\Phi(\boldsymbol{x}_n)\}$ y $\{\Psi(\boldsymbol{x}'_n)\}$

$$y_n = \boldsymbol{u}^T \Phi(\boldsymbol{x}_n) \tag{4.73}$$

$$y'_n = \boldsymbol{v}^T \Psi(\boldsymbol{x}'_n) \tag{4.74}$$

tales que los conjuntos de puntos proyectados $\{y_n\}$ y $\{y'_n\}$ estén máximamente correlados

$$\arg\max_{u,v} \frac{\sum_{n=1}^{N} y_n y'_n}{\sqrt{\left(\sum_{n=1}^{N} y_n^2\right)\left(\sum_{n=1}^{N} y_n'^2\right)}} \tag{4.75}$$

Puesto que tanto $\boldsymbol{u}$ como $\boldsymbol{v}$ pueden ser reescalados sin cambiar el problema, éste puede ser reescrito introduciendo restricciones sobre las varianzas de $\{y_n\}$ y de $\{y'_n\}$

$$\arg\max_{v,u} \sum_{n=1}^{N} y_n y'_n \qquad \text{donde} \sum_{n=1}^{N} y_n^2 = \sum_{n=1}^{N} y_n'^2 = 1 \tag{4.76}$$

Para evitar las soluciones triviales se añade a (4.76) un factor de regularización de la forma $\tau(u^T u + v^T v)/2$. El efecto de esto es que en base al *representer theorem*[7] (véase Schölkopf y Smola, 2001, cap. 4) está garantizado que las soluciones

[7] El *representer theorem* (Kimeldorf y Wahba, 1971) establece que la solución al problema de en-

para $\boldsymbol{u}$ y $\boldsymbol{v}$ son de la forma

$$\boldsymbol{u} = \sum_{n=1}^{N} a_n \Phi(\boldsymbol{x}_n) \tag{4.77}$$

$$\boldsymbol{v} = \sum_{n=1}^{N} b_n \Psi(\boldsymbol{x}'_n) \tag{4.78}$$

lo que permite escribir (4.76), incluyendo el factor de regularización, como

$$\arg\max_{u,v} \boldsymbol{a}^T \mathbf{K}\mathbf{K}' \boldsymbol{b} \qquad \text{donde} \begin{cases} \boldsymbol{a}^T \mathbf{K}^2 \boldsymbol{a} + \tau \boldsymbol{a}^T \boldsymbol{a} = 1 \\ \boldsymbol{b}^T \mathbf{K}'^2 \boldsymbol{b} + \tau \boldsymbol{b}^T \boldsymbol{b} = 1 \end{cases} \tag{4.79}$$

donde $\mathbf{K}$ y $\mathbf{K}'$ son las matrices de kernel de $\{\boldsymbol{x}_n\}$ y $\{\boldsymbol{x}'_n\}$ respectivamente, $\boldsymbol{a} = [a_1, \ldots, a_N]^T$ y $\boldsymbol{b} = [b_1, \ldots, b_N]^T$. Usando el método de los multiplicadores de Lagrange y las condiciones de Karush-Kuhn-Tucker, el problema de optimización (4.79) se puede escribir como un problema de autovalores generalizado

$$\begin{bmatrix} 0 & \mathbf{K}'\mathbf{K} \\ \mathbf{K}\mathbf{K}' & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{b} \end{bmatrix} = \lambda \begin{bmatrix} \mathbf{K}'^2 + \tau\mathbf{I} & 0 \\ 0 & \mathbf{K}^2 + \tau\mathbf{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{b} \end{bmatrix} \tag{4.80}$$

De esta manera, el procedimiento completo para el KCCA con centrado por clases de dos conjuntos de datos cualquiera se resume en el algoritmo 4.4.

Respecto al coeficiente de regularización τ, las simulaciones numéricas demuestran que el que sea del orden de $N^{-1/3}$ es condición suficiente para la convergencia del problema en casos generales. El valor óptimo de τ depende de de las propiedades estadísticas de los datos de entrada, tales como la distribución espectral de las matrices de kernel (Fukumizu *et al.*, 2007).

contrar un $f \in \mathbb{H}$ —donde $\mathbb{H}$ es un espacio de Hilbert— tal que $\sum_{n=1}^{N} C(\boldsymbol{y}_n, \boldsymbol{x}_n) + \tau\|f\|^2$ y donde C es convexa en f, se puede expresar como $f(\boldsymbol{x}) = \sum_{n=1}^{N} a_n k(\boldsymbol{x}_n, \boldsymbol{x})$.

Algoritmo 4.4. KCCA de dos conjuntos de datos con centrado por clases.

1. Calcular las matrices $\mathbf{K}$ y $\mathbf{K}'$ a partir de las funciones de kernel $k(\boldsymbol{x}_n, \boldsymbol{x}_m)$ y $k'(\boldsymbol{x}'_n, \boldsymbol{x}'_m)$ seleccionadas. En general no hay motivo alguno para que ambas funciones tengan que ser la misma. La elección sólo depende de la naturaleza de los datos.

2. Centrar el conjunto de datos en el espacio de características utilizando (4.27)

$$\tilde{\mathbf{K}} = (\mathbf{I} - \mathbf{W})\,\mathbf{K}\,(\mathbf{I} - \mathbf{W})$$
$$\tilde{\mathbf{K}}' = \left(\mathbf{I} - \mathbf{W}'\right)\mathbf{K}'\left(\mathbf{I} - \mathbf{W}'\right)$$

   donde $\mathbf{W} = \mathbf{P}\mathbf{D}\mathbf{P}^T$ y $\mathbf{D}$ es una matriz diagonal de elementos $d_{ll} = \frac{1}{\sum_{n=1}^{N} p_{nl}}$. Y siendo definida $\mathbf{W}'$ de la misma manera pero con $\mathbf{P}'$ en lugar de con $\mathbf{P}$.

3. Resolver el problema de autovalores generalizado (4.80) en la página anterior

$$\begin{bmatrix} 0 & \tilde{\mathbf{K}}'\tilde{\mathbf{K}} \\ \tilde{\mathbf{K}}\tilde{\mathbf{K}}' & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{b} \end{bmatrix} = \lambda \begin{bmatrix} \tilde{\mathbf{K}}'^2 + \tau\mathbf{I} & 0 \\ 0 & \tilde{\mathbf{K}}^2 + \tau\mathbf{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{b} \end{bmatrix}$$

   para los R autovectores $[\boldsymbol{a}_i^T \boldsymbol{b}_i^T]^T$ de mayor autovalor λ_i.

4. Proyectar el conjunto de datos en el nuevo subespacio mediante la expresión

$$\boldsymbol{y}_n = \mathbf{A}^T \tilde{\boldsymbol{k}}_n \qquad \text{donde } \mathbf{A} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_R]$$
$$\boldsymbol{y}'_n = \mathbf{B}^T \tilde{\boldsymbol{k}}'_n \qquad \text{donde } \mathbf{B} = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_R]$$

Capítulo

5

Correspondencia de puntos de interés mediante KPCA

Una de los pasos fundamentales en el reconocimiento gestual y en otras tareas en las que se usa la visión por computador es la asignación de correspondencias entre puntos. Generalmente, después de algunas etapas de preprocesamiento, se procede a extraer algunos puntos de interés de las imágenes. Estos puntos se escogen de forma que posean características identificativas de los objetos de la escena, de manera que en etapas posteriores pueden ser utilizados para segmentar o detectar estos objetos en la imagen. Además también sirven, en una etapa posterior o en paralelo a la detección, para extraer información sobre el movimiento, la deformación o la forma tridimensional de los objetos. Para estos casos es necesario disponer de una secuencia de imágenes donde aparezca el mismo objeto, en las que se procede a asignar correspondencias entre los puntos de interés. Es decir, se intentan agrupar todos los puntos de interés de todas las imágenes en función de que representen al mismo punto del objeto. Para hacer eso se utilizan las características de cada punto, por lo que deben ser lo suficientemente identificativas. Entre dichas características destacan el uso del color, la textura, los bordes de la figura, etc. Sea como fuere, lo realmente relevante a la hora de escoger las características y los puntos de interés es que haya información suficiente para asignar las correspondencias. Es importante destacar que una asignación correcta es de vital importancia en muchas tareas de visión por computador.

Por ejemplo, si se utiliza un modelo de distribución de puntos (PDM) (Cootes *et al.*, 1992, 1995) es necesario asegurar una asignación de correspondencias para aprender con garantías los modos de variación de la forma del objeto. En caso contrario el modelo podría acabar representando los errores de correspondencia en lugar de los modos de variación.

En todo caso la correspondencia de puntos no sólo puede mejorarse haciendo una buena selección de puntos de interés y de sus características, sino también añadiendo información sobre el tipo de transformaciones que pueden afectar al objeto. Si se conoce de ante mano que la transformación debe ser rígida, se puede restringir el conjunto de las posibles soluciones de correspondencia a aquellas que encajan con transformaciones afines o de perspectiva. Mientras que si por el contrario las transformaciones no son rígidas, se puede desarrollar un modelo del objeto que restringa las correspondencias, por ejemplo, mediante *splines* o difeomorfismos (Hongyu *et al.*, 2004),

Como se ha comentado (véase el capítulo 2 en la página 13), el principal inconveniente cuando se trata de identificar la postura de una mano humana es que ésta no presenta suficientes puntos de interés. Es decir, a la resolución de una cámara convencional, una mano desnuda tiene un color mas o menos uniforme y carece de textura. Tampoco presenta formas angulosas que permitan el uso de detectores de esquinas, algo que ayudaría a identificar los puntos de interés y facilitaría la correspondencia. Teniendo en cuenta todo esto, parece que en lugar de considerar el problema desde el punto de vista de detectar una serie de puntos individuales y encajarlos, podría más interesante tomar una gran número de ellos y buscar que sea la información relativa a la distribución de los mismos la que permita realizar la asignación de correspondencias. Al fin y al cabo la mayor parte de la información aprovechable, desde el punto de vista de determinar la postura, está en la forma de la mano. En este sentido parece que el KPCA puede ser de gran ayuda, puesto que una función de kernel adecuada aseguraría la invarianza frente a diferentes tipos de transformaciones entre imágenes, al tiempo que el mapeo no lineal en el espacio de características podría capturar las relaciones no lineales entre los puntos de cada una de éstas.

Una aproximación similar fue desarrollada por Shapiro y Brady (1992), que construyen una matriz de similaridades usando una función de peso gaussiana entre los puntos de una misma imagen, con el objetivo de capturar la información acerca de la estructura de la misma. A continuación los autovectores de la matriz de similaridad de cada imagen son dispuestos como las columnas de una matriz modal. En dichas matrices se puede entender que cada fila corresponde a las coordenadas de un punto en el autoespacio, por lo que las correspondencias se pueden obtener comparando las filas de las matrices modales de diferentes imágenes. Esto puede ser visto como proyectar los puntos individuales en un autoespacio, para a continuación hallar las correspondencias buscando el punto más cercano en dicho autoespacio. Luo y Hancock (1999) usan escalado multidimensional (MDS) para, mediante un alineamiento de Procrustes en el autoespacio, intentar resolver los problemas derivados de la diferente estructura de los conjuntos de datos. Wang y Hancock (2006) exploran el uso del KPCA tanto para la correspondencia de puntos en objetos rígidos como articulados.

5.1. Correspondencia de puntos en objetos rígidos

La idea fundamental detrás de la asignación de correspondencias de puntos mediante KPCA es la de capturar relaciones invariantes a diferentes tipos de transformaciones entre los puntos de una misma imagen, esperando que esas relaciones sirvan para obtener la correspondencia de los puntos entre imágenes diferentes. El procedimiento concreto para la correspondencia en objetos rígidos se resume en el algoritmo 5.1.

Suponiendo que se dispone de dos imágenes a cada una de las cuales se les ha extraído un conjunto de puntos $\{\boldsymbol{x}_n\}$ y $\{\boldsymbol{x}'_m\}$, de tamaño N y M respectivamente, representados por sus coordenadas en la imagen correspondiente, se busca hallar la correspondencia entre ambos conjuntos de datos. El primer paso es crear una matriz de similaridad $\mathbf{K}$ para cada imagen, de forma parecida a como lo

Algoritmo 5.1. Correspondencia de puntos en objetos rígidos.

1. Se crean las matrices de similaridad $\mathbf{K}$ y $\mathbf{K}'$ para los conjuntos de puntos $\{\boldsymbol{x}_n\}$ y $\{\boldsymbol{x}'_m\}$ de cada imagen mediante cualquier medida de similaridad que se defina con una función semidefinida positiva (véase el apartado 4.3 en la página 65).
2. A cada matriz se le aplica el procedimiento del KPCA resumido en el algoritmo 4.1.
3. Se hayan las correspondencias de cada punto proyectado $\boldsymbol{y}_n$ resolviendo (5.2)
$$\boldsymbol{y}'_m = \arg\min_{\boldsymbol{y}'_m} \|\boldsymbol{y}_n - \boldsymbol{y}'_m\| \qquad \text{para } n = 1, \ldots, N$$

hacen Shapiro y Brady (1992)

$$k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \exp(-\|\boldsymbol{x}_n - \boldsymbol{x}_m\|^2 / c) \tag{5.1}$$

Esta expresión es la misma que la de la función de kernel gaussiana (véase (4.17) en la página 66) por lo que la matriz $\mathbf{K}$ no es mas que una matriz de kernel. En realidad, tal y como comenta Wang (2007), el método descrito por Shapiro y Brady (1992) no es más que un caso particular de KPCA donde los datos en el espacio de características tienen media cero y la función de kernel es gaussiana. En general, cualquier medida de similaridad que se defina mediante una función semidefinida positiva (véase el apartado 4.3 en la página 65) puede utilizarse para crear la matriz $\mathbf{K}$. Esto permite escoger una medida adaptada a la naturaleza de los datos.

Sea cual sea la función de kernel escogida, para cada imagen se debe seguir el mismo procedimiento que para el KPCA, que ha sido descrito en el capítulo 4 y resumido en el algoritmo 4.1 en la página 69. Es decir, la matriz $\mathbf{K}$ debe ser centrada, para a continuación resolver el problema de autovalores y, finalmente, proyectar el conjunto de puntos en su subespacio de componentes principa-

les. Suponiendo que los conjuntos de puntos proyectados en su subespacio de componentes principales, para cada una de las imágenes, son $\{\boldsymbol{y}_n\}$ e $\{\boldsymbol{y}'_m\}$ respectivamente, el problema de hallar la correspondencia de un punto $\boldsymbol{y}_n$ puede expresarse como

$$\boldsymbol{y}'_m = \arg\min_{\boldsymbol{y}'_m} \|\boldsymbol{y}_n - \boldsymbol{y}'_m\| \qquad \text{para } n = 1, \ldots, N \tag{5.2}$$

donde se ha usado la distancia euclídea para comparar los puntos $\boldsymbol{y}_n$ e $\boldsymbol{y}'_m$. Aunque puede utilizarse cualquier otra, como, por ejemplo, ponderando la distancia euclídea con una función de peso gaussiana.

5.2. Correspondencia de puntos en objetos articulados

Como se ha comentado anteriormente, Wang y Hancock (2006) describen una técnica para la asignación de correspondencias de puntos en objetos articulados. Esta técnica —parte de la cual se resumen en el algoritmo 5.2— se basa en suponer que cada elemento del objeto tiene una etiqueta única, de forma que los puntos de la imagen extraídos del mismo elemento tiene la misma etiqueta, es decir, pertenecen a la misma clase. Así, de un par de imágenes no sólo se extraen sendos conjuntos de puntos $\{\boldsymbol{x}_n\}$ y $\{\boldsymbol{x}'_m\}$, sino que además cada uno tiene una probabilidad de pertenecer a cada elemento del objeto. Esto se expresa a través de las matrices $\mathbf{P}$ y $\mathbf{P}'$ respectivamente, tales que el elemento p_{nl} contiene la probabilidad de que el n-ésimo punto pertenezca a la clase l y que $\sum_{l=1}^{L} p_{nl} = 1$ para todo n, donde L es el número de etiquetas del objeto.

Para hallar las correspondencias, al igual que en el apartado 5.1, se comienza calculado la matrices de similaridad $\mathbf{K}$ y $\mathbf{K}'$ de ambos conjuntos de puntos usando la función de kernel seleccionada. Posteriormente, ambas matrices son ponderadas utilizando la probabilidad de que cada par de puntos tengan la misma etiqueta. Así, para la primera imagen la matriz $\mathbf{K}$ se ponderaría utilizando la

Algoritmo 5.2. Correspondencia de puntos en objetos articulados.

1. Se crean las matrices de similaridad $\mathbf{K}$ y $\mathbf{K}'$ para los conjuntos de puntos $\{\boldsymbol{x}_n\}$ y $\{\boldsymbol{x}'_m\}$ de cada imagen mediante cualquier medida de similaridad que se defina con una función semidefinida positiva (véase el apartado 4.3 en la página 65).

2. Las matrices $\mathbf{K}$ y $\mathbf{K}'$ son ponderadas utilizando la probabilidad de que cada par de puntos tengan la misma etiqueta

$$\tilde{k}_{nm} = \sum_{l=1}^{L} p_{nl} p_{ml} k_{nm}$$

   suponiendo que se conocer las matrices de probabilidad $\mathbf{P}$ y $\mathbf{P}'$.

3. A cada matriz se le aplica el procedimiento del KPCA con centrado por clases resumido en el algoritmo 4.2.

4. Se hayan las correspondencias de cada punto proyectado $\boldsymbol{y}_n$ resolviendo (5.4)

$$\boldsymbol{y}'_m = \arg\min_{\boldsymbol{y}'_m} \sum_{l=1}^{L} p_{nl} p'_{ml} \exp(-\|\boldsymbol{y}_n - \boldsymbol{y}'_m\|^2/c) \qquad \text{para } n = 1, \ldots, N$$

expresión

$$\tilde{k}_{nm} = \sum_{l=1}^{L} p_{nl} p_{ml} k_{nm} \tag{5.3}$$

La matriz $\tilde{\mathbf{K}}$ es simétrica por construcción, puesto que $\mathbf{K}$ también lo es. Sin embargo, no está garantizado que sea semidefinida positiva, por lo que en teoría es necesario calcular todos sus autovalores para comprobar que así sea. A continuación, sobre ambas matrices de aplica el procedimiento del KPCA con centrado por clases, resumido en el algoritmo 4.2. Hecho esto sólo queda comparar ambos conjuntos de puntos proyectos para asignar las correspondencias según el vecino más próximo, como ocurre en (5.2). Sin embargo, aunque esto es posible hacerlo utilizando la distancia euclídea, Wang y Hancock (2006) proponen ponderar dicha distancia con una función de peso gaussiana y con la probabilidad de que ambos puntos tengan la misma etiqueta

$$\boldsymbol{y}'_m = \arg\min_{\boldsymbol{y}'_m} \sum_{l=1}^{L} p_{nl} p'_{ml} \exp(-\|\boldsymbol{y}_n - \boldsymbol{y}'_m\|^2 / c) \qquad \text{para } n = 1, \ldots, N \tag{5.4}$$

El único aspecto que queda por definir es como asignar las probabilidades de las etiquetas para cada punto. Pues, aunque las probabilidades de la primera de las imágenes pueden ser conocidas, no ocurre lo mismo con las de la segunda. Una solución podría ser el estimarlas a partir de las correspondencias con la primera de las imágenes, pero sin etiquetas en ambas imágenes no se puede aplicar esta técnica de asignación de correspondencias.

5.3. Refinamiento de las probabilidades mediante relajación

El etiquetado con relajación (Haralick y Shapiro, 1979) es una técnica muy utilizada en el procesamiento de imágenes, el reconocimiento de patrones y la inteligencia artificial. En el caso de la visión por computador se fundamenta en que cada píxel tiende a ser de la misma clase que sus vecinos más próximos. Suponiendo que a cada punto se le puede asignar una de varias etiquetas, para lo cual

cada uno tiene una probabilidad determinada de que se le asigne una etiqueta dada, el proceso de relajación va actualizando iterativamente dichas probabilidades de forma que siempre incremente la probabilidad de que un punto tenga la misma etiqueta que sus vecinos. Para ser exactos, la relajación propaga las probabilidades de un punto en base a las de sus vecinos y a unos coeficientes de compatibilidad que establecen que etiquetas pueden permanecer juntas, al ser asignadas a puntos adyacentes, y cuales no.

Retomando el problema de la asignación de correspondencias, se puede suponer que se conocen las etiquetas para el conjunto de puntos de la primera imagen $\{\boldsymbol{x}_n\}$, ya sea porque se asignaron manualmente o porque se han propagado desde la asignación de dicha imagen con una imagen anterior —por ejemplo, cuando la técnica se aplica a una secuencia de vídeo—. Mientras que a los puntos de la segunda imagen $\{\boldsymbol{x}'_m\}$ se les puede asignar probabilidades uniformes para todas las etiquetas, aplicando posteriormente un proceso de relajación con el fin de que las probabilidades se reajusten apropiadamente.

Es importante destacar que usando el etiquetado con relajación se está introduciendo, a través de los coeficientes de compatibilidad, información sobre la estructura esperada del objeto. En lugar de refinar la asignación utilizando información sobre el tipo de transformaciones que pueden afectar a los puntos, tal y como se comentó al comienzo de este capítulo, se están introduciendo restricciones sobre el tipo de relaciones que se esperan observar.

En la literatura sobre etiquetado con relajación hay una gran variedad de algoritmos diferentes (Kittler e Illingworth, 1986). Sin embargo se utilizarán las fórmulas más comunes, las de Rosenfeld *et al.* (1976). Como se ha comentado, las etiquetas no se asignaron independientemente sino que están sujetas a restricciones contextuales. Éstas se definen mediante una matriz de $M \times M$ por bloques

$$\mathbf{R} = \begin{bmatrix} \mathbf{R}_{11} & \cdots & \mathbf{R}_{1M} \\ \vdots & \ddots & \vdots \\ \mathbf{R}_{M1} & \cdots & \mathbf{R}_{MM} \end{bmatrix} \tag{5.5}$$

denominada matriz de coeficientes de compatibilidad, donde cada matriz $\mathbf{R}_{nm}$ es una matriz de $L \times L$

$$\mathbf{R}_{nm} = \begin{bmatrix} r_{nm}(1,1) & \cdots & r_{nm}(1,L) \\ \vdots & \ddots & \vdots \\ r_{nm}(L,1) & \cdots & r_{nm}(L,L) \end{bmatrix} \tag{5.6}$$

que indican la compatibilidad de cada etiqueta asignada al punto n-ésimo con cualquier otra signada al punto m-ésimo, de manera que valores altos se corresponden con mayor compatibilidad y valores bajos con incompatibilidad. El algoritmo de etiquetado con relajación recibe como entrada unas probabilidades iniciales de etiquetado $\mathbf{P}^{(0)}$ y lo actualiza iterativamente, con arreglo a lo indicado en la matriz de coeficientes de compatibilidad R, de acuerdo a la expresión

$$p_{nl}^{(t+1)} = \frac{p_{nl}^{(t)} q_{nl}^{(t)}}{\sum_{j=1}^{L} p_{nj}^{(t)} q_{nj}^{(t)}} \tag{5.7}$$

donde

$$q_{nl}^{(t)} = \sum_{m=1}^{M} \sum_{j=1}^{L} r_{nm}(l,j) p_{mj}^{(t)} \tag{5.8}$$

es la fuerza con la que los puntos del entorno apoyan que l sea la etiqueta correcta para el punto n-ésimo. Generalmente, el algoritmo itera hasta que la distancia entre etiquetados sucesivos es lo suficientemente pequeña o cuando se alcanza un número máximo de iteraciones, asignando finalmente a cada punto la etiqueta para la que tiene la mayor probabilidad.

Este proceso puede ser simplificado, por ejemplo no considerando en (5.8) todos los puntos del conjunto, si no sólo los vecinos más próximos. Wang y Hancock (2006) además simplifican la matriz $\mathbf{R}$ suponiendo homogeneidad espacial. Es decir, que las matrices $\mathbf{R}_{nm}$ que componen $\mathbf{R}$ son independientes de n y de m,

definiendo sus elementos así

$$r_{nm}(l,j) = r_{lj} = \begin{cases} 1 & \text{si } \boldsymbol{x}_n \text{ y } \boldsymbol{x}_m \text{ pertenecen al mismo elemento rígido} \\ -1 & \text{en cualquier otro caso} \end{cases} \tag{5.9}$$

También se reescribe (5.8) exponenciando el numerador, para a continuación normalizar

$$q_{nl}^{(t)} = \frac{\exp\left(\sum_{m\in\mathbf{N}_n}\sum_{j=1}^{L} r_{lj}p_{mj}^{(t)}k_{nm}\right)}{\sum_{l=1}^{L}\exp\left(\sum_{m\in\mathbf{N}_n}\sum_{j=1}^{L} r_{lj}p_{mj}^{(t)}k_{nm}\right)} \tag{5.10}$$

donde $\mathbf{N}_n$ es el conjunto de vecinos más próximos al punto n-ésimo y se ha introducido k_{nm} para ponderar la propagación de las etiquetas en función de la distancia entre los puntos. Esta modificación también lleva a reescribir (5.7)

$$p_{nl}^{(t+1)} = \frac{p_{nl}^{(t)} + \alpha q_{nl}^{(t)}}{\sum_{j=1}^{L} p_{nj}^{(t)} + \alpha q_{nj}^{(t)}} \tag{5.11}$$

donde α es una parámetro constante. Así el procedimiento para la asignación de correspondencias en objetos articulados sería tal y como se describe en el algoritmo 5.3.

En todo caso es importante tener en cuenta que el comportamiento del etiquetado con relajación es muy dependiente de la elección de los coeficientes de compatibilidad (Haralick *et al.*, 1980; O'Leary y Peleg, 1983). Por tanto no hay garantías de que con la $\mathbf{R}$ escogida la técnica vaya a tener el comportamiento esperado. Una aproximación interesante es la de aprender los coeficientes de relajación sobre un conjunto de datos de entrenamiento, de forma que se optimice el funcionamiento del proceso de relajación (Pelillo y Refice, 1994). Sin embargo, esto puede resultar bastante complejo en el caso del reconocimiento gestual de los movimientos de una mano. El elevado número de grados de libertad, las importantes diferencias entre perspectivas de una misma mano y la autooclusión, pueden hacer que el problema sea intratable, debido al número de muestras necesario para el entrenamiento, o que sea necesario entrenar diferentes matrices

Algoritmo 5.3. Correspondencia de puntos en objetos articulados con relajación.

1. Se inicializan las matrices de probabilidad $\mathbf{P}$ y $\mathbf{P}'$. La primera puede ser conocida, mientras que con la segunda se puede utilizar una distribución de probabilidad uniforme para todas las etiquetas de un mismo punto.

2. Se construye en base a (5.9) la matriz de coeficientes de compatibilidad $\mathbf{R}$ a partir del conjunto de puntos de la primera imagen $\{\boldsymbol{x}_n\}$ y de sus etiquetas

$$r_{lj} = \begin{cases} 1 & \text{si } \boldsymbol{x}_n \text{ y } \boldsymbol{x}_m \text{ pertenecen al mismo elemento rígido} \\ -1 & \text{en cualquier otro caso} \end{cases}$$

3. Se resuelve la asignación para los conjuntos $\{\boldsymbol{x}_n\}$ y $\{\boldsymbol{x}'_m\}$ (véase el algoritmo 5.2 en la página 100) utilizando las matrices $\mathbf{P}$ y $\mathbf{P}'$ respectivamente.

4. Se calcula el error de correspondencia $e = \sum_{n=1}^{N} \|\boldsymbol{x}_n - \boldsymbol{x}'_n\|_F$, deteniendo el procedimiento en caso de que el error no varíe de forma significativa respecto a la iteración anterior.

5. Se ejecuta el proceso de relajación definido por (5.10) sobre $\mathbf{P}'$

$$p'^{(t+1)}_{nl} = \frac{p'^{(t)}_{nl} + \alpha q'^{(t)}_{nl}}{\sum_{j=1}^{L} p'^{(t)}_{nj} + \alpha q'^{(t)}_{nj}}$$

   donde

$$q'^{(t)}_{nl} = \frac{\exp\left(\sum_{m\in\mathbf{N}_n} \sum_{j=1}^{L} r_{lj} p'^{(t)}_{mj} k_{nm}\right)}{\sum_{l=1}^{L} \exp\left(\sum_{m\in\mathbf{N}_n} \sum_{j=1}^{L} r_{lj} p'^{(t)}_{mj} k_{nm}\right)}$$

   Este proceso iterativo puede detenerse cuando la variación en las probabilidades de una iteración a la siguiente es lo suficientemente pequeña o después de un número máximo de iteraciones

6. Se vuelve al paso (3), repitiendo el proceso.

de compatibilidad para diferentes puntos de vista de la mano.

5.4. Experiencias con la asignación de correspondencias

Para comprobar las prestaciones de la asignación de correspondencias mediante KPCA en secuencias de imágenes de manos, se fijó un punto de vista genérico y se uso el software descrito en el apartado 6.2 en la página 158 para sintetizar 7 260 manos haciendo diferentes gestos con los dedos. En dicha cifra también se incluyen distintas posturas estáticas de la muñeca. Los gestos se muestrearon con entre 10 y 15 imágenes de la mano en las diferentes posturas de cada gesto, generándose un total de 90 750 imágenes. Adicionalmente se sintetizaron 100 000 manos en posturas escogidas aleatoriamente. De cada postura se generaron dos versiones: la primera utilizando una modelo que simulase la piel humana, para obtener una imagen sintética lo más fiel posible a la realidad, y la segunda donde cada extremidad aparece teñida de un color diferente, con la idea de que sirva como referencia para el etiquetado de los segmentos. De toda la galería se han extraído dos posturas al azar para ilustrar los resultados de las experiencias con el encaje de puntos. En la figura 5.1 se pueden observar las primeras 6 posturas de la mano del primer gesto extraído, que se denominará gesto «A», mientras que la figura 5.2 contiene imágenes para las mismas posturas pero con los diferentes elementos de la mano marcados con distintos colores. El mismo tipo de imágenes se puede observar en la figura 5.3 y en la figura 5.4 pero para lo que se denominará el gesto «B».

5.4.1. Reducción de la dimensionalidad

Las primeras experiencias realizadas consistieron en comprobar el comportamiento de KPCA a la hora de capturar información acerca de la forma de la mano. Para eso se tomaron posturas al azar y se extrajeron los puntos de la imagen que formaban parte de la mano, junto con las etiquetas obtenidas gracias a los distintos colores asignados a cada extremidad. De cada conjunto de puntos

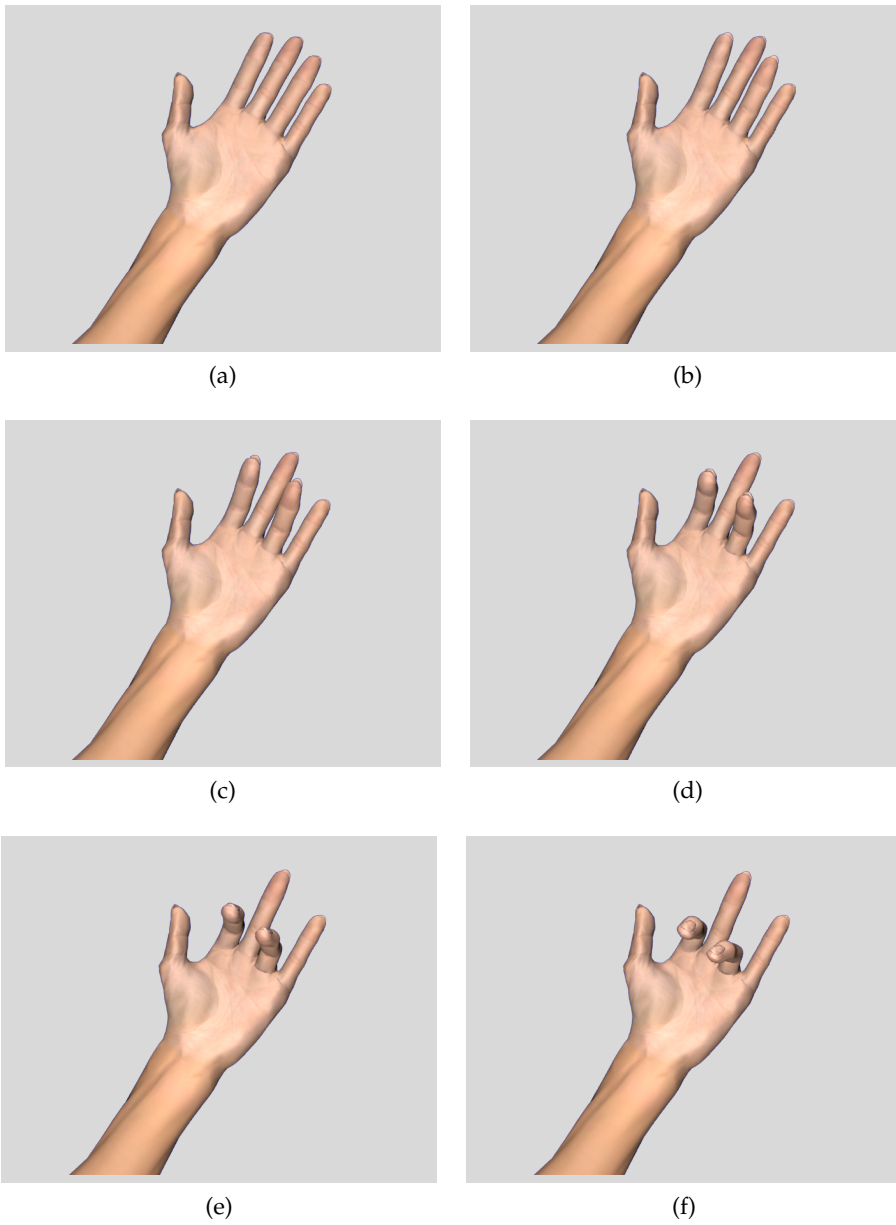


Figura 5.1. De izquierda a derecha y de arriba a abajo, las primeras 6 posturas del gesto «A» de la mano sintética con piel.

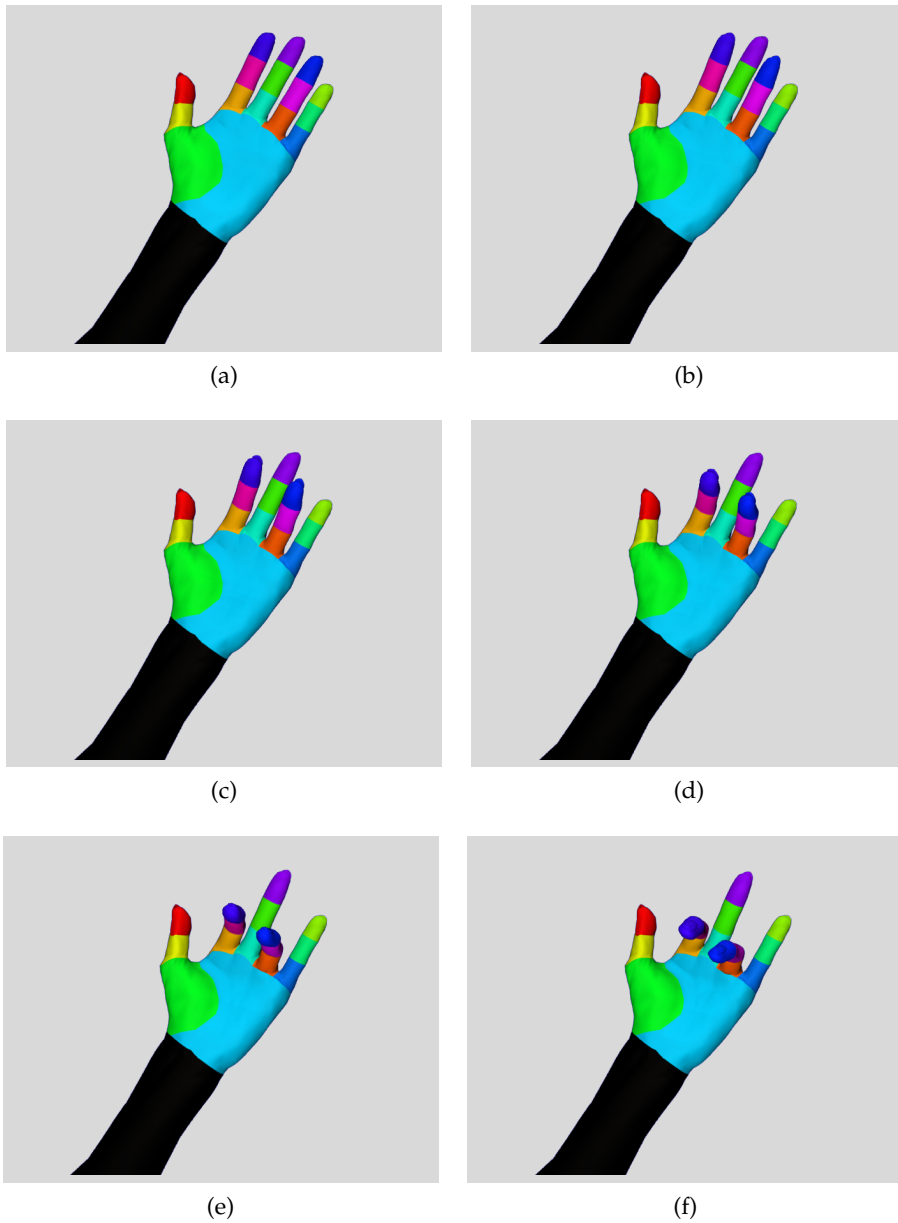
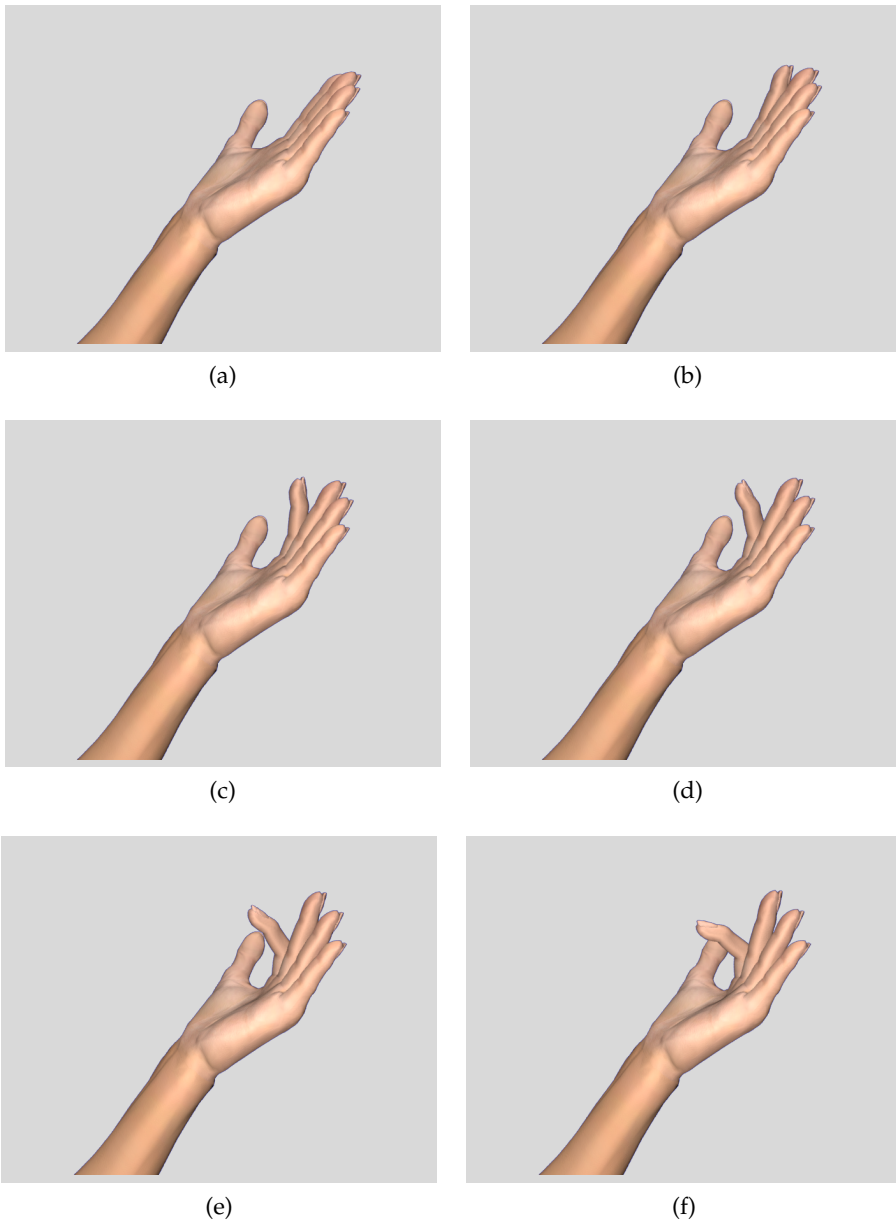


Figura 5.2. De izquierda a derecha y de arriba a abajo, las primeras 6 posturas del gesto «A» de la mano sintética con las extremidades etiquetadas en color.



(a) (b) (c) (d) (e) (f)

Figura 5.3. De izquierda a derecha y de arriba a abajo, las primeras 6 posturas del gesto «B» de la mano sintética con piel.

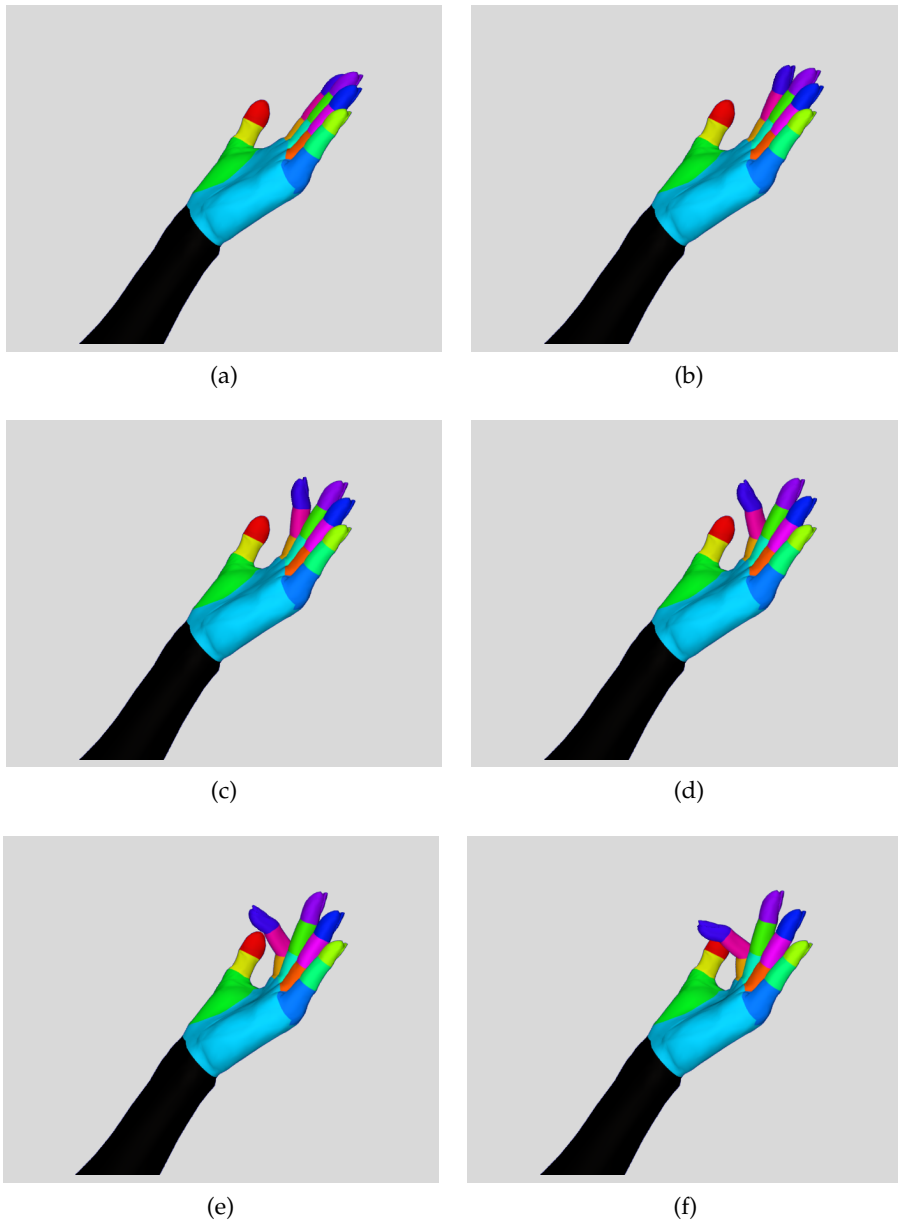


Figura 5.4. De izquierda a derecha y de arriba a abajo, las primeras 6 posturas del gesto «B» de la mano sintética con las extremidades etiquetadas en color.

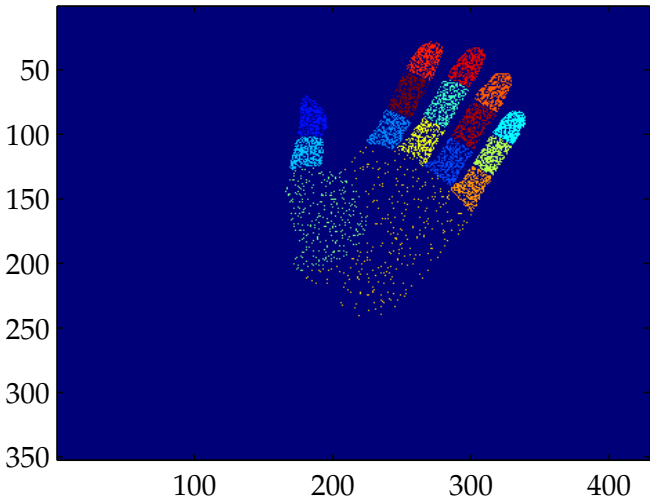


Figura 5.5. Ejemplo de la elección de puntos de la mano sintética de la figura 5.2(a).

se seleccionaron aleatoriamente 300 por etiqueta, con el objetivo de que el problema no fuera excesivamente grande. Es importante destacar que si el número total de puntos fuera N, la matriz $\mathbf{K}$ sería de $N \times N$. En la figura 5.5 se puede observar un ejemplo de los puntos seleccionados y etiquetados para la postura de la figura 5.2(a).

A cada conjunto de puntos se le hizo un KPCA para la funciones de kernel gaussianas (4.17), polinomial (4.18) y triangular (4.19) (véase el apartado 4.3 en la página 65), tanto asumiendo que la mano es un objeto rígido —sin utilizar la información de las etiquetas— como suponiendo que es un objeto articulado. Para este último caso la matriz de probabilidades $\mathbf{P}$ se construyó de forma que:

$$p_{nl} = \begin{cases} 1 & \text{si } l = l_n \\ 0 & \text{en cualquier otro caso} \end{cases} \tag{5.12}$$

donde l_n es la etiqueta del n-ésimo punto. En el caso de la función de kernel

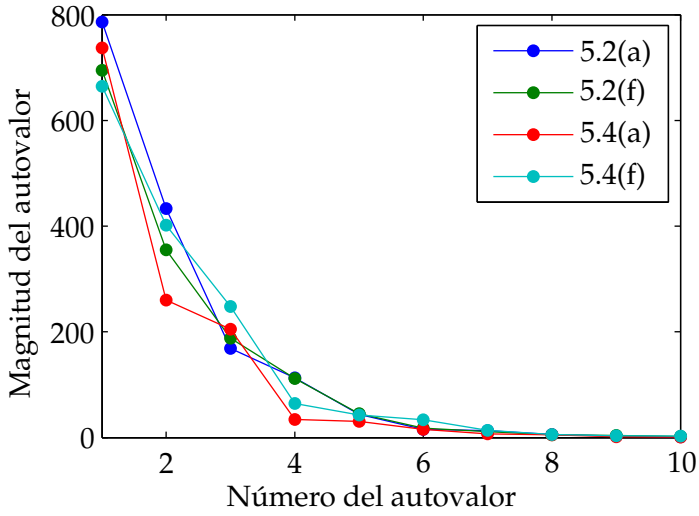


Figura 5.6. Autovalores para las posturas de las figuras 5.2(a), 5.2(f), 5.4(a) y 5.4(f) considerando la mano como un objeto rígido y usando una función de kernel gaussiana ($h = 2$).

gaussiana se fijó el valor del parámetro c mediante

$$c = h \sum_{n=1}^{N} \sum_{m=1}^{N} \frac{\|\boldsymbol{x}_n - \boldsymbol{x}_m\|^2}{N^2 - N} \tag{5.13}$$

donde h es una constante, tal y como proponen Wang y Hancock (2006). Esta expresión hace que el valor de c sea proporcional al valor medio de la distancia al cuadrado entre los puntos. Para ilustrar como el KPCA reduce la dimensionalidad, en la figura 5.6 se pueden observar los 10 autovalores de mayor magnitud para las figuras 5.2(a), 5.2(f), 5.4(a) y 5.4(f). Estos autovalores fueron obtenidos utilizando una función de kernel gaussiana (4.17) con $h = 2$ y suponiendo que la mano es un objeto rígido, por lo que no se utilizó la información proporcionada por las etiquetas. Por el contrario, para la figura 5.7 se obtuvieron los autovalores asumiendo que la mano es un objeto articulado. En ambos casos se puede observar que el KPCA captura la mayor parte de la variabilidad de los factores

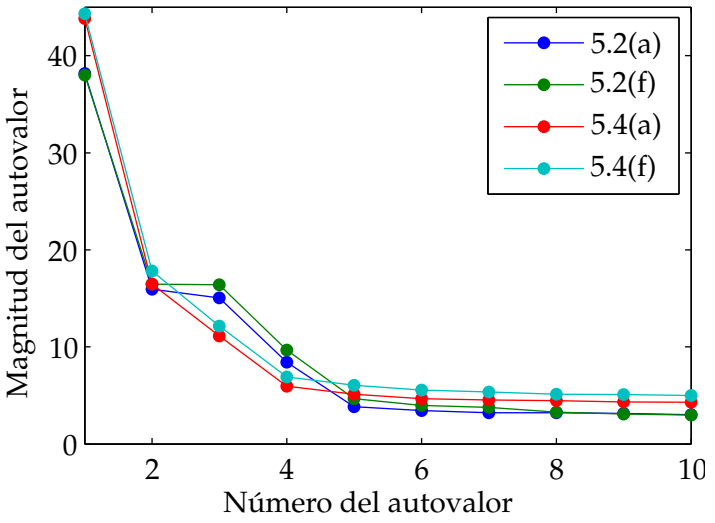


Figura 5.7. Autovalores para las posturas de las figuras 5.2(a), 5.2(f), 5.4(a) y 5.4(f) considerando la mano como un objeto articulado y usando una función de kernel gaussiana ($h = 2$).

subyacentes a las relaciones entre los puntos con 3 o 4 dimensiones, en función de si se trata de una postura del gesto «B» o del gesto «A» respectivamente. Obviamente, las variaciones globales de la postura de la mano tienen mayor peso que los cambios de posición en las extremidades. Por eso en ambas figuras es sencillo comprobar el efecto del movimiento en la articulación de la muñeca pero no el de los dedos al flexionarse.

En la tabla 5.1 se indican otros resultados acerca del número de dimensiones necesario[1] para capturar la información sobre las relaciones entre los puntos para diferentes posturas, con distintas funciones de kernel y con diferentes parámetros cada uno. Poco se puede decir a priori acerca de estos resultados. Cada posible parámetro para cada tipo de función de kernel genera un espacio de características distinto, donde el espacio de entrada define una variedad. En gene-

[1] La prueba de sedimentación (Cattell, 1966) fue el método utilizado para determinar el número de dimensiones, por ser uno de los más comunes. Sin embargo, es importante destacar que este método es susceptible de verse afectado por la subjetividad del investigador.

ral, no se puede predecir que función de kernel hará que las componentes principales extraídas posteriormente vayan a ser las «adecuadas» para la asignación de correspondencias. Eso sólo se puede intentar averiguar experimentalmente.

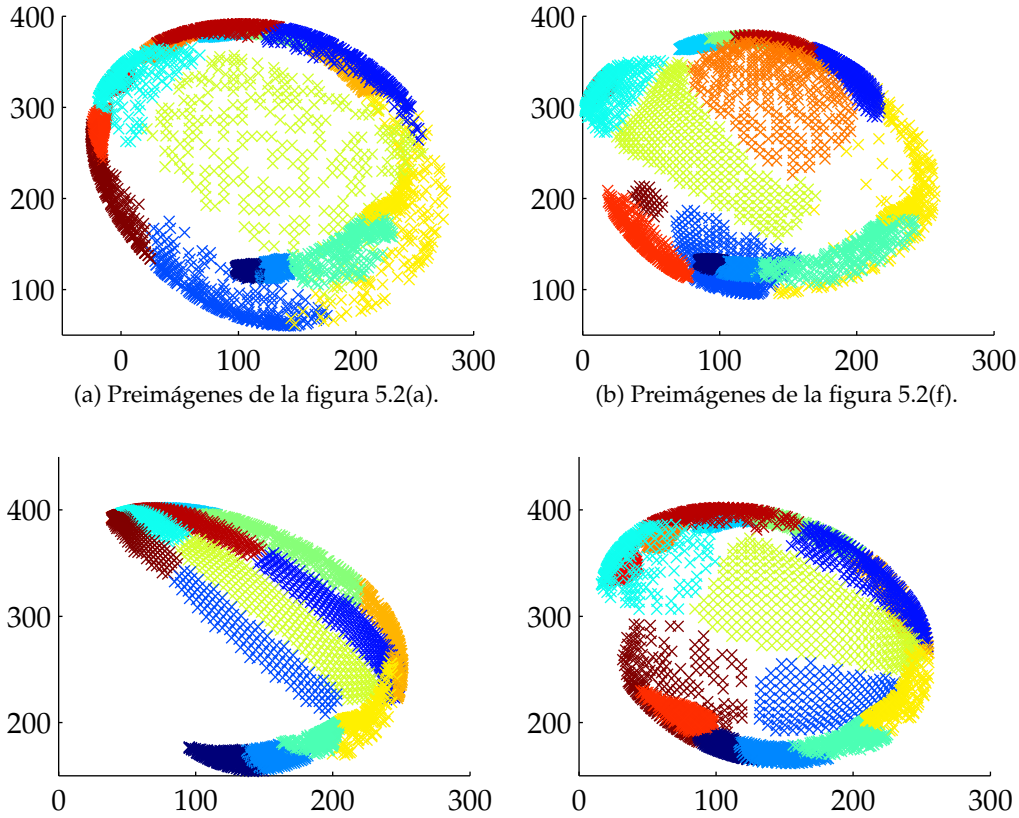
5.4.2. Estimación de la preimagen

Obtener la preimagen de los puntos puede ser de interés de cara a obtener algo de información acerca de lo que está ocurriendo en el espacio de características. Por ejemplo, la figura 5.8 muestra las preimágenes de los puntos de las figuras 5.2(a), 5.2(f), 5.4(a) y 5.4(f) suponiendo que la mano es un objeto rígido. Tras hacer un KPCA con función de kernel gaussiana ($h = 2$) para cada conjunto de puntos y proyectarlos en un subespacio de dimensión 4 —tal y como se recomienda en la tabla 5.1— se procedió a calcular la preimagen de cada uno de los puntos. Éstas se han coloreado en base a la etiqueta de su punto correspondiente antes del KPCA, con el objetivo de facilitar la visualización. En general, poco se puede decir acerca de las imágenes formadas por las preimágenes, de su relación con las imágenes originales y de como cambian las primeras al modificarse las últimas. Como se comentó en el apartado 4.5 en la página 71, un punto $P\Phi(\boldsymbol{x}_n)$ no tiene porqué estar dentro de la variedad del conjunto de datos, por lo que no está garantizado que tenga preimagen. Las técnicas descritas en el apartado 4.5 intentan estimar la preimagen de $P\Phi(\boldsymbol{x}_n)$ como aquel punto $\hat{\boldsymbol{x}}$ del espacio de entrada que al ser proyectado en el espacio de características $\mathbb{F}$ quede lo más cerca posible de $P\Phi(\boldsymbol{x}_n)$. Esto de ninguna manera garantiza parecido alguno entre la imagen original y la formada a partir de las preimágenes de los puntos. Sólo que los puntos y sus preimágenes correspondientes están próximas en el espacio de características.

El fenómeno opuesto ocurre con la función de kernel polinomial. En la figura 5.9 se pueden observar las preimágenes de los puntos de las figuras 5.2(f) y 5.4(f). En este caso las imágenes creadas a partir de las preimágenes tienen cierto parecido con las imágenes originales. Esto en cierta medida era de esperar puesto que las funciones de kernel polinomial tienden a capturar la información

Tabla 5.1. Número de dimensiones relevantes para diferentes posturas y funciones de kernel gaussianas (4.17), polinomial (4.18) y triangular (4.19).

Figura	Modelo	Función de kernel	Parámetros	Dimensiones
5.2(a)	rígido	gaussiana	$h = 1$	4
5.2(a)	rígido	gaussiana	$h = 5$	3
5.2(a)	rígido	gaussiana	$h = 10$	3
5.2(a)	rígido	polinomial	$d \in [1, 9]$	2
5.2(a)	rígido	triangular	$d = 0,5$	4
5.2(a)	rígido	triangular	$d = 1$	3
5.2(a)	rígido	triangular	$d = 1{,}5$	2
5.2(a)	articulado	gaussiana	$h \in [1, 10]$	4
5.2(a)	articulado	polinomial	$d = 1$	4
5.2(a)	articulado	polinomial	$d = 3$	3
5.2(a)	articulado	polinomial	$d = 5$	2
5.2(a)	articulado	polinomial	$d = 9$	1
5.2(a)	articulado	triangular	$d = 0{,}5$	2
5.2(a)	articulado	triangular	$d = 1$	4
5.2(a)	articulado	triangular	$d = 1{,}5$	4
5.4(a)	rígido	gaussiana	$h \in [1, 10]$	3
5.4(a)	rígido	polinomial	$d \in [1, 9]$	2
5.4(a)	rígido	triangular	$d \in [1/2, 3/2]$	3
5.4(a)	articulado	gaussiana	$h \in [1, 10]$	3
5.4(a)	articulado	polinomial	$d = 1$	3
5.4(a)	articulado	polinomial	$d = 2$	2
5.4(a)	articulado	polinomial	$d = 5$	3
5.4(a)	articulado	polinomial	$d = 9$	7
5.4(a)	articulado	triangular	$d = 0{,}5$	2
5.4(a)	articulado	triangular	$d = 1$	2
5.4(a)	articulado	triangular	$d = 1{,}5$	3



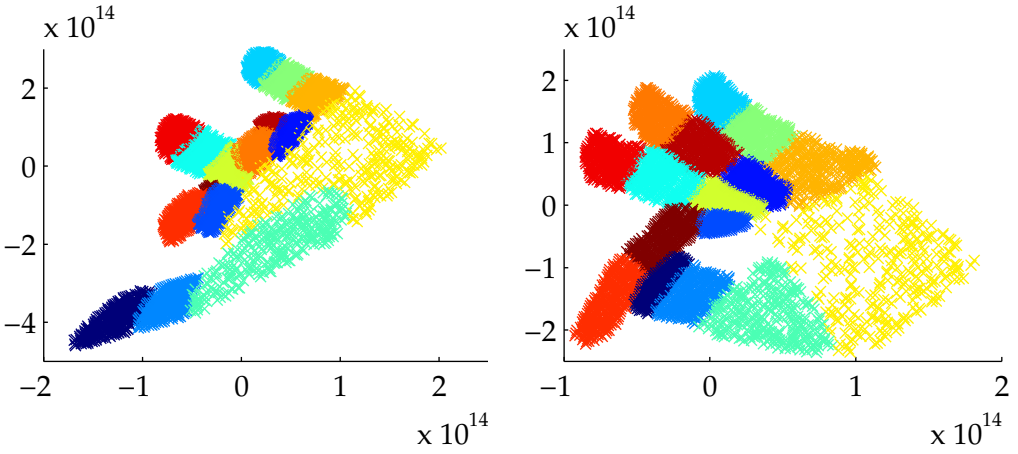
(a) Preimágenes de la figura 5.2(a).

(b) Preimágenes de la figura 5.2(f).

(c) Preimágenes de la figura 5.4(a).

(d) Preimágenes de la figura 5.4(f).

Figura 5.8. Preimágenes de los puntos de diversas posturas para un KPCA con función de kernel gaussiana ($h = 2$), subespacio de componentes principales de dimensión $R = 4$ y modelando la mano como un objeto rígido.

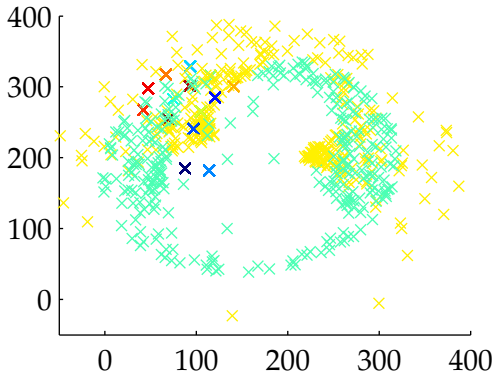


(a) Preimágenes de la figura 5.2(f). (b) Preimágenes de la figura 5.4(f).

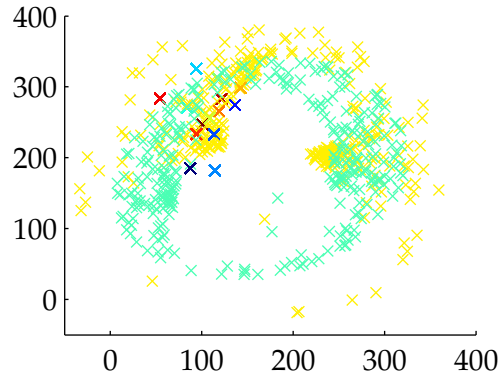
Figura 5.9. Preimágenes de los puntos de diversas posturas para un KPCA con función de kernel polinomial ($d = 2$), subespacio de componentes principales de dimensión $R = 2$ y considerando la mano como un objeto rígido.

subyacente a la direccionalidad de los datos, algo que desde el punto de vista del encaje puede ser muy interesante. Sin embargo, el subespacio de componentes principales adecuado para este tipo de función de kernel y este tipo de datos es 2, el mismo número de dimensiones que hay en el espacio de entrada. Por tanto, es muy posible que la función de kernel polinomial no sea capaz de extraer las factores no lineales subyacentes a las relaciones entre los puntos. Eso sería un problema, puesto que se partió de la hipótesis de que eso era necesario para asegurar una asignación de correspondencias robusta.

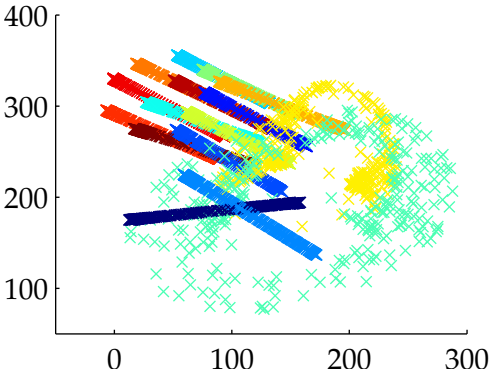
Respecto a considerar la mano como un objeto articulado, en la figura 5.10 se pueden observar las preimágenes de los puntos de las figuras 5.2(f) y 5.4(f) para un KPCA con función de kernel gaussiana ($h = 2$). Las figuras de la parte superior, 5.10(a) y 5.10(b), se obtuvieron después de que el KPCA finalmente proyectara los puntos de las imágenes originales en un subespacio de dimensión $R = 4$ —tal y como se recomienda en la tabla 5.1 en la página 115—. Observando dichas figuras es fácil comprobar que sólo se está preservando la información de



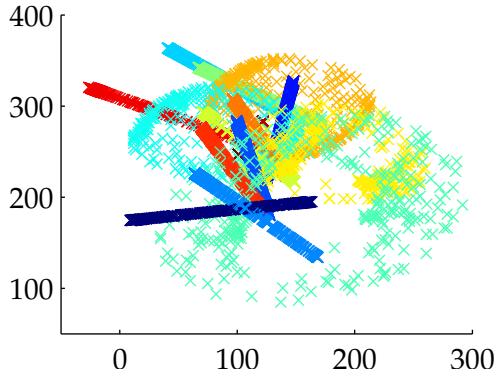
(a) Preimágenes de la figura 5.2(a) para la proyección con $R = 4$.



(b) Preimágenes de la figura 5.2(f) para la proyección con $R = 4$.



(c) Preimágenes de la figura 5.2(a) para la proyección con $R = 20$.



(d) Preimágenes de la figura 5.2(f) para la proyección con $R = 20$.

Figura 5.10. Preimágenes de los puntos de diversas posturas para un KPCA con función de kernel gaussiana ($h = 2$) y considerando la mano como un objeto articulado. Las de arriba son de puntos proyectados en un subespacio de componentes principales de dimensión $R = 4$, mientras que las de abajo son de las mismas imágenes pero proyectando los puntos en un subespacio de dimensión $R = 20$.

las regiones etiquetadas de mayor tamaño, que son las que componen la palma de la mano. Cada una de las otras regiones se colapsa en un único punto al obtener las preimágenes. Por el contrario, para figuras de la parte inferior, 5.10(c) y 5.10(d), el KPCA proyectó los puntos en un subespacio de dimensión $R = 20$. En ese caso los diferentes segmentos se colapsan en líneas con diversas orientaciones. Esto lleva a pensar que, pese a que la mayor parte de la variabilidad de los datos puede ser preservada con 4 dimensiones, desde el punto de vista de la asignación de correspondencias puede ser interesante conservar más dimensiones para disponer de más información.

Pese a que el cálculo de las preimágenes en general no tiene porqué devolver nada parecido a las imágenes originales, es importante tener en cuenta que esto depende de las funciones de kernel escogidas y de la estructura de los datos a analizar, entre muchos otros factores. Por ejemplo, Mika *et al.* (1999) utilizan con éxito el KPCA y la posterior estimación de la preimagen para eliminar ruido de diversas imágenes. En concreto, hacen que cada imagen de una base de datos de imágenes sea un punto del espacio de entrada. El KPCA se aplica sobre un subconjunto de dichos puntos para construir el subespacio de componentes principales. A partir de ese punto cualquier imagen de la base de datos puede ser proyectada en dicho subespacio, para posteriormente estimar su preimagen. El resultado es una versión reconstruida de la imagen original.

5.4.3. Correspondencia de puntos de objetos articulados sin relajación

Para comprobar el funcionamiento de la técnica de encaje de puntos en objetos articulados —descrita en el algoritmo 5.2 en la página 100— se procedió a probarla pero sin ponderar la correspondencia, utilizando la probabilidad de que el punto original y el asignado tengan la misma etiqueta. Es decir, en lugar de utilizar (5.4) se utilizó (5.2), la ecuación de la correspondencia para objetos rígidos. Adicionalmente, dos aspectos debieron ser tenidos en cuenta. El primero es que el signo de cada autovector no es único, puesto que cambiar el sentido no viola la condición de ortonormalidad de la base del subespacio de componentes prin-

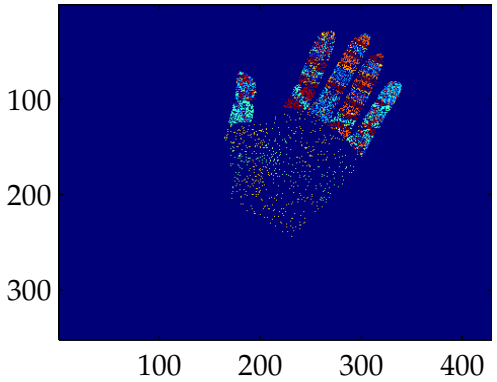
cipales. Sin embargo, es vital que los autovectores de ambos subespacios tengan la misma orientación para que los puntos proyectados puedan ser comparados utilizando (5.2) o (5.4). Por tanto es necesario añadir una etapa que corrija el signo de los autovectores, después de resolver el problema de autovalores (4.11) en la página 63 y antes de proyectar los puntos en el subespacio de componentes principales (4.13) en la página 64. Debido a la relación (4.6) entre los autovectores $\boldsymbol{v}_i$ de la matriz de covarianza de los datos y los autovectores $\boldsymbol{a}_i$ de la matriz de kernel K, para modificar el signo en los primeros basta con hacerlo en los segundos. Así, el signo del vector de coeficientes $\boldsymbol{a}'_i$, de los puntos de la segunda imagen, puede ser corregido usando como referencia los coeficientes $\boldsymbol{a}_i$, de la primera, según la regla

$$\boldsymbol{a}'_i = \begin{cases} \boldsymbol{a}'_i & \text{si } \boldsymbol{a}_i^T \boldsymbol{a}'_i > 0 \\ -\boldsymbol{a}'_i & \text{si } \boldsymbol{a}_i^T \boldsymbol{a}'_i < 0 \end{cases} \tag{5.14}$$

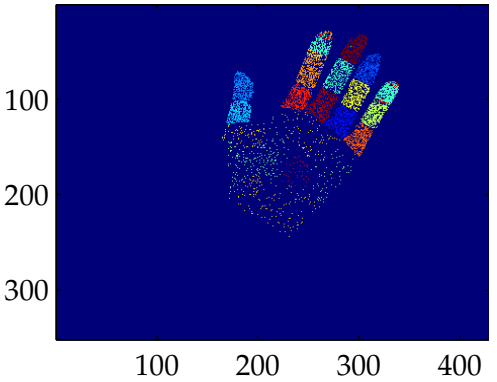
El segundo aspecto que debe ser tenido en cuenta sólo es relevante cuando se utiliza la función de kernel polinomial (4.18). Como se comentó en la página 66, este tipo de funciones de kernel no es invariante a cambios de escala, puesto que el producto escalar tampoco lo es. Para resolver este problema una posible solución es normalizar todos los autovalores por el de mayor magnitud para cada imagen, en el momento de normalizar los coeficientes $\boldsymbol{a}_i$ y $\boldsymbol{a}'_i$ (véase el apartado 4.2 en la página 64) antes de proyectar los puntos en el subespacio de componentes principales (4.13).

Correspondencia sin ponderación basada en etiquetas

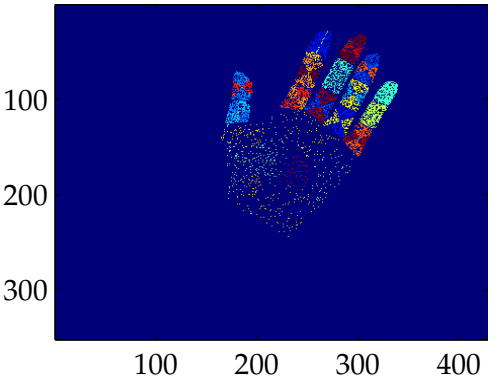
La figura 5.11 muestra el resultado de la asignación de correspondencias, en las condiciones descritas en el apartado 5.4.3, utilizando una función de kernel gaussiana, entre las imágenes de las figuras 5.2(a) y 5.2(b) y para distintas dimensiones del subespacio de componentes principales. Tal y como parecía intuirse en el análisis de las preimágenes, el número de dimensiones recomendado en la



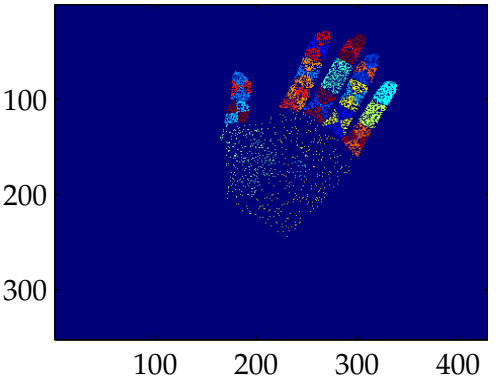
(a) Correspondencia con función de kernel gaussiana ($h = 2$) y $R = 4$.



(b) Correspondencia con función de kernel gaussiana ($h = 2$) y $R = 20$.



(c) Correspondencia con función de kernel gaussiana ($h = 2$) y $R = 40$.

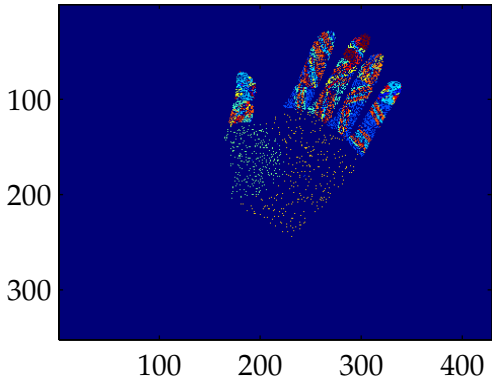


(d) Correspondencia con función de kernel gaussiana ($h = 10$) y $R = 40$.

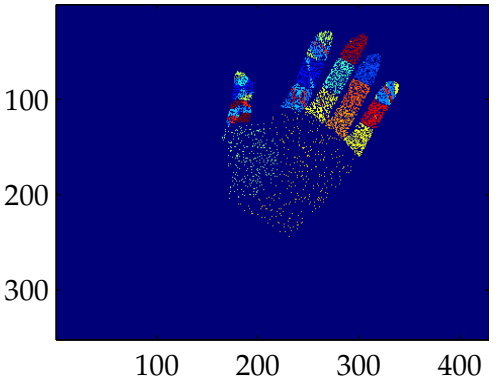
Figura 5.11. Correspondencia de objetos articulados entre las imágenes de las figuras 5.2(a) y 5.2(b), utilizando funciones de kernel gaussianas y distintas dimensiones para el subespacio de componentes principales.

tabla 5.1 no es suficiente para un correcto encaje de puntos. En la figura 5.11(a) se muestra el resultado cuando el subespacio de componentes principales tiene dimensión $R = 4$ —tal y como recomienda el análisis de autovalores—. Sin embargo, los resultados no se corresponden con lo esperado. Según va aumentado el número de dimensiones preservadas en el KPCA mejoran notablemente los resultados del encaje, hasta el punto de que en la figura 5.11(c) se puede observar cierto patrón. En esa misma imagen se puede ver que se comenten errores en el centro de la palma. En la figura 5.11(d) esos errores se corrigen parcialmente al aumentar el parámetro h. Éste controla el ancho de la campana de la función de kernel gaussiana. Por tanto, parece lógico que para capturar la información de los segmentos de mayor tamaño haya que aumentar h, con el objetivo de que dicha campana cubra la mayor parte de los puntos del segmento. El mismo comportamiento se observa para las posturas del gesto «B» y, en general, para todas las posturas generadas sintéticamente. En la figura 5.12 se puede observar un resultado similar, sólo que obtenido utilizando una función de kernel polinomial de grado $d = 2$, y lo mismo ocurriría si hiciera la prueba con la función de kernel triangular.

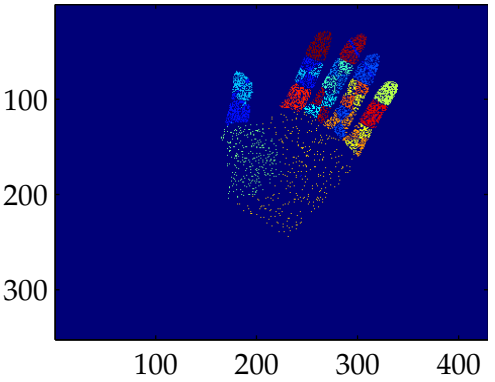
Incluso en los mejores casos la asignación no es correcta, siendo lo más interesante el patrón que exhibe. En condiciones normales cabría esperar errores en las correspondencias en el sentido, por ejemplo, de puntos mal etiquetados en la frontera de los segmentos. Sin embargo, en las figuras 5.11(d) y 5.12(d) se observa que en ocasiones a segmentos completos se les asigna la misma etiqueta, aunque esta no sea la correcta, mientras que en otros casos los segmentos son seccionados en cuatro y las etiquetas asignadas en parejas de dos segmentos. Curiosamente, esta técnica de encaje de correspondencias parece que sólo funciona correctamente en los segmentos que forman la palma de la mano, que además son los de mayor tamaño en extensión dentro de la imagen. Usando las técnicas descritas en el apartado 4.6 se descubre que los subespacios de componentes principales para ambas imágenes, en general, no tienen porqué encajar. Y eso pese a que para estas pruebas se partió de la hipótesis de que se conocía con exactitud el etiquetado en ambas imágenes. En concreto, para el caso de la



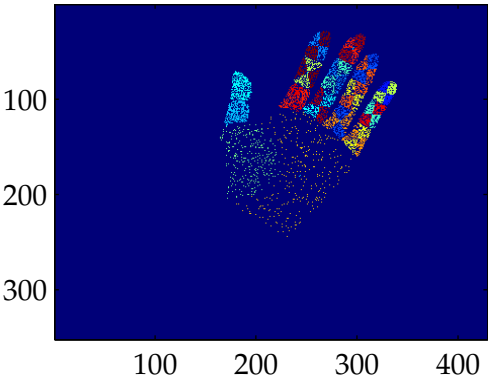
(a) Correspondencia con función de kernel polinomial ($d = 2$) y $R = 3$.



(b) Correspondencia con función de kernel polinomial ($d = 2$) y $R = 10$.



(c) Correspondencia con función de kernel polinomial ($d = 2$) y $R = 20$.



(d) Correspondencia con función de kernel polinomial ($d = 2$) y $R = 30$.

Figura 5.12. Correspondencia de objetos articulados entre las imágenes de las figuras 5.2(a) y 5.2(b), utilizando una función de kernel polinomial y distintas dimensiones para el subespacio de componentes principales.

figura 5.11(d) y utilizando la definición de Wang *et al.* (2006), desarrollada en el apartado 4.6.1 en la página 82, la distancia entre ambos subespacios es de 0,3943, mientras que para el caso de la figura 5.12(d) la distancia es de 0,3264.

La distancia entre subespacios propuesta por Wang *et al.* (2006) se calcula mediante la expresión (4.50) en la página 83, donde la parte entre paréntesis, compuesta por los dos sumatorios interiores, es una matriz $\mathbf{Q}$ de $R \times R$ tal que $q_{ij} = (\boldsymbol{u}_i^T \boldsymbol{v}_j)^2$, siendo $\boldsymbol{u}_i$ y $\boldsymbol{v}_j$ los autovectores del subespacio de componentes principales para la primera y la segunda imagen, respectivamente. Es decir, que esa matriz, que también tiene un papel central en el cálculo de los ángulos principales (4.52), esta compuesta por el resultado de hacer el producto de cada autovector de uno de los subespacios por los autovectores del otro subespacio. Si ambos subespacios fueran el mismo, la matriz $\mathbf{Q}$ sería la matriz identidad $\mathbf{I}$. Pero, por ejemplo, para el caso de la figura 5.11(d) la matriz $\mathbf{Q}$ es:

$$\mathbf{Q} = \begin{bmatrix}
0{,}99 & 0{,}08 & 0{,}00 & 0{,}00 & 0{,}00 & 0{,}00 & 0{,}00 & 0{,}00 & \cdots & 0{,}00 \\
0{,}08 & 0{,}99 & 0{,}00 & 0{,}00 & 0{,}00 & 0{,}00 & 0{,}00 & 0{,}00 & \cdots & 0{,}00 \\
0{,}00 & 0{,}00 & 0{,}99 & 0{,}14 & 0{,}00 & 0{,}00 & 0{,}00 & 0{,}00 & \cdots & 0{,}00 \\
0{,}00 & 0{,}00 & 0{,}14 & 0{,}99 & 0{,}00 & 0{,}00 & 0{,}00 & 0{,}00 & \cdots & 0{,}00 \\
0{,}00 & 0{,}00 & 0{,}00 & 0{,}00 & 1{,}00 & 0{,}00 & 0{,}00 & 0{,}00 & \cdots & 0{,}00 \\
0{,}00 & 0{,}00 & 0{,}00 & 0{,}00 & 0{,}00 & 0{,}00 & 0{,}00 & 0{,}00 & \cdots & 0{,}02 \\
0{,}00 & 0{,}00 & 0{,}00 & 0{,}00 & 0{,}00 & 0{,}00 & 0{,}00 & 0{,}00 & \cdots & 0{,}00 \\
0{,}00 & 0{,}00 & 0{,}00 & 0{,}00 & 0{,}00 & 0{,}00 & 0{,}00 & 0{,}00 & \cdots & 0{,}00 \\
0{,}00 & 0{,}00 & 0{,}00 & 0{,}00 & 0{,}00 & 0{,}99 & 0{,}00 & 0{,}00 & \cdots & 0{,}00 \\
0{,}00 & 0{,}00 & 0{,}00 & 0{,}00 & 0{,}00 & 0{,}00 & 1{,}00 & 0{,}00 & \cdots & 0{,}00 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0{,}00 & 0{,}00 & 0{,}00 & 0{,}00 & 0{,}02 & 0{,}00 & 0{,}00 & 0{,}00 & \cdots & 0{,}98
\end{bmatrix}$$

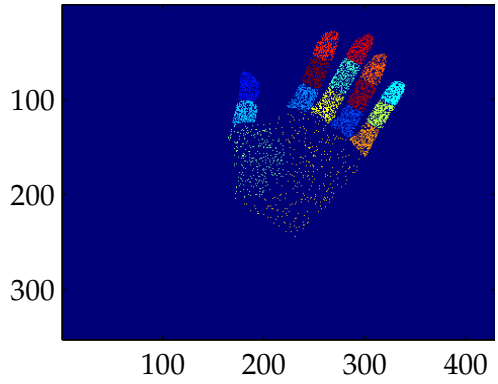
que obviamente no es una la matriz $\mathbf{I}$. Al margen de los pocos elementos que ni son próximos a 0 ni a 1, parece que los autovectores para ambas imágenes son parecidos, pero que no siempre generan la misma componente de los conjuntos

de puntos proyectados $\{\boldsymbol{y}_n\}$ e $\{\boldsymbol{y}'_m\}$, en los subespacios de componentes principales. Un ejemplo de eso se puede observar en el elemento q_{96} de la matriz $\mathbf{Q}$. El noveno autovector genera la novena componente de los datos proyectos en el primer subespacio, pero el autovector que más se le parece en el segundo subespacio es el sexto. Este fenómeno se repite a lo largo y ancho de toda la matriz $\mathbf{Q}$ y no sólo para este caso concreto, sino también para diferentes pares de imágenes y tipos de funciones de kernel. En todo caso es importante tener en cuenta que, puesto que el subespacio de componentes principales se obtiene calculando los R autovectores más importantes del espacio de características, y dado que parece que entre un espacio y otro éstos pueden haber permutado, no hay garantías de que ambos subespacios tengan la misma base. Es decir, que en la base de uno de ellos puede haber un autovector que no fue incluido en la base del otro.

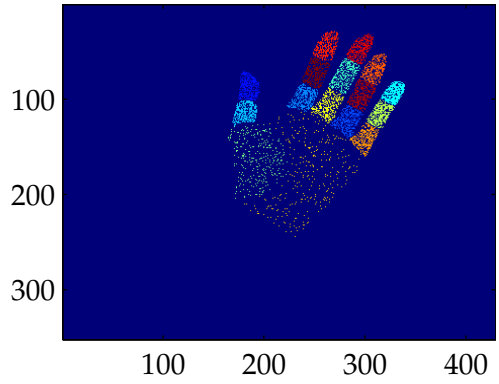
Una posible solución a este problema sería utilizar la divergencia de Kullback-Leibler (véase el apartado 4.6.3 en la página 85) para etiquetar los puntos proyectados en su subespacio de componentes principales, aunque dichos subespacios sean distintos. En principio, sería posible comparar la distribución de puntos proyectados de la segunda imagen que tienen una etiqueta dada con las distribuciones de puntos proyectados de la primera imagen para cada etiqueta posible. Aquella pareja de distribuciones con menor divergencia de Kullback-Leibler indicaría la etiqueta más probable para los puntos de la segunda imagen. Si bien es cierto que con este procedimiento a toda una región se le asigna la misma etiqueta, el resultado sería muy similar a lo visto previamente. Para calcular la divergencia Kullback-Leibler hay que estimar la matriz de covarianza a partir de los puntos proyectados. Dicha matriz se descompone en los mismos autovectores que generan el subespacio de componentes principales. Por tanto, esta posible solución es sensible al mismo problema que afecta al algoritmo de asignación de correspondencias original.

Correspondencia con ponderación basada en etiquetas

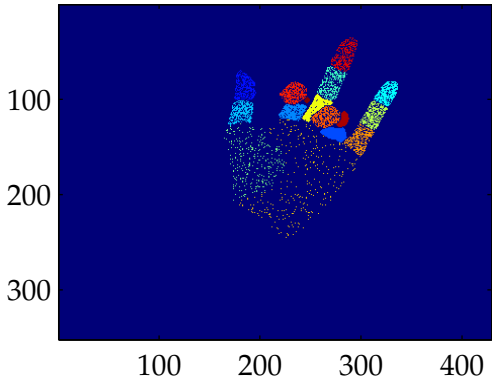
La figura 5.13 muestra el resultado de la correspondencia utilizando el algorit-



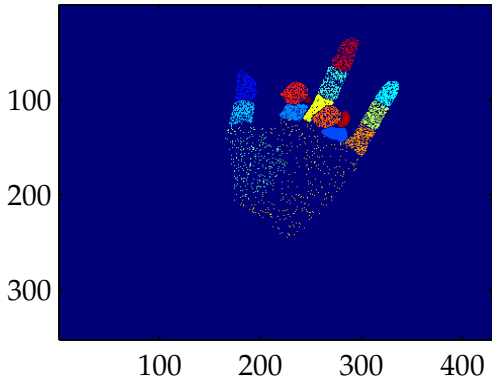
(a) Correspondencia entre las figuras 5.2(a) y 5.2(b) con función de kernel gaussiana y $R = 4$.



(b) Correspondencia entre las figuras 5.2(a) y 5.2(b) con función de kernel gaussiana y $R = 40$.



(c) Correspondencia entre las figuras 5.2(a) y 5.2(f) con función de kernel gaussiana y $R = 4$.



(d) Correspondencia entre las figuras 5.2(a) y 5.2(f) con función de kernel gaussiana y $R = 40$.

Figura 5.13. Arriba, correspondencia de objetos articulados entre las imágenes de las figuras 5.2(a) y 5.2(b), utilizando una función de kernel gaussiana ($h = 2$) y distintas dimensiones para el subespacio de componentes principales. Abajo, correspondencia en las mismas condiciones de puntos de objetos articulados entre las imágenes de las figuras 5.2(a) y 5.2(f).

mo 5.2 sin modificación alguna. Es decir, teniendo en cuenta la probabilidad de que los puntos en ambas imágenes tengan la misma etiqueta. Las figuras 5.13(a) y 5.13(b) son el resultado del encaje entre las figuras 5.2(a) y 5.2(b). La primera, cuando el subespacio de componentes principales es de dimensión $R = 4$, y la segunda, cuando es de $R = 40$. Las figuras de la parte inferior se han obtenido de forma similar pero para el caso de intentar hacer la correspondencia de las figuras 5.2(a) y 5.2(f).

A diferencia de lo que se observa en la figura 5.11, parece que el algoritmo funciona perfectamente. Curiosamente lo hace incluso con $R = 4$, cuando antes parecía obvio que si se recortaba mucho el subespacio de componentes principales se perdía información acerca de los segmentos rígidos de menor tamaño. En realidad, utilizando (5.4) para la asignación de correspondencias ésta se muestra muy robusta con la función de kernel gaussiana, proporcionando una asignación perfecta para diferentes valores para el parámetro h, dimensiones del subespacio de componentes principales e imágenes. Lamentablemente, no se aprecia el mismo resultado ni utilizando la función de kernel polinomial ni la triangular, pues con ambos a todos los puntos siempre se les acaba asignado la misma clase. En todo caso (5.4) sólo se distingue de (5.2) en el uso de la información conocida acerca del etiquetado de los puntos. Por tanto, es de esperar que en ambos casos los subespacios generados por los conjuntos de puntos de ambas imágenes no sean los mismos. Es decir, que la asignación de correspondencias realmente no está utilizando la información adecuada. Es muy posible que los buenos resultados se deban a que para comprobar las prestaciones de esta técnica se ha partido de suponer que las etiquetas son perfectamente conocidas para ambas imágenes. Cada punto tiene probabilidad 1,0 sólo para una de las etiquetas y 0,0 para las otras, haciendo que (5.4) funcione. Sin embargo, no parece probable que estos buenos resultados se mantengan al introducir la relajación. Cuando eso ocurra el problema será más real, en el sentido de que las etiquetas de la segunda imagen realmente no se conocerán, por lo que habrá que hacer algún tipo de suposición sobre el etiquetado inicial que posteriormente la relajación deberá refinar. Puesto que ese etiquetado inicial en nada tiene que parecerse al real, es muy posible

que entonces el algoritmo se muestre mucho menos robusto por el problema ya comentado con los subespacios.

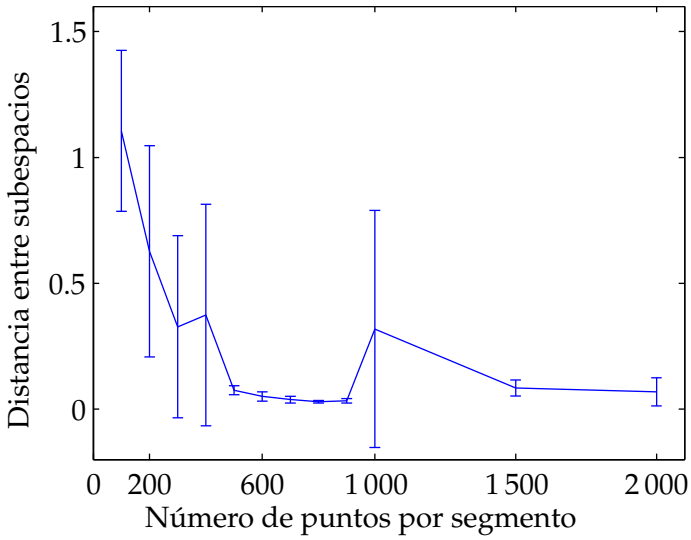
5.4.4. Efecto de la elección de los puntos en la correspondencia

Pese a los buenos resultado en el apartado 5.4.3, todo parece indicar que los subespacios generados por los conjuntos de puntos de las dos imágenes que intervienen en el encaje no tienen porqué ser ni tan siquiera parecidos, tal y como revelaron las experiencias comentadas en el apartado 5.4.3, incluso entre imágenes relativamente parecidas como son las de las figuras 5.2(a) y 5.2(b).

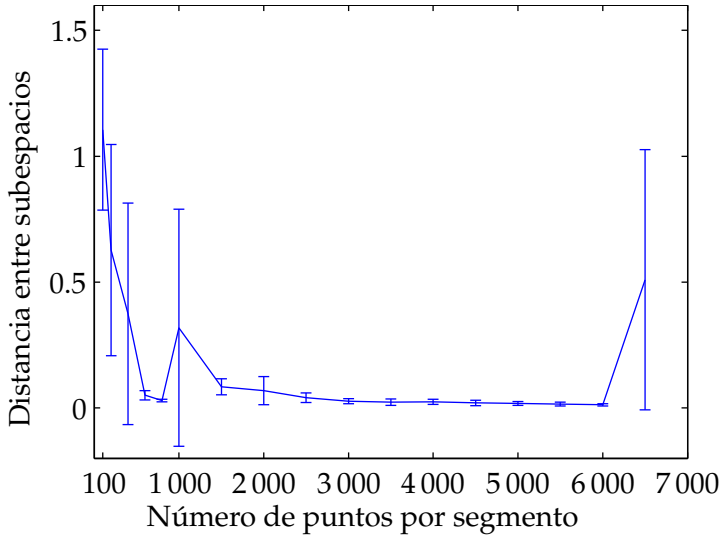
En principio, detrás de este problema podría estar el alineamiento entre los puntos de las dos imágenes. Resulta muy sencillo generar dos matrices **A** y **B** de manera que ambas sean iguales, excepto por que a la segunda se le permuta una columna respecto a la primera. Si se resolviese el problema de los autovalores para ambas matrices, se comprobaría que los autovectores no son invariantes a las permutaciones, aunque los autovalores si lo son. Es, por tanto, de esperar que si los puntos en ambas imágenes no se escogen en el mismo orden, las filas y las columnas de la matriz de kernel **K** permuten, haciendo que los subespacios generados por dichos conjuntos de puntos se viesen afectados. Estudiando la literatura se puede ver que es común someter los puntos de interés a un análisis de Procrustes, con el objetivo de permutarlos de forma que se preserve cierto orden en la elección de los puntos entre las distintas imágenes (Rathi *et al.*, 2006). Pero eso se hace porque todos los puntos de una misma imagen se vectorizan para configurar un espacio de entrada formado por el conjunto de todas las imágenes. En ese caso es importante el orden en el que se escogen lo puntos, puesto que eso afecta directamente a la posición de cada uno en el vector y, por tanto, a la medida de similaridad de la función de kernel. Sin embargo, en el presente caso es sencillo comprobar que la permutación de los puntos no afecta al resultado del KPCA. Basta con realizar un análisis de Procrustes de los conjuntos de puntos de ambas imágenes, permutando los de la segunda para minimizar el promedio de la distancia con los puntos de la primera $\frac{1}{N}\sum_{n=1}^{N}\|\boldsymbol{x}_n - \boldsymbol{x}'_n\|$, antes

de aplicar la técnica de encaje de correspondencias. El resultado será el mismo que si el análisis de Procrustes no se hubiera hecho, independientemente de la función de kernel y de los conjuntos de puntos escogidos.

Aunque el problema no se resuelva alineando los puntos entre ambas imágenes, es posible que algo tenga que ver la elección que de ellos se hace. Para comprobarlo se puede tomar una figura, escoger aleatoriamente varios conjuntos de puntos y calcular la distancia (4.50) entre los subespacio de componentes principales que éstos crean al utiliza el KPCA. En la figura 5.14 se puede observar la media de dicha distancia y el error estándar para diferentes cantidades de puntos escogidos por segmento, utilizando una función de kernel gaussiana ($h = 5$) y la imagen de la figura 5.2(a). Mientras que en la figura 5.15 se pueden observar los resultados de la misma prueba pero para una función de kernel poligonal ($d = 2$). En ambos casos se aprecia que con número bajo de puntos por segmento, la distancia entre los subespacios puede variar sustancialmente. Todo depende de la «fortuna» que se tenga al escoger dichos puntos. Algunos conjuntos de puntos generan el mismo subespacio de componentes principales, mientras que muchos otros no. Conforme el número de puntos por segmento aumenta, mas parecidos son los subespacios y menos varía la distancia entre ellos. En el caso de la función de kernel gaussiana el primer mínimo está en los 800 puntos por segmento, aunque entre utilizar 600 o 900 puntos no parece haber gran diferencia. De la misma manera, para la función de kernel polinomial la curva de la media se vuelve prácticamente plana en torno a 600 puntos. Es importante destacar que todos los segmentos, a excepción de los dos que forman la palma de la mano —los de mayor tamaño— tienen algo más de 600 puntos o menos. Por tanto, lo que ocurre es que los subespacios de componentes principales que se generan a través del KPCA dependen de los puntos concretos escogidos para generarlos. Sin embargo, al principio se había supuesto que el subespacio de componentes principales no cambiaría sustancialmente con la elección de los puntos, si ésta era lo suficientemente grande como para capturar convenientemente la forma de los segmentos, puesto que en ese caso el KPCA podría extraer la información acerca de las relaciones que hay entre los puntos de dichos segmentos. Según

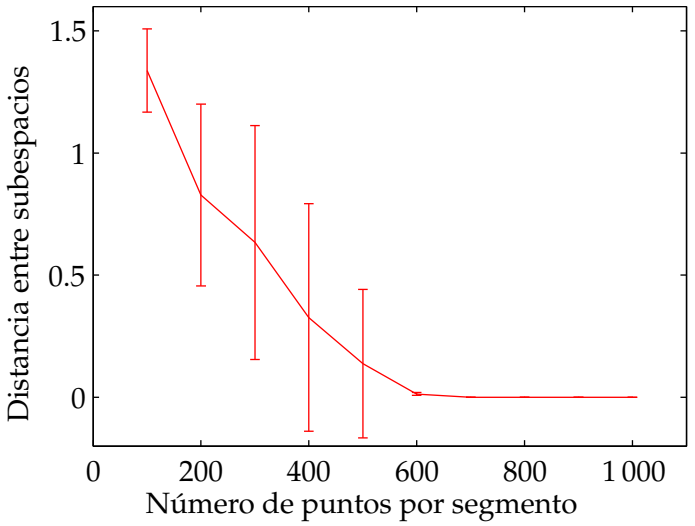


(a) Media para entre 100 y 2 000 puntos por segmento

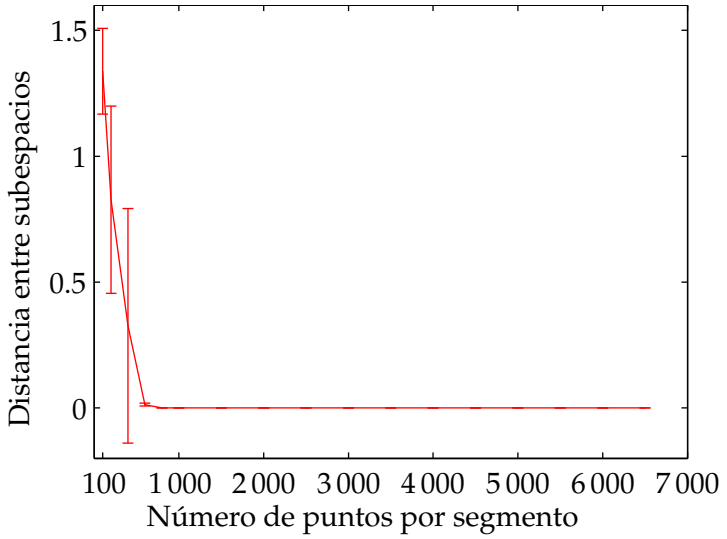


(b) Media para entre 100 y 7 000 puntos por segmento

Figura 5.14. Media de la distancia entre los subespacios generados por el KPCA, con función de kernel gaussiana ($h = 10$), para dos conjuntos de puntos distintos escogidos aleatoriamente sobre la imagen de la figura 5.2(a).



(a) Media para entre 100 y 1 000 puntos por segmento



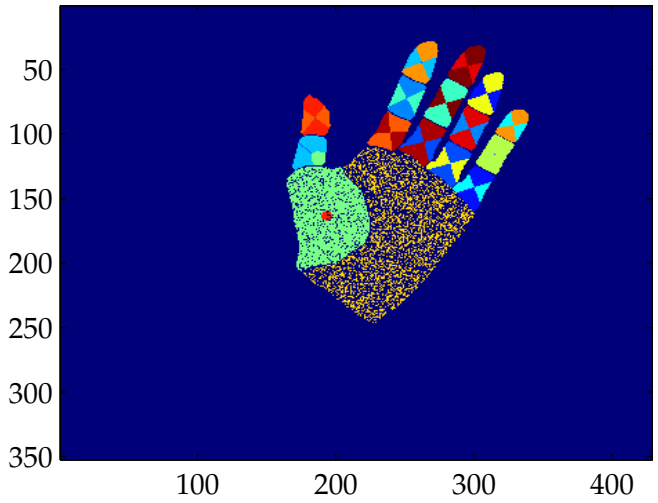
(b) Media para entre 100 y 7 000 puntos por segmento

Figura 5.15. Media de la distancia entre los subespacios generados por el KPCA, con función de kernel polinomial ($d = 2$), para dos conjuntos de puntos distintos escogidos aleatoriamente sobre la imagen de la figura 5.2(a).

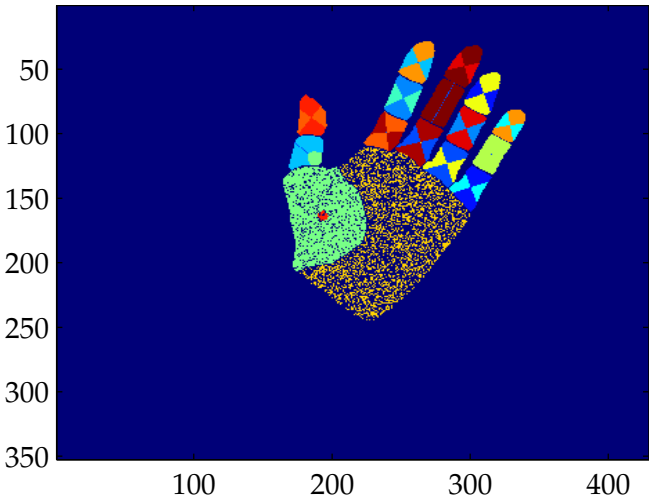
aumenta el número de puntos se reducen las posibilidades de elegir, de manera que en torno a los 600 puntos —en el caso de la figura la figura 5.2(a)— se estarían escogiendo todos los puntos posibles de los segmentos de menor tamaño. Con 1000 puntos o más realmente sólo se estaría escogiendo aleatoriamente los puntos de los dos segmentos que forman la palma de la mano, mientras que para las otras regiones se estarían escogiendo todos. Este mismo fenómeno ocurre para todas las imágenes generadas de posturas de la mano. Lo que varía entre una postura y otra es el punto dónde la distancia ya no se reduce significativamente, pues éste depende del tamaño de los diferentes segmentos. Por tanto, es necesario escoger un número de puntos adecuado, que no debe ser demasiado alto por los requerimientos de memoria del KPCA. Lo cierto es que la distancia converge muy lentamente hacia cero cuando se incrementa el número de puntos. La figura 5.16 muestra el resultado del encaje de correspondencias para 3 000 puntos por segmento y en las mismas circunstancias que se utilizaron para obtener las figuras 5.11(d) y 5.12(d), que son las que mostraban, hasta el momento, los mejores resultados cuando las correspondencias se asignan mediante (5.2) en la página 99. Como se puede apreciar, el problema no ha sido resuelto aumentando el número de puntos. Si se examinara la matriz $\mathbf{Q}$, se podría comprobar que los autovectores de ambos subespacios encajan un poco mejor, en el sentido de se mantiene diagonal para más dimensiones correspondientes a los mayores autovalores, pero no lo suficiente para que la asignación de correspondencias mejore sustancialmente. El mayor inconveniente es que aumentar aun más el número de puntos es muy complejo porque se multiplican los requerimientos de memoria.

5.4.5. Correspondencia de objetos articulados mediante KCCA

Puesto que el problema es que los subespacios de componentes principales para dos conjuntos de puntos, entre los que se quiere hacer encaje de correspondencias, son diferentes, se podría pensar en utilizar KCCA. Tal y como fue comentado en el apartado 4.7 en la página 90, el KCCA permite obtener un subes-



(a) Correspondencia con 3 000 puntos por segmento y función de kernel gaussiana ($h = 5$).



(b) Correspondencia con 3 000 puntos por segmento y función de kernel polinomial ($d = 2$).

Figura 5.16. Correspondencia de objetos articulados entre las imágenes de las figuras 5.2(a) y 5.2(b), utilizando diferentes funciones de kernel y seleccionado 3 000 puntos por segmento.

Algoritmo 5.4. Correspondencia de puntos en objetos articulados con KCCA.

1. Se crean las matrices de similaridad $\mathbf{K}$ y $\mathbf{K}'$ para los conjuntos de puntos $\{\boldsymbol{x}_n\}$ y $\{\boldsymbol{x}'_m\}$ de cada imagen mediante cualquier medida de similaridad que se defina con una función semidefinida positiva (véase el apartado 4.3 en la página 65).

2. Las matrices $\mathbf{K}$ y $\mathbf{K}'$ son ponderadas utilizando la probabilidad de que cada par de puntos tengan la misma etiqueta

$$\tilde{k}_{nm} = \sum_{l=1}^{L} p_{nl} p_{ml} k_{nm}$$

   suponiendo que se conocen las matrices de probabilidad $\mathbf{P}$ y $\mathbf{P}'$.

3. A ambas matrices a la vez se les aplica el procedimiento del KCCA con centrado por clases, resumido en el algoritmo 4.4.

4. Se hallan las correspondencias de cada punto proyectado $\boldsymbol{y}_n$ resolviendo (5.4)

$$\boldsymbol{y}'_m = \arg\min_{\boldsymbol{y}'_m} \sum_{l=1}^{L} p_{nl} p'_{ml} \exp(-\|\boldsymbol{y}_n - \boldsymbol{y}'_m\|^2/c) \qquad \text{para } n = 1, \ldots, N$$

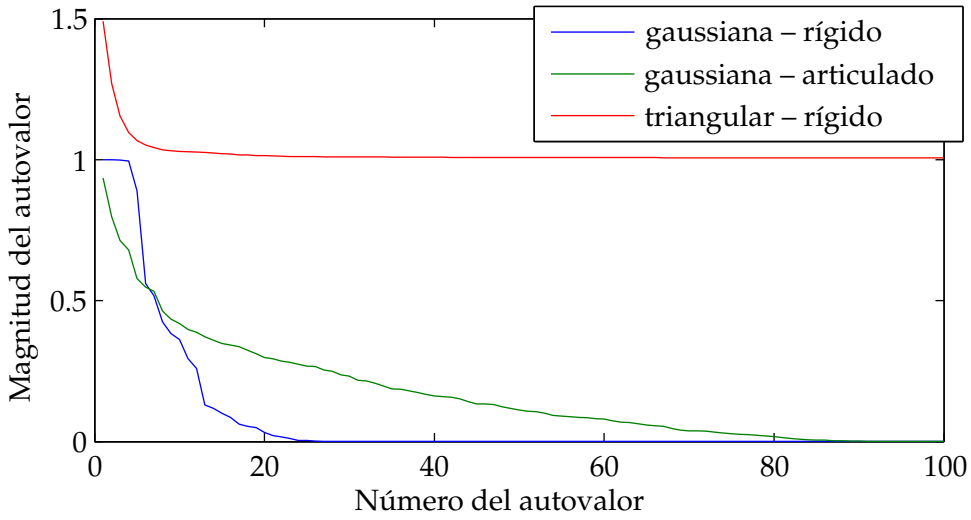


Figura 5.17. Análisis de autovalores generalizados para las posturas de las figuras 5.2(a) y 5.2(b), utilizando KCCA y distintas funciones de kernel.

pacio donde se maximiza la correlación de los puntos proyectados en el espacio de características $\mathbb{F}$ mediante la transformación no lineal $\Phi(x)$. El procedimiento sería el mismo que el seguido por el algoritmo 5.1 en la página 98 —para objetos rígidos— o por 5.2 en la página 100 —para objetos articulados— con la salvedad de realizar un KCCA de los conjuntos de puntos de las dos imágenes a la vez, en lugar de aplicar un KPCA a cada uno por separado. Concretamente, en el algoritmo 5.4 se resume el procedimiento para las asignación de correspondencias de objetos articulados.

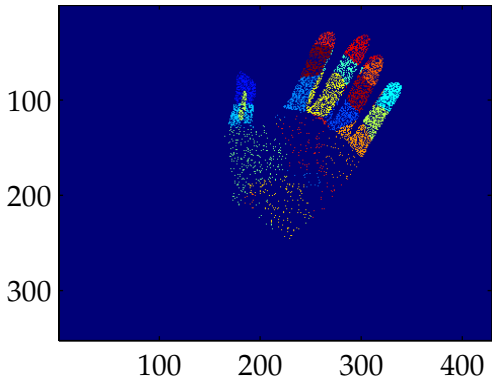
En la figura 5.17 se pueden observar los 100 autovalores de mayor magnitud del problema de autovalores generalizado (4.80) en la página 92, para una selección de 300 puntos de las figuras 5.2(a) y 5.2(b). Puesto que no se están escogiendo las direcciones de mayor variación de los datos, sino aquellas donde éstos muestran mayor correlación, el número de dimensiones necesario para conservar la mayor parte de la información es mucho mayor, pues la cantidad de variación capturada por cada autovector desciende mucho más lentamente

que cuando se utiliza el KPCA (véase las figuras 5.6 y 5.7). La figura 5.17 sólo muestra los resultados para una función de kernel guassiana ($h = 2$) —con una curva para el caso de la mano como un objeto rígido y la otra para la mano como un objeto articulado— y para la función de kernel triangular ($d = 1{,}5$). Uno de los requisitos para resolver el problema de autovalores generalizado (4.80) es que la matriz

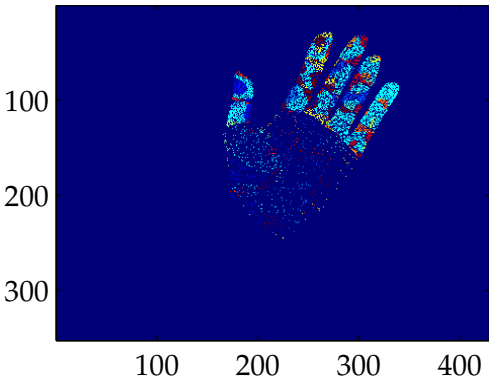
$$\begin{bmatrix} \mathbf{K}'^2 + \tau\mathbf{I} & 0 \\ 0 & \mathbf{K}^2 + \tau\mathbf{I} \end{bmatrix}$$

sea definida positiva. Desafortunadamente, con la función de kernel polinomial parece que esta condición no se garantiza, por lo que no se ha podido aplicar el KCCA. Mientras que para la función de kernel triangular ($d = 1{,}5$) con el modelo articulado, en los casos probados la matriz de autovalores siempre vale **0**. Sea como fuere, en la curvas mostradas se aprecia que se necesitan muchas más dimensiones para preservar la información de la relación entre los puntos seleccionados de cada imagen. En el caso concreto de la función de kernel triangular, incluso preservando los primeros 1 000 autovectores, no se consigue que sus autovalores desciendan por debajo de 1,0.

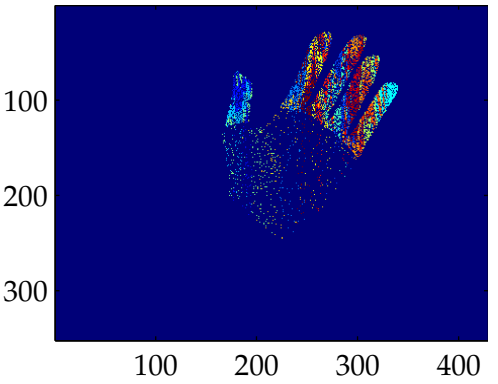
La figura 5.18 muestra el resultado de hacer el encaje de correspondencias mediante KCCA entre las imágenes de las figuras 5.2(a) y 5.2(b), con un espacio de componentes principales de dimensión $R = 100$. Las figuras de la parte superior — 5.18(a) y 5.18(b)— muestran el resultado utilizando una función de kernel gaussiana, con un modelo rígido y articulado respectivamente y correspondencias sin utilizar la información acerca de las etiquetas (5.2). Aunque era de suponer que el subespacio generado, por ser único y maximizar la correlación entre ambos conjuntos de datos, fuera mejor desde el punto de vista del encaje de correspondencias, lo cierto es que no ocurre así. El resultado mostrado en la figura 5.18(a) para el caso del modelo rígido parece razonable, pero ciertamente lleva a engaño, pues el mismo rendimiento se hubiera conseguido si la correspondencia se hubiera hecho directamente en el espacio de entrada $\mathbb{R}^2$. En general, cuanto menos se parecen las imágenes a encajar —por ser posturas más



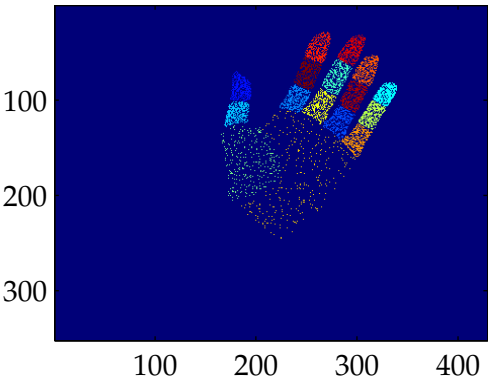
(a) Correspondencia de objetos rígidos con función de kernel gaussiana ($h = 2$).



(b) Correspondencia de objetos articulados mediante (5.2) con función de kernel gaussiana ($h = 2$).



(c) Correspondencia de objetos rígidos con función de kernel triangular ($d = 1{,}5$).



(d) Correspondencia de objetos articulados mediante (5.4) con función de kernel gaussiana ($h = 2$).

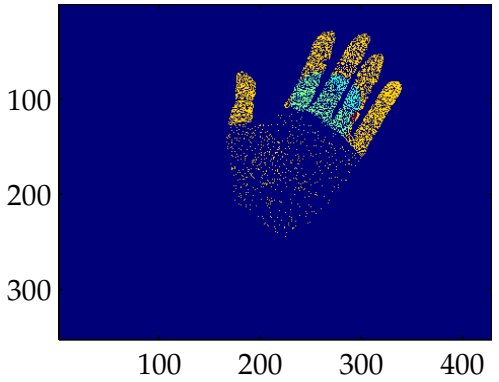
Figura 5.18. Encaje de correspondencias mediante KCCA entre las imágenes de las figuras 5.2(a) y 5.2(b) con un subespacio de componentes principales de dimensión $R = 100$.

alejadas dentro del movimiento de un mismo gesto— peor es la asignación de correspondencias. En la figura 5.18(d) se puede observar el resultado con la función de kernel gaussiana pero utilizando la información acerca de las etiquetas en el encaje (5.4). El resultado, al igual que en 5.13 en la página 126, es prácticamente perfecto. Nuevamente el encaje en base a (5.4) se muestra muy robusto, pero es de esperar que eso sea debido al peso de la información del etiquetado respecto a la de los puntos en el subespacio del KCCA.

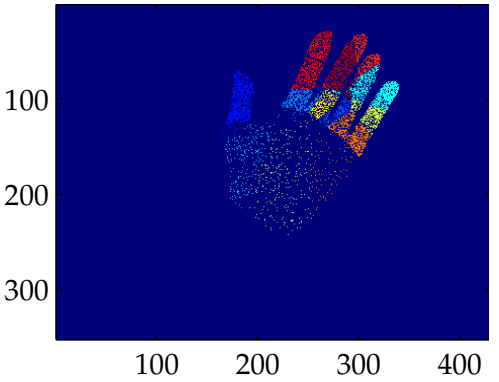
5.4.6. Encaje de puntos de objetos articulados con relajación

Finalmente, lo único que queda es comprobar qué ocurre cuando no se conocen las probabilidades asignadas a la segunda imagen. Se supone que en una hipotética aplicación real de esta técnica se tienen una serie de puntos de interés de la primera imagen convenientemente etiquetados. Estos pueden haber sido etiquetados manualmente o, en el caso de secuencias de vídeo, por una etapa de asignación de correspondencias anterior. Entonces, de la segunda imagen se seleccionan aleatoriamente una serie de puntos de interés, se inicializan sus probabilidades y se hace un KPCA con centrado por clases a ambos conjuntos de puntos. Posteriormente se utiliza (5.4) en la página 101 para asignar las correspondencias y se estima un error para la misma. Si este error ha variado de forma lo suficientemente significativa respecto a una iteración anterior, se utiliza un proceso de relajación para reajustar las probabilidades de que cada punto pertenezca a una clase concreta y se vuelve a repetir el KPCA de los conjuntos de puntos. Las etapas y las ecuaciones de este proceso se describen con mayor detalle en el algoritmo 5.3 en la página 105.

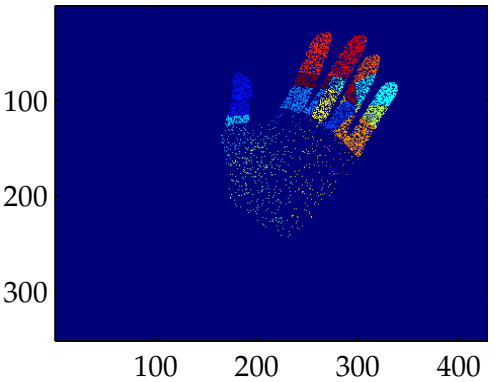
Para analizar qué se puede conseguir de la asignación de correspondencias en objetos articulados con relajación, se procedió a visualizar el resultado de la primera asignación del algoritmo, la que ocurre justo antes de la primera ejecución del proceso de relajación. En la figura 5.19 se puede observar el resultado para el primer encaje, cuando las probabilidades de los puntos de la segunda imagen utilizan una probabilidad uniforme de ser de cada etiqueta y la función de kernel



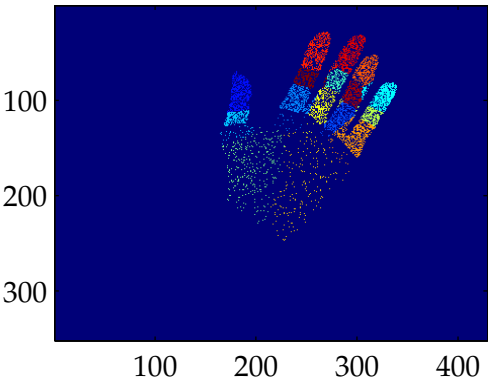
(a) Correspondencia utilizando las probabilidades conocidas para los puntos de la primera imagen.



(b) Correspondencia utilizando una mezcla al 50 % entre las probabilidades conocidas y la probabilidad uniforme para los puntos de la primera imagen.



(c) Correspondencia utilizando una mezcla al 20 % para las probabilidades conocidas y al 80 % para la probabilidad uniforme para los puntos de la primera imagen.

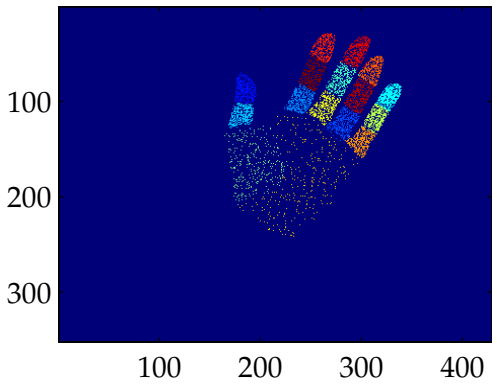


(d) Correspondencia utilizando una mezcla al 10 % para las probabilidades conocidas y al 90 % para la probabilidad uniforme para los puntos de la primera imagen.

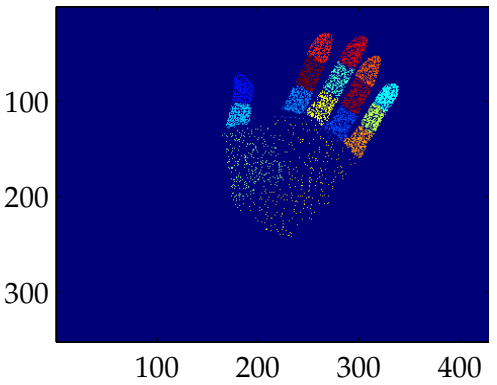
Figura 5.19. Primer encaje de correspondencias de un proceso de asignación entre objetos articulados con relajación, utilizando las figuras 5.2(a) y 5.2(b) con función de kernel gaussiana. Las probabilidades se han tomado de forma que las de los puntos de la segunda imagen siempre son uniformes, pero las de los puntos de la primera son una mezcla con porcentajes variables entre la probabilidad conocida de dichos puntos y una probabilidad uniforme.

es gaussiana. Al igual que en el caso comentado en el apartado 5.4.3, no se pudo aplicar el el algoritmo 5.3 con las funciones de kernel polinomial o triangular. La figura 5.19(a) muestra que los resultados no son tan buenos como cabría esperar en base a lo visto en la figura 5.13. El motivo es que las matrices de probabilidad $\mathbf{P}$ y $\mathbf{P}'$ son un factor determinante en la asignación de correspondencia. En las figuras 5.19(b), 5.19(c) y 5.19(d) se puede observar el encaje inicial cuando las probabilidades de los puntos de la primera imagen se van mezclando con una probabilidad uniforme. Según se van pareciendo más las probabilidades para los puntos de la primera imagen con los de la segunda, mejor funciona la asignación de correspondencias. El problema es que para que las probabilidades de ambos conjuntos de puntos se parezcan es necesario ignorar la información que aportan las probabilidades de los puntos de la primera imagen.

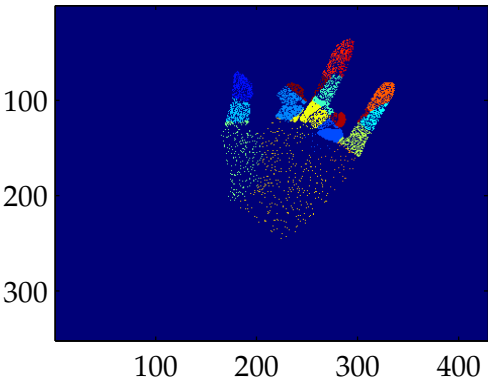
Una posible solución a este problema sería utilizar el algoritmo 5.1 para obtener una primera aproximación del etiquetado de los puntos de la segunda imagen. Así la matriz $\mathbf{P}'$ de probabilidad de los puntos de la segunda imagen podría inicializarse a partir de la frecuencia de las etiquetas que aparecen en la vecindad de un punto dado. En la figura 5.20 se puede observar la aplicación de este procedimiento a las parejas de las figuras 5.2(a) y 5.2(b) —en la parte superior— y las figuras 5.2(a) y 5.2(f) —en la parte inferior—. En las primeras no es fácil observarlo, pero el encaje inicial no es correcto debido al leve movimiento del dedo entre las postura de las figuras 5.2(a) y 5.2(b). Era de esperar que el proceso de relajación resolviera esto. Sin embargo, tras ejecutar el algoritmo 5.3 por completo, se puede apreciar en la figura 5.20(b) que estos fallos no se corrigieron. En la parte inferior, donde se utilizan posturas que difieren más que las primeras, el fenómeno se observa mucho mejor. El etiquetado inicial, considerando la mano como un objeto rígido, prácticamente funciona como un encaje en $\mathbb{R}^2$. Es decir, a cada punto de la segunda imagen se le asigna la etiqueta que tendría si estuvieran en el mismo sitio pero en la primera imagen. A partir de aquí el proceso de relajación lo único que puede hacer es suavizar la asignación de las probabilidades, sin que las etiquetas cambien sustancialmente. Esto último es especialmente cierto en el presente caso, donde los coeficientes de compatibilidad



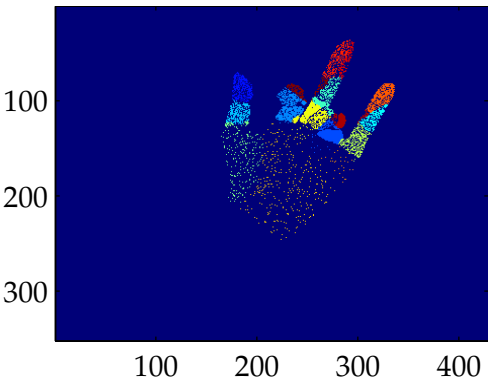
(a) Primer encaje entre las figuras 5.2(a) y 5.2(b).



(b) Resultado final del encaje entre las figuras 5.2(a) y 5.2(b).



(c) Primer encaje entre las figuras 5.2(a) y 5.2(f).

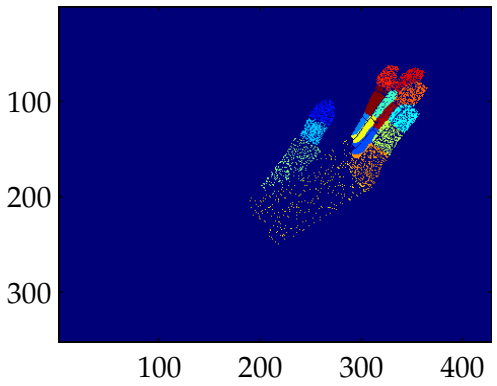


(d) Resultado final del encaje entre las figuras 5.2(a) y 5.2(f).

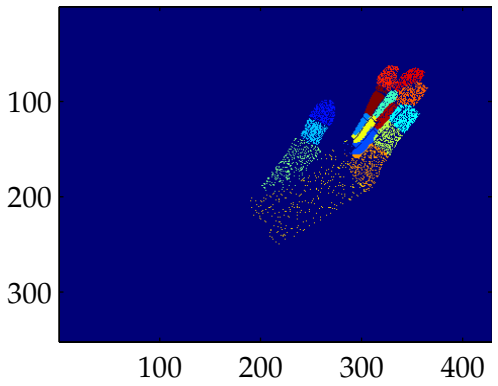
Figura 5.20. Encaje de correspondencias para objetos articulados con relajación usando posturas del gesto «A». En la columna de la izquierda se muestra el resultado de la asignación tras inicializar las matrices de probabilidad. Mientras que en la columna de la derecha se puede observar el resultado final del encaje. Las de la parte superior se obtuvieron calculando las correspondencias entre las las figuras 5.2(a) y 5.2(b). Mientras que para la parte inferior se utilizaron las figuras 5.2(a) y 5.2(f).

R seleccionados (5.9) se han escogido para garantizar que las regiones etiquetadas son uniformes, careciendo de información real sobre las regiones que son y que no son compatibles. Sin embargo, como se ha comentado anteriormente, es muy complejo el aprender los coeficientes de compatibilidad, a partir de posturas que sirvan de muestras de entrenamiento (Pelillo y Refice, 1994), en un objeto con tantos grados de libertad como tiene la mano humana. En la figura 5.21 se pueden observar los mismos resultados pero utilizando posturas del gesto «B». En dicha figura se ve bastante mejor como el etiquetado inicial no es correcto, incluso entre figuras tan parecidas como son la 5.4(a) y la 5.4(b), y como el proceso de relajación no es capaz de corregirlo.

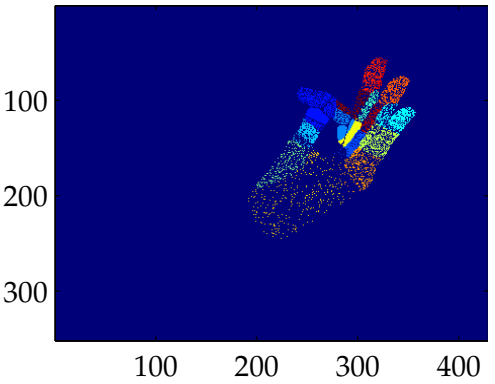
El algoritmo 5.3 requiere que se le especifiquen algunos parámetros. En la parte referente al proceso de relajación hay que establecer qué se considera el vecindario de un punto. En las pruebas que se muestran en las figuras 5.20 y 5.21 se han considerado así a los ocho vecinos más próximos. En general, aumentar el número de vecinos no tiene mayor efecto que el de suavizar un poco más los valores de probabilidad entre los mismos. También hay que establecer las condiciones de parada, tanto para el algoritmo completo como para el proceso de relajación. En las pruebas éstas se fijaron a una variación en el error inferior al 1 % para ambos casos, con un máximo de 100 iteraciones para el primero y 1 000 para el proceso de relajación. En todo caso, la variación de estos parámetros tampoco tiene efecto algun,o pues el algoritmo completo suele detenerse después de una pocas iteraciones, mientras que el proceso de relajación es bastante lento en el ajuste de las probabilidades, requiriendo muchas más interacciones aunque sin alcanzar nunca las 1 000 fijadas como máximo. En este sentido, sería de esperar que el parámetro α de (5.11) fuese un factor determinante. Sin embargo, tal y fueron escogidas (5.11) y (5.10), la única forma de que un punto pierda probabilidad para un etiqueta determinada es que reciba una aportación de 0 para la misma, perdiendo probabilidad a través de la normalización en (5.11). Esto hace que el proceso sea algo más lento, respecto a otros procesos de relajación descritos en la literatura, incluso con valores de α relativamente altos. Pero en ningún caso introducir modificaciones en este sentido —variando el valor de α



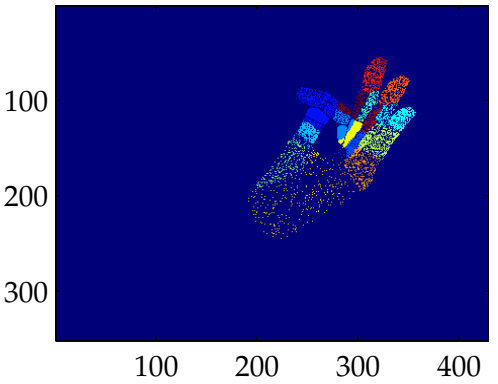
(a) Primer encaje entre las figuras 5.4(a) y 5.4(b).



(b) Resultado final del encaje entre las figuras 5.4(a) y 5.4(b).



(c) Primer encaje entre las figuras 5.4(a) y 5.4(f).



(d) Resultado final del encaje entre las figuras 5.4(a) y 5.4(f).

Figura 5.21. Encaje de correspondencias para objetos articulados con relajación usando posturas del gesto «B». En la columna de la izquierda se muestra el resultado de la asignación tras inicializar las matrices de probabilidad, mientras que en la columna de la derecha se puede observar el resultado final del encaje. Las de la parte superior se obtuvieron calculando las correspondencias entre las las figuras 5.4(a) y 5.4(b), mientras que para la parte inferior se utilizaron las figuras 5.4(a) y 5.4(f).

o utilizando un proceso de relajación más tradicional— permite que los errores iniciales de asignación de correspondencias sean corregidos.

Capítulo

6

Detalles de implementación

Las técnicas utilizadas para intentar resolver el problema del reconocimiento gestual requieren de una cantidad importante de recursos computacionales. Por ejemplo, para aplicar el KPCA a un conjunto de 10 000 puntos es necesario disponer de memoria para una matriz $\mathbf{K}$ de $10\,000 \times 10\,000$ elementos. Además, para dicha matriz hay que resolver el problema de autovalores (4.11), proceso que requiere una cantidad significativa de tiempo de CPU, incluso en los equipos más modernos. Aparte de esto, en el algoritmo propuesto por Wang y Hancock (2006) hay que comparar todos los puntos del subespacio de componentes principales del conjunto de puntos de la primera imagen, con todos los puntos del segundo conjunto al utilizar (5.4) en el encaje de correspondencias. Si bien para conjuntos pequeños esto no es un problema, con 10 000 se necesitan $10\,000 \times 10\,000$ cálculos de la distancia y 10 000 búsquedas del máximo. A continuación se presentan algunos de los desarrollos realizados para afrontar éstos y otros desafíos.

6.1. Búsqueda de vecinos más próximos mediante Cover-Tree

Para resolver (5.2) y (5.4) es necesario encontrar el vecino $\boldsymbol{y}'_m$ más próximo a un punto $\boldsymbol{y}_n$ dado. La solución más sencilla de calcular la distancia de $\boldsymbol{y}_n$ a cada punto $\boldsymbol{y}'_m$, recordando durante el proceso cuál fue el último punto más próxi-

mo encontrado, funciona para conjuntos pequeños. Pero el problema se vuelve rápidamente intratable a poco que crezca el conjunto de datos o el número de dimensiones del espacio. Este tipo de búsqueda tiene un tiempo de ejecución del $\mathcal{O}(NR)$, donde N es el número de elementos del conjunto y R es la dimensionalidad del espacio.

En la literatura se han propuesto diversas técnicas para resolver el problema de la búsqueda de vecinos más próximos para conjuntos de datos de gran tamaño y/o espacios de elevada dimensionalidad. Por ejemplo, KD-Tree (Bentley, 1980; Friedman *et al.*, 1977), R-Tree (Guttman, 1984), VP-Tree (Yianilos, 1993) y BK-Tree (Burkhard y Keller, 1973). Los dos últimos utilizan árboles métricos, por lo que sólo pueden ser utilizados cuando el espacio de los datos es un espacio métrico[1]. Los árboles métricos aprovechan las propiedades de los espacios métricos —como la desigualdad triangular de los espacios euclídeos— para hacer que el acceso a los datos sea mucho más eficiente. No todos estos algoritmos y estructuras de datos presentan un buen rendimiento cuando se trabaja en espacios de elevada dimensionalidad. Tal es el caso, por ejemplo, del KD-Tree. Si esto ocurre, se suele optar por soluciones aproximadas a la búsqueda de los vecinos más próximos, de manera que se entiende que no es tan importante encontrarlos con exactitud como hacerlo en un tiempo «razonable», por lo que basta con utilizar algún algoritmo que sea capaz se de hallar alguno de los vecinos más cercanos. En la actualidad las técnicas de búsqueda de vecinos más próximos ϵ-aproximadas han ganado mucha popularidad de cara a enfrentar el problema en espacios de elevada dimensionalidad.

Una de las técnicas más prometedoras de búsquedas de vecinos más próximos es el Cover-Tree (Beygelzimer *et al.*, 2006a). Especialmente si se tiene en cuenta que el tiempo de búsqueda para esta técnica depende de la dimensión intrínseca del conjunto de datos.

[1] Un espacio métrico es aquel donde se ha definido la noción de distancia entre elementos del espacio.

6.1.1. Cover-Trees para la búsqueda de vecinos más próximos

El Cover-Tree (Beygelzimer *et al.*, 2006a) es una estructura de datos que permite consultas tanto para obtener los k vecinos más próximos como para obtener todos los vecinos en un radio ϵ de un dato dado. La estructura de datos es un árbol métrico, por lo que sólo puede ser utilizada cuando el espacio de los datos es un espacio métrico. Dicha estructura requiere $\mathcal{O}(N)$ de espacio de almacenamiento, independientemente de las propiedades métricas del espacio. El árbol puede ser construido en tiempo $\mathcal{O}(c^6 N \log N)$ y consultado en $\mathcal{O}(c^{12} \log N)$. Es decir, el tiempo de búsqueda es sólo proporcional al logaritmo del número de datos N y a c, la *constante de expansión* del conjunto de datos. La constante de expansión del espacio de los datos $\mathbb{S}$ se define como el valor más pequeño —con $c \geq 2$— tal que $|B_{\mathbb{S}}(p, 2r)| \leq c|B_{\mathbb{S}}(p, r)|$, donde $B_{\mathbb{S}}(p, r) = q \in \mathbb{S} : d(p, q) \leq r$ es la esfera de radio r alrededor del punto $p \in \mathbb{S}$. Si los puntos en el espacio $\mathbb{S}$ estuvieran distribuidos uniformemente en una superficie de dimensión D, entonces $c \sim 2^D$, motivo por el cual la constante de expansión es considerada una medida de la dimensión intrínseca de los datos. En el KPCA la transformación no lineal $\Phi(x)$ mapea los puntos $\boldsymbol{x}_n$ del espacio de entrada de dimensión D en una variedad, también de dimensión D, en el espacio de características $\mathbb{F}$ de dimensión M. Por tanto, el hecho de que el tiempo de búsqueda del Cover-Tree esté condicionado a la dimensión intrínseca de los datos D, y no a la dimensión M del espacio donde han sido mapeados, puede ser muy beneficioso pues esta última suele ser muy elevada.

La estructura de datos del Cover-Tree es un árbol por niveles donde cada nivel «cubre» a todos los niveles por debajo de él. Cada nivel se indexa por una escala entera i que se decrementa según se desciende por el árbol. Suponiendo que C_i sea el conjunto de nodos en el nivel i, un árbol de Cover-Tree cumple con las siguientes propiedades para todos los niveles i:

Anidamiento $C_i \subset C_{i-1}$.

Cubrimiento Para cada $p \in C_{i-1}$, existe un $q \in C_i$ que satisface que $d(p, q) \leq 2^i$,

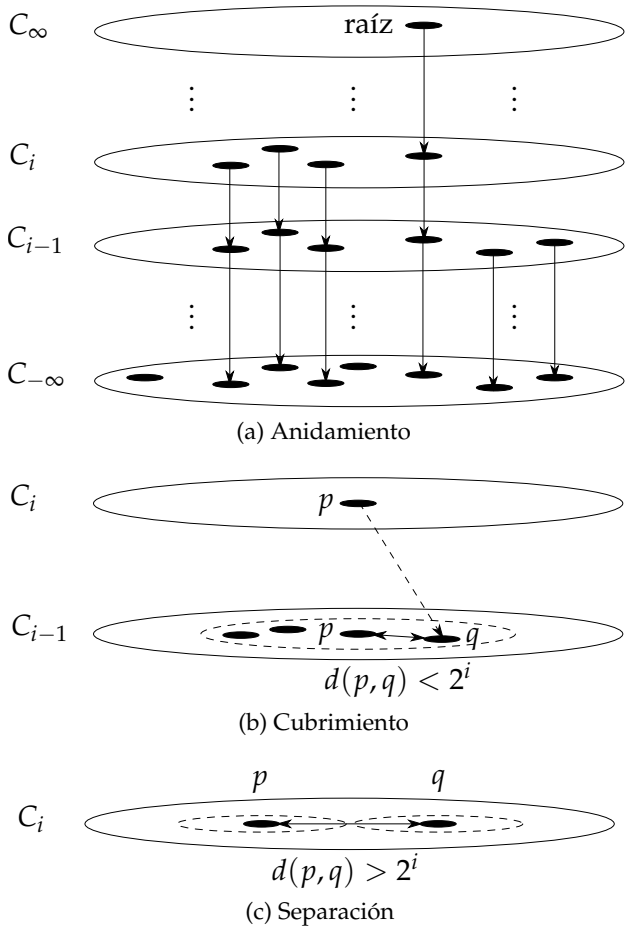


Figura 6.1. Ilustración de las propiedades del árbol de Cover-Tree.

y exactamente uno tal que q es padre de p.

Separación Para todo $p, q \in C_i,\ d(p,q) > 2^i$.

En la figura 6.1 se puede observar una representación gráfica de estas propiedades.

Beygelzimer *et al.* (2006a) describen tanto la estructura de datos como los algoritmos para la construcción de la misma y el borrado y la inserción de los puntos. Éstos se describen en términos de una representación implícita donde existen un número infinito de niveles, siendo C_∞ el nivel que contiene el nodo raíz del árbol. Sin embargo, la implementación real de estos algoritmos utiliza una representación explícita que sólo ocupa $\mathcal{O}(N)$ de espacio de almacenamiento. Es importante recordar que si un punto $p \in S$ aparece por primera vez en el nivel i, entonces debe estar en todos los niveles por debajo del i, siendo un hijo de sí mismo en todos esos niveles. Así, la representación explícita del árbol de Cover-Tree colapsa todos los nodos donde el único hijo es el propio padre, sabiendo que implícitamente deberían estar ahí pero sin reservar espacio para ellos. En general, un nodo explícito es aquel que tiene otro padre aparte de él mismo o un hijo aparte de sí mismo, lo que automáticamente establece un límite superior al espacio de almacenamiento requerido.

6.1.2. Cover-Tree desde MATLAB

Puesto que los algoritmos de visión por computador se implementaron para `MATLAB`, era necesario disponer de una *toolbox* de Cover-Tree para dicho entorno. Beygelzimer *et al.* (2006c) proporcionan una implementación del Cover-Tree en C++, en base a lo comentado en una versión extendida de su artículo (Beygelzimer *et al.*, 2006b). El artículo original describe la estructura del árbol del Cover-Tree, así como los algoritmos de inserción, eliminación de datos y de búsqueda del vecino más próximo de un punto dado. Ninguno de estos algoritmos se implementa en la librería proporcionada por Beygelzimer *et al.* (2006c). En su lugar, ésta ofrece funciones para construir un árbol de Cover-Tree a partir de un con-

junto de datos y para buscar los vecinos más próximos de todos los puntos de otro conjunto de datos que sirve de consulta (Beygelzimer *et al.*, 2006b). Puesto que las primeras son menos eficientes que las segundas, en Beygelzimer *et al.* (2006c) sólo se implementaron estas últimas.

De cara a su utilización desde MATLAB, se tuvieron que realizar algunas modificaciones al código original de Beygelzimer *et al.* (2006c).

Implementación del algoritmo de inserción

El código original proporciona las funciones

```
template<class P>
node<P> batch_create(v_array<P> points);

template <class P>
void k_nearest_neighbor(const node<P> &top_node,
    const node<P> &query, v_array<v_array<P> > &results,
    int k);

template <class P>
void epsilon_nearest_neighbor(const node<P> &top_node,
    const node<P> &query, v_array<v_array<P> > &results,
    float epsilon);

template <class P>
void unequal_nearest_neighbor(const node<P> &top_node,
    const node<P> &query, v_array<v_array<P> > &results);
```

para construir y buscar vecinos más próximos por lotes (Beygelzimer *et al.*, 2006b). Aunque estos algoritmos son más eficientes que los que operan con datos individuales, en ocasiones puede ser interesante añadir puntos sin tener que reconstruir el árbol completo, especialmente si la cantidad de puntos a insertar es lo suficientemente pequeña como para que compense. La implementación de Beygelzimer *et al.* (2006c) busca maximizar la eficiencia, por lo que la estructura del

árbol de Cover-Tree que utilizan es notablemente diferente a la propuesta en el artículo original (Beygelzimer *et al.*, 2006a), careciendo de los elementos necesarios para incluir el algoritmo de inserción. Esto obligó a hacer importantes modificaciones en la estructura de dicho árbol, para finalmente poder introducir la función

```
template <class P>
void insert(const P &p, node<P> &top_node);
```

Cálculo de doble precisión para las distancias

El código original de Beygelzimer *et al.* (2006c) utiliza plantillas para los datos —que se definen como de tipo `P`— a incluir en el árbol, de forma que no se fuerza al programador a utilizar ningún formato concreto para los puntos. Independientemente de la estructura, clase o tipo que el programador utiliza para `P`, se debe definir una función capaz de calcular la distancia entre dos puntos

```
float distance(P v1, P v2, float upper_bound);
```

Dicha función debe retornar un flotante de precisión sencilla con la distancia entre los puntos `v1` y `v2`. Para aumentar la precisión cuando se trabaja con puntos muy próximos, se redefinió la función de distancia para que retornara flotantes de precisión doble. Se reajustaron los campos de las estructuras que definen los nodos del árbol, así como las variables intermedias y otras estructuras utilizadas en el código, para el nuevo rango de distancias posibles. En todo caso, puesto que con las nuevas restricciones se consume más memoria y más tiempo de CPU en las búsquedas, se han mantenido ambas versiones, la que usa flotantes de precisión sencilla y la que los usa de precisión doble.

Implementación del MEX para MATLAB

Para invocar código C/C++ o FORTRAN desde `MATLAB` es necesario que éste sea incluido en un ejecutable de `MATLAB` (MEX). Un MEX no es más que una libre-

ría de enlace dinámico que debe cumplir con ciertas convenciones. Por ejemplo, todo MEX tiene una función «punto de entrada»

```
void mexFunction(int nlhs, mxArray *plhs[], int nrhs,
    const mxArray *prhs[])
```

que es llamada cuando el usuario invoca el nombre del MEX como si de una función nativa se tratara. A través de este punto de entrada se reciben los argumentos desde `MATLAB` y se devuelven los resultados de las operaciones realizadas.

Para invocar las funciones de la implementación del Cover-Tree, se desarrolló un MEX de nombre `covertree_call` que, a través de su punto de entrada, debe recibir los siguientes argumentos:

1. Manejador que identifique al árbol de Cover-Tree sobre el que se va a realizar la operación.

2. Cadena de caracteres que identifica la operación a realizar sobre el árbol de Cover-Tree

3. Un número variable de argumentos que dependen de la operación solicitada.

En el apartado A.1 en la página 175 se resumen las operaciones soportadas con los argumentos recibidos y devueltos para cada una. Es importante destacar, que sea cuál sea la operación a realizar siempre es necesario proporcionar en el primer argumento de la función el manejador del árbol de Cover-Tree, excepto para el caso de la que inicializa dicho árbol —`batch_create`— que requiere que éste sea una matriz vacía `[]`. En realidad, el manejador no es más que un puntero al nodo raíz del árbol, sólo que convertido en un escalar entero sin signo de 64 bits antes de devolverlo a `MATLAB`, dado que éste no sabe manejar punteros.

Utilización del gestor de memoria de MATLAB

El código original de Beygelzimer *et al.* (2006c) utiliza directamente la gestión de memoria del sistema operativo a través de las funciones `malloc()` y `free()`.

Sin embargo, eso hace que dicha memoria no quede bajo el control del gestor de memoria de MATLAB. Cuando una función del API de MATLAB invocada desde un MEX falla, el MATLAB retoma el control, libera los recursos reservados —incluída la memoria— y devuelve el control al usuario a través de la ventana de comandos. Si se permitiera a la librería pedir memoria directamente al sistema, dicha memoria no estaría bajo el control de MATLAB, por lo que no sería liberada en caso de error en alguna función del API. Por tanto, en lugar de las funciones `malloc()` y `free()` se deben utilizar las funciones `mxMalloc()` y `mxFree()` de MATLAB.

Toda la memoria reservada dentro de un MEX —incluída la que se reserva mediante `mxMalloc()`, `mxFree()` u otras funciones que se utilizan para construir matrices— lo es en modo no persistente. Esto quiere decir que al salir del MEX toda la memoria reservada se libera automáticamente, lo cual es deseable en caso de error pero no cuando las operaciones se ejecutan correctamente. Para asegurar que la memoria reservada para el árbol no es liberada entre invocaciones al MEX se creó una pila donde almacenar los punteros a las regiones de memoria que, en caso de terminar la ejecución con éxito, no deben ser liberadas. Antes de devolver el control a MATLAB se recorre la pila y se marca la memoria como persistente mediante las funciones `mexMakeMemoryPersistent()` y `mexMakeArrayPersistent()` del API.

Mejora del rendimiento mediante paso por referencia

Oficialmente en MATLAB los argumentos siempre se pasan por valor. Esto quiere decir que cada matriz que recibe una función debe ser copiada antes de pasársela a dicha función, lo cual consume una cantidad importante de memoria y de tiempo de CPU. Realmente MATLAB utiliza internamente «copia en escritura», de manera que los argumentos son pasados por referencia y sólo se copian cuando la función invocada los tiene que modificar. Esto mejora notablemente la eficiencia del programa, pero es un mecanismo que no está disponible cuando se desarrolla un MEX.

Por ejemplo, en `covertree_call` los puntos que se pasan en los argumen-

tos para construir el árbol de Cover-Tree deben ser duplicados antes de insertarlos. De igual forma, al hacer una consulta los resultados encontrados en el árbol deben ser duplicados antes de ser devueltos a MATLAB. Sin embargo, en covertree_call se ha utilizado la función de MATLAB no documentada

```
mxArray* mxCreateReference(const mxArray *pm);
```

que evita que sea necesario copiar los datos al recibirlos y devolverlos de MATLAB. Todos los elementos de MATLAB tienen un contador interno que indica desde cuántos puntos del programa está siendo referenciado el elemento. Este contador se decrementa cada vez que se intenta destruir el elemento, aunque sólo es destruido si el contador llega a 0. Usando mxCreateReference con los datos de entrada se le está diciendo a MATLAB que éstos están referenciados en algún sitio —concretamente en el árbol de Cover-Tree donde van a ser insertados—. De igual manera, al devolverlos en una consulta hay que incrementar el contador, para que cuando MATLAB intente destruirlos siga quedando constancia de que existe una referencia desde el árbol de Cover-Tree, evitando que la memoria sea liberada. Lamentablemente, esta función no está soportada de forma oficial.

Funciones de distancia entre los datos

Los datos pasados por MATLAB al MEX son del tipo mxArray*, cuya estructura interna sólo es conocida por el programa. Puesto que el código de la implementación del Cover-Tree utiliza plantillas, no es necesario convertir estos datos a alguno de los tipos soportados por C++. En su lugar se puede utilizar directamente mxArray* como el tipo P de los puntos a almacenar en el árbol. Sin embargo es necesario proporcionar una función capaz de calcular la distancia entre dos puntos de tipo mxArray*

```
float distance(mxArray* v1, mxArray* v2, float upper_bound);
```

Esta función no está programada dentro del MEX sino que debe ser proporcionada durante la creación del árbol —mediante batch_create— pues su funcionamiento depende de la naturaleza de los datos en MATLAB. En la actualidad el

MEX acepta que la función de distancia le sea proporcionada de cualquiera de las siguientes maneras:

- Como un manejador de función de `MATLAB`. Esto permite utilizar funciones anónimas de `MATLAB` para calcular las distancias.
- Como una cadena de caracteres que indica el nombre de una función de `MATLAB`. Esto permite que la función de cálculo de la distancia se implemente en un archivo de *script* de `MATLAB`.
- Como una cadena de caracteres que indica el nombre de una librería de enlace dinámico y el de una función, separados por dos puntos. Esto permite utilizar funciones implementadas en C/C++.

Las dos primeras opciones son las más flexibles puesto que permiten utilizar funciones programadas directamente en `MATLAB`. Sin embargo, las transiciones del código del MEX a `MATLAB` y viceversa son tremendamente ineficientes. Puesto que la función de cálculo de la distancia debe invocarse varias veces durante la ejecución de las operaciones, las primeras opciones sólo son interesantes para hacer pruebas con pequeños conjuntos de datos. La tercera opción permite utilizar funciones escritas directamente en C/C++. Aunque estas funciones siempre tendrán que utilizar el API de `MATLAB` para acceder al contenido de los datos, pues éstos se pasan como elementos del tipo `mxArray*`, serán más eficientes que si estuvieran programadas en el lenguaje de `MATLAB`.

Para facilitar el desarrollo de funciones para el cálculo de la distancia en C++, junto con el MEX se incluye una pequeña librería con plantillas para el cálculo de la resta y el producto escalar de vectores, así como para el cálculo de la distancia euclídea entre vectores de tipo genérico `T`. Estas plantillas han sido codificadas para facilitar la vectorización del código generado por el compilador mediante el uso de instrucciones SIMD[2]. También se proporcionan implementaciones de

[2]Las instrucciones SIMD se utilizan para conseguir paralelismo a nivel de datos. Son instrucciones que aplican una misma operación sobre un conjunto más o menos grande de datos. Un ejemplo de este repertorio es el SSE que incluyen los procesadores modernos de la familia x86.

esas plantillas optimizadas para tipos concretos. Por ejemplo, para flotantes de tipo doble y sencillo se ofrece una implementación del producto escalar basada en las funciones de la librería de álgebra lineal que acompaña a `MATLAB`.

Serialización y deseralización del árbol de Cover-Tree

Una de las operaciones más costosas del Cover-Tree es la construcción del árbol a partir de los datos iniciales. Por ello es interesante poder guardarlo y recuperarlo de un dispositivo de almacenamiento persistente a voluntad, en lugar de tener que volver a construirlo en cada ocasión. Sin embargo, la implementación original de Beygelzimer *et al.* (2006c) no ofrece esta posibilidad.

Se utilizo la librería `boost::serialization` para implementar la serialización y deserialización de la estructura del árbol de Cover-Tree, así como su almacenamiento y recuperación en y a partir de archivos, tanto en formato de texto como binario. El principal inconveniente de esta aproximación es que los datos almacenados en el árbol también deben ser serializados pero, puesto que éstos son del tipo `mxArray*`, están en un formato sólo conocido por `MATLAB`. Afortunadamente en dicho programa existen las funciones no documentadas

```
mxArray* mxSerialize(const mxArray *pm);
mxArray* mxDeserialize(const void *pr, mwSize n);
```

que permiten serializar y deserializar estructuras `mxArray` respectivamente.

Distancia a los vecinos más próximos

Las funciones de la implementación original

```
template <class P>
void k_nearest_neighbor(const node<P> &top_node,
    const node<P> &query, v_array<v_array<P> > &results,
    int k);

template <class P>
void epsilon_nearest_neighbor(const node<P> &top_node,
```

```
    const node<P> &query, v_array<v_array<P> > &results,
    float epsilon);

template <class P>
void unequal_nearest_neighbor(const node<P> &top_node,
    const node<P> &query, v_array<v_array<P> > &results);
```

devuelven los vecinos más próximos según ciertos criterios. Por ejemplo, la primera función devuelve los `k` vecinos más próximos a cada uno de los puntos consultados, mientras que `epsilon_nearest_neighbor()` busca todos los puntos en un radio `epsilon` en torno a cada uno de los puntos. Sin embargo, en ocasiones puede ser interesante conocer la distancia entre dichos vecinos y el punto consultado para utilizarlo en alguna etapa de procesamiento posterior. Por ello, el código de ambas funciones fue modificado con el objetivo de que en `results` no sólo fueran devueltos los puntos sino tambieénn su distancia al punto del que son vecinos más próximos.

Interfaz orientada a objetos en MATLAB

Para facilitar la utilización del Cover-Tree desde `MATLAB` se implementó una clase que ocultase las singularidades del MEX. La clase se denomina `CoverTree` y hereda de la clase abstracta `handle`. Estas clases tiene la propiedad de que si una instancia de la clase es copiada, ambos objetos realmente referenciarán a un mismo y único objeto. Se suele heredar de este tipo de clases cuando se pretende que sus instancias se comporten como manejadores de algo, como, por ejemplo, los objetos gráficos de `MATLAB`. Heredar de `handle` es interesante, puesto que en caso contrario cada vez que se copiara un objeto de la clase `CoverTree` habría que duplicar el árbol de Cover-Tree completo, con el consumo de tiempo y memoria que eso implicaría. En el apartado A.2 en la página 178 se resumen los métodos públicos de la clase con los argumentos recibidos y devueltos para cada uno, mientras que en el apartado A.3 en la página 181 se listan las propiedades públicas.

Tabla 6.1. Funciones de generación de posturas en `MATLAB`.

Función	Descripción
`interphandposes`	Genera posturas a partir de dos posturas dadas mediante interpolación esférica lineal (SLERP)[2].
`makehandposes`	Genera posturas aleatorias considerando todas las restricciones comentadas en el apartado 3.1.2 en la página 42.
`makehandposes2`	Genera posturas muestreando gestos prefijados. Los gestos utilizados son todas las combinaciones de flexión de los dedos, incluido el pulgar, para diferentes ángulos de los 3 grados de libertad de la muñeca.
`makerandposes`	Genera posturas aleatorias considerando únicamente el intervalo de ángulos válidos para cada articulación.

6.2. Síntesis de imágenes desde MATLAB con Blender

Tal y como se comentó en el capítulo 3 en la página 35, se creó en el programa `Blender` una mano sintética, con un modelo cinemático (véase el apéndice B en la página 183) que incluye restricciones estáticas sobre el rango de los ángulos válidos para cada articulación (véase la tabla 3.2 en la página 45). Al mismo tiempo se desarrollaron para `MATLAB` una serie de funciones capaces de generar matrices que describen posturas que cumplen con todas las restricciones, tanto estáticas como dinámicas, comentadas en el apartado 3.1.2 en la página 42. En la tabla 6.1 se resumen las características de las funciones más relevantes. La cuestión entonces es cómo sintetizar con `Blender` las manos en las posturas generadas con dichas funciones de `MATLAB`.

[2]La interpolación esférica lineal (SLERP) fue introducida por Ken Shoemake en el contexto de la interpolación de cuaterniones para la animación 3D (Shoemake, 1985). Fundamentalmente consiste en suponer que se recorre a velocidad constante un arco de radio unidad entre los cuaterniones que describen las rotaciones entre las que se quiere interpolar. A través del parámetro de interpolación —entre 0,0 y 1,0, donde 0,0 es un extremo del arco y 1,0 es el otro— se escoge una rotación intermedia concreta.

6.2.1. La clase BlenderPoses

Todas las funciones comentadas en la tabla 6.1 retornan objetos `BlenderPoses` de `MATLAB`, que son el elemento básico de la solución desarrollada. Fundamentalmente un objeto `BlenderPoses` contiene una matriz de $4 \times 4 \times N \times M$ que almacena M configuraciones de postura de la mano para N articulaciones. Estos objetos pueden ser construidos vacíos

```
>> BP = BlenderPose();
```

inicializados con una lista de nombres de articulaciones, mediante un vector *cell* o una matriz `BONENAMES` de cadenas de caracteres

```
>> BP = BlenderPose(BONENAMES);
```

o reservando espacio para un número fijo `N` de configuraciones de postura.

```
>> BP = BlenderPose(BONENAMES, N);
```

Por eficacia y sencillez, la clase `BlenderPoses` sobrecarga los operadores correspondientes para que las instancias de la misma puedan ser manipuladas de forma similar a una matriz $N \times M$ convencional. Así, por ejemplo, es posible obtener las matrices 4×4 con las posturas de todas las articulaciones para la i-ésima configuración de la mano

```
>> A = BP(i);
```

o sólo la matriz 4×4 de la j-ésima articulación en la i-ésima configuración

```
>> A = BP(i, j);
```

Es decir, la primera dimensión de un objeto `BlenderPose` sirve para indexar la configuración de la postura, mientras que la segunda dimensión permite indexar la articulación concreta que interese recuperar. De forma similar se pueden asignar valores a los elementos de esas matrices

```
>> BP(i) = A;
>> BP(i, j) = B;
```

incluso por bloques

```
>> BP(i + 1:100) = A;
>> BP(i + 1:100, j + 1:10) = B;
>> BP(1:10, 10:end) = C;
```

pero en ningún caso se pueden exceder las dimensiones de la matriz almacenada en el objeto `BlenderPoses`. Es decir, la asignación permite acceder a las configuraciones y articulaciones existentes pero nunca añadir nuevas.

Para añadir nuevas configuraciones se puede utilizar el método `setpose()`. Éste permite fijar una postura `M` concreta para una articulación determinada

```
>> BP = setpose(BP, i, j, M);
```

con la ventaja adicional de que si no hay espacio reservado para la configuración de la mano indicada, éste se reserva antes de almacenar el nuevo valor. De igual manera, para añadir una nueva articulación se utiliza el método `addbone()`

```
>> [BP, BONENO] = addbone(BP, BONENAME);
```

donde `BONENAME` es el nombre de la nueva articulación y `BONENO` su ubicación en la lista de nombres de articulaciones. En ambos casos las posturas para las nuevas configuraciones y articulaciones se inicializan con la posición de reposo.

Al igual que con las matrices convencionales, el contenido de dos o más objetos `BlenderPoses` puede concatenarse para crear uno nuevo con todas las configuraciones de postura de los objetos originales. Sin embargo, esto sólo es posible en el caso de que los objetos contengan el mismo número de articulaciones y exclusivamente a lo largo de la primera dimensión. Es decir, varios objetos `BlenderPoses` sólo se pueden concatenar a lo largo de las configuraciones de postura, generando un nuevo objeto `BlenderPoses`

```
>> NBP = cat(1, BP1, BP2);
```

El número total de configuraciones de postura de un objeto `BlenderPoses` se puede conocer mediante el método `length()`, como si de un vector de `MATLAB` se tratara

```
>> LEN = length(BP);
```

mientras que para obtener el número de articulaciones y su lista de nombres se puede utilizar la propiedad `bones`.

```
>> BONENAMES = BP.bones;
>> NUMBONES = length(BP.bones);
```

Puesto que a la hora de acceder a las matrices de las posturas de una articulación concreta ésta debe ser indicada mediante su posición en la lista de nombres de las articulaciones, la clase `BlenderPoses` proporciona el método `findbone()` para conocer dicho índice a partir del nombre de la articulación.

```
>> i = findbone(BP, 'wrist');
```

Métodos para el guardado y recuperación de las posturas

Una de las funcionalidades clave de la clase `BlenderPoses` es la posibilidad de almacenar y recuperar los objetos desde el sistema de archivos. Ésta es una característica fundamental porque, como se verá más adelante, es necesaria para exportar las posturas e importarlas en el programa de modelado `Blender`, con el objetivo de sintetizar las imágenes de las manos.

El método `save()` permite guardar el contenido del objeto `BlenderPoses` en un archivo de nombre `FILENAME`

```
>> BP = save(BP, FILENAME, MODE);
```

en formato HDF[3]. Si bien hubiera sido posible escoger el propio formato de `MATLAB` para guardar la información de las posturas, de hacerlo así pocos programas externos hubieran podido acceder a dicha información. La mayor parte de las librerías y programas no pueden abrir archivos de `MATLAB` de versiones posteriores a la 6, pero hasta la versión 7.3 no se soportaron archivos de más de

[3]HDF son las siglas de *hierarchical data format*, que es el nombre de un conjunto de formatos de archivos y librerías diseñados para almacenar y organizar grandes cantidades de datos. La última versión de este formato —conocida como HDF5— utiliza sólo dos tipos de objetos: los *datasets* —matrices multidimensionales de tipo homogéneo— y grupos —contenedores que puede alojar otros grupos y *datasets*—.

2 GB. En todo caso, dicha versión 7.3 utiliza el formato HDF5, sólo que de forma un tanto particular para poder dar soporte a las singularidades de `MATLAB`. El método `save()` utiliza la función `hd5fwrite()` de `MATLAB` para guardar la lista de nombres de articulaciones y la matriz de posturas en los *datasets* `'/bonenames'` y `'/poses/matrix'`, respectivamente.

De igual forma, el método `load()` permite recuperar los datos guardados por el método `save()` desde un archivo en formato HDF5

```
>> BP = load(BP, FILENAME);
```

Funciones auxiliares

El programa de modelado (Blender) utiliza matrices de transformación de 4×4 para definir las posturas de las articulaciones (véase el apartado B.1 en la página 183), motivo por el cual la clase `BlenderPoses` almacena ese tipo de matrices para cada articulación en cada configuración de la postura. Sin embargo, trabajar con estas matrices no resulta demasiado cómodo cuando lo que se pretende es, por ejemplo, poder indicar fácilmente que una articulación concreta rota cierto número de grados en un eje determinado. Para facilitar la manipulación de las matrices de transformación se desarrollaron una serie de funciones auxiliares, que han sido recopiladas en la tabla 6.2.

6.2.2. Síntesis de imágenes con Blender

La clase `BlenderPoses` descrita en el apartado 6.2.1 incluye el método `render()`, que es capaz de comunicarse con `Blender` para sintetizar las imágenes de la mano en las distintas configuraciones de postura almacenadas en el objeto sobre el que se invoca. El método, como mínimo, necesita que se le especifique la ruta al proyecto de `Blender` `PROJECT_BLENDER` que debe utilizar para la síntesis

```
>> render(BP, PROJECT_BLENDER);
```

Tabla 6.2. Funciones de manipulación de las matrices de transformación según las convenciones de `Blender` (véase el apartado B.1 en la página 183).

Función	Descripción
`makepose`	Genera una matriz de transformación de 4×4 a partir de una matriz 3×3 de rotación, de un vector 3×1 de escala y de un vector 3×1 de traslación.
`mat2rst`	Descompone una matriz de transformación de 4×4 en la matriz 3×3 de rotación, el vector 3×1 de escala y el vector 3×1 de traslación correspondiente.
`dcm2eul`	Convierte una matriz 3×3 de rotación en un vector con los ángulos de Euler.
`eul2dcm`	Convierte un vector con los ángulos de Euler en una matriz 3×3 de rotación.
`rotx`	Construye una matriz 3×3 que describe una rotación en el ángulo especificado del eje X.
`roty`	Construye una matriz 3×3 que describe una rotación en el ángulo especificado del eje Y.
`rotz`	Construye una matriz 3×3 que describe una rotación en el ángulo especificado del eje Y.
`dcm2quat`	Convierte una matriz 3×3 de rotación en un cuaternión.
`quat2dcm`	Convierte un cuaternión en una matriz 3×3 de rotación.
`quatconj`	Calcula el cuaternión conjugado de uno dado.
`quatavg`	Calcula el promedio ponderado de dos o más cuaterniones dados.
`quatslerp`	Interpola dos cuaterniones mediante SLERP.

Tabla 6.3. Opciones para la síntesis de imágenes de la clase `BlenderPoses` de `MATLAB`.

Opción	Descripción
`'BlenderCmd'`	Especifica la ubicación del ejecutable del programa `Blender`. Por defecto se usa `'blender'`.
`'RenderPath'`	Especifica la ruta en la que guardar los archivos con las imágenes generadas. Por defecto se usa el directorio actual del modelo a utilizar para la síntesis.
`'LogFile'`	Especifica el nombre del archivo de registro donde almacenar la salida por pantalla de `Blender` durante la síntesis. Por defecto no se guarda ningún registro.
`'PosesRange'`	Especifica el intervalo de configuraciones de postura que se quiere sintetizar. Por defecto se sintetizan todas las posturas almacenadas en el objeto `BlenderPoses`.
`'BonesRange'`	Especifica el intervalo de articulaciones que debe ser considerado al establecer las posturas del modelo antes de la síntesis. Por defecto se utilizan todas las articulaciones.

En general dicho proyecto contiene el modelo descrito en el capítulo 3 en la página 35. Además al método se le pueden especificar una serie de opciones mediante las parejas de nombre y valor tan comunes en las funciones de `MATLAB`

```
>> render(BP, PROJECT_BLENDER, OPTIONNAME1, OPTIONVALUE1,
    OPTIONNAME2, OPTIONVALUE2,...);
```

En la tabla 6.3 se muestra una breve descripción de todas las opciones soportadas. Independientemente de éstas, al invocar el método `render()` éste debe realizar una serie de tareas:

1. Salvar las posturas en un archivo en formato HDF5. El método `save()` de la clase `BlenderPoses` (véase el apartado 6.2.1 en la página 161) soporta un parámetro de modo que permite indicar si el archivo guardado puede ser utilizado en el futuro para importar los datos de las posturas en `Blender`. Si dicho modo fue especificado y no se han modificado las posturas desde entonces, `render()` utilizará directamente el archivo en el que las posturas

fueron guardadas. En caso contrario guardará las posturas en un archivo temporal utilizando el método `save()` de la clase.

2. Configurar las variables de entorno con las opciones especificadas. Opciones como el nombre del archivo que contiene las posturas o el intervalo a sintetizar se pasan al programa `Blender` mediante variables de entorno, por lo que éstas deben ser configuradas adecuadamente.

3. Ejecutar `Blender` para iniciar la síntesis. En la línea de comandos del programa se le pasan opciones como la ruta en la que debe guardar los archivos con las imágenes generadas, la ruta del archivo con el proyecto de `Blender` que contiene el modelo de la mano y el nombre del *script* de `Blender` que debe ser ejecutado para importar los datos de las posturas e iniciar la síntesis.

Al programa de modelado `Blender` se le pueden añadir nuevas funcionalidades de forma sencilla mediante el uso de *scripts*. Estos *scripts* se programan en `Python`, tienen acceso directo a parte del API del programa y pueden ser incluidos dentro del archivo del proyecto de `Blender`. Cuando el método `render()` de la clase `BlenderPoses` ejecuta a `Blender` también le notifica, a través de las opciones de la línea de comandos, que tras iniciarse debe ejecutar el script `RenderPose.py`.

RenderPose: funcionamiento del script de Blender

Las tareas fundamentales de `RenderPose.py` son establecer la postura de las articulaciones y lanzar la síntesis de las imágenes para cada configuración de la postura. El *script* comienza recuperando los valores de las variables de entorno para obtener el intervalo de configuraciones de postura, el intervalo de articulaciones a posicionar y el nombre del archivo que contiene los datos de las posturas. A continuación `RenderPose.py` accede al contenido de dicho archivo

en formato HDF5 con la ayuda del paquete `Pytables`[4]. A partir de este punto el *script* itera sobre las configuraciones almacenadas en el *dataset* `'/poses/matrix'`. Para cada configuración extrae la matriz 4×4 de postura de cada articulación, averigua el nombre de dichas articulaciones a través de la lista almacenada en el *dataset* `'/bonenames'` y utiliza el API de `Blender` para modificar adecuadamente la postura del modelo de la mano. Cuando todas las articulaciones de una misma configuración han sido modificadas, lanza el proceso de síntesis con la ayuda del API del programa y vuelve a iterar. El proceso se repite hasta que todas las configuraciones de postura del intervalo especificado son sintetizadas.

En general el proceso descrito puede ser relativamente lento. El almacenamiento de los datos en disco en formato HDF5, aparte de que puede requerir una cantidad importante de espacio en disco, suele ser bastante costoso en lo que a tiempo se refiere. Igualmente, una vez lanzado `Blender`, el proceso de síntesis no comienza inmediatamente porque se requieren algunas decenas de minutos para leer el archivo de datos, si esté contiene una cantidad importante de posturas. Finalmente la tarea que más tiempo requiere es el proceso de síntesis en sí mismo. Por ejemplo, la generación de 1 000 000 posturas con 16 articulaciones cada una puede necesitar algo menos de dos semanas.

[4] `Pytables` es un paquete para el manejo de grandes conjuntos de datos jerárquicos. Fundamentalmente permite el acceso a archivos HDF5, encargándose de todas las cuestiones relativas a la gestión de la memoria y de los recursos de disco. Así, un programa sólo tiene que solicitar acceso a los *datasets* concretos que son de su interés, haciéndose el paquete responsable de las operaciones necesarias para ofrecer dicho acceso. Su uso es especialmente interesante en el software de adquisición de datos, simulación, monitorización, sistemas de registro, etc.

Conclusiones

Se ha presentado en esta memoria un trabajo que hace aportaciones a la resolución de los problemas subyacentes al reconocimiento gestual, específicamente en lo que respecta a la identificación de la postura de la mano mediante el uso de técnicas de visión por computador. El combinar ambos aspectos no ha sido escogido al azar sino porque, como se verá a continuación, forman una pareja interesante de estudiar. Por un lado la mano es, de todos miembros del cuerpo humano, el principal órgano para la manipulación física; por lo que resolver el problema del reconocimiento gestual para ella es un requisito para que esta tecnología pueda llegar a ser utilizada de forma rutinaria. Por otro lado, las soluciones que ya existen para el reconocimiento gestual de la mano adolecen de tener un coste relativamente alto y de ser bastante incómodas. No en vano, para manipular objetos del mundo real no hace falta ponerse unos guantes cableados, por lo que lo mismo cabría esperar de aquellas soluciones que nos permitan manipular elementos del mundo virtual. Tal y como se ha comentado, en este sentido las técnicas basadas en visión por computador tienen mucho que decir, pues permiten resolver el problema de la manera más cómoda posible para el usuario, sin introducir elementos externos que pueden obstruir sus movimientos u obligarle a cambiar su manera de interactuar. Por tanto, el reconocimiento gestual de la postura de la mano y la visión por computador van a formar una pareja cada vez más importante de cara al futuro. Sin embargo, como se ha visto, esta combinación no está exenta de problemas, pues la estructura articulada de la mano humana representa todo un reto para las técnicas actuales de visión por

computador.

En el presente trabajo se ha enfocado el problema desde dos puntos de vista diferentes. Por un lado, comprobando cómo pueden ayudar a resolver el problema del reconocimiento gestual algunas técnicas clásicas de visión por computador. Por el otro, se ha optado por el uso de técnicas de reducción de la dimensionalidad, con el objetivo de inferir los parámetros de la postura de la mano, sorteando los problemas de las técnicas clásicas.

Respecto al trabajo sobre las técnicas clásicas, se ha intentado comprobar si los métodos de factorización pueden ayudar a reconstruir la forma y el movimiento de la mano. Entre todas las técnicas posibles se optó por la de Tomasi y Kanade (1992) por ser una de las más utilizadas. Como se ha comentado, este tipo de técnicas están condicionadas al funcionamiento de los algoritmos de seguimiento de puntos. Entre estos algoritmos, uno de los más utilizados y que ha demostrado tener muy buenas prestaciones en distintas condiciones es el KLT. Para comprobar el funcionamiento del KLT en el seguimiento de puntos característicos de una mano se desarrollaron diversos programas con el objetivo de probar y ajustar la técnica. Los resultados obtenidos revelaron que con una cámara y en condiciones de iluminación convencionales era necesario emplear ventanas relativamente grandes; pero que, incluso así, no es posible hacer un seguimiento de las características de la mano de manera fiable. Esto hace que la aplicación de cualquier técnica de factorización sea imposible, sin recurrir a artificios como, por ejemplo, introducir textura mediante unos guantes. Para resolverlo se decidió enfocar el problema desde el punto de vista de la utilización de técnicas de reducción de la dimensionalidad, en lugar de seguir usando técnicas clásicas de visión por computador.

Las técnicas de reducción de la dimensión en el pasado se han mostrado muy potentes para el reconocimiento gestual, cuando se han utilizado para el análisis de datos capturados mediante sensores. Suponiendo que cada imagen no es más que el producto de un proceso controlado por unos parámetros —las posturas— que queremos descubrir, parece razonable el utilizar técnicas de reducción de dimensionalidad que intenten desvelar esos factores subyacentes.

Puesto que la mayor parte de esas técnicas requieren de una cantidad significativa de muestras, que son difíciles de conseguir mediante la grabación de secuencias de vídeo con personas reales, se desarrolló un modelo sintético. Para ello se estudió la mano, tanto desde el punto de vista de su estructura articular como desde el de la piel que le da su apariencia, y se desarrolló un modelo completo de la misma. El modelo, implementado en el programa `Blender`, incluye un modelo cinemático, que reproduce las características de la mano desde el punto de vista del movimiento, y un modelo de apariencia basado en el modelo dicromático de reflexión. Para controlar el modelo cinemático, y así poder sintetizar la mano en diversas posturas, se desarrolló software capaz de construir tanto posturas estáticas como gestos dinámicos realizables por manos reales, en base a los últimos trabajos sobre las restricciones en el movimiento de la mano. El mismo software también permite sintetizar por lotes el modelo, en las configuraciones de postura generadas.

En lugar de enfrentar directamente las técnicas de reducción de la dimensionalidad al problema del reconocimiento gestual, se ha planteado que sería posible su uso como alternativa a las técnicas clásicas de asignación de correspondencias —o de seguimiento de puntos— que anteriormente habían dado problemas. De entre las técnicas de reducción de la dimensionalidad, se optó por el KPCA, que pertenece al grupo de los métodos basados en kernel, los cuales han tenido en los últimos años una importante repercusión en los campos del reconocimiento de patrones y del aprendizaje automático. Además existen trabajos previos que proponen que el KPCA puede ser utilizado tanto para la asignación de correspondencias de objetos rígidos como articulados, siendo éste último caso el de la mano. Independientemente de esto, en este trabajo se han hecho algunas aportaciones de cara a la utilización del KPCA en el problema del reconocimiento gestual de la postura de la mano:

- Se han desarrollado las ecuaciones para el cálculo de las preimágenes cuando el conjunto de datos ha sido centrado por clases durante el KPCA. Este tipo de análisis es el utilizado en la asignación de correspondencias de ob-

jetos articulados, por lo que poder calcular la preimagen es muy interesante para esta clase de problemas. Además el desarrollo se ha realizado tanto para la estimación mediante el método iterativo de Mika *et al.* (1999) —el más clásico— como para un método no interactivo más reciente. Además se ha tenido en cuenta tanto funciones de kernel gaussianas como polinomiales.

- Se ha propuesto una forma de calcular la distancia de Kullback-Leibler entre dos conjuntos de datos en sus subespacios de componentes principales. Esto, por ejemplo, es interesante de cara a comparar las distribuciones de puntos pertenecientes a diferentes clases, como ocurre en la asignación de correspondencias de objetos articulados.

- Se ha propuesto una estimador contraído para la matriz de covarianza de los puntos proyectados en su subespacio de componentes principales.

Una tarea muy común en los algoritmos de asignación de correspondencias es la recuperación de los vecinos más próximos. Teniendo en cuenta que el volumen de datos puede ser muy alto —ya sean píxeles o imágenes completas— y que los espacios donde residen dichos datos pueden ser de dimensión bastante elevada, se vio que era necesario utilizar algún tipo de estructura que permitiera hacer consultas en tiempos razonables. El resultado es la implementación de una *toolbox* para `MATLAB` del algoritmo Cover-Tree. Esta estructura es ideal cuando los datos residen en un espacio de cierta dimensión mayor que la dimensión intrínseca del conjunto de datos.

Desarrolladas las herramientas necesarias, se ha intentado comprobar si una técnica de reducción de la dimensionalidad como KPCA puede servir para resolver el problema de la asignación de correspondencias, cuando la textura del objeto no tiene información suficiente para utilizar técnicas clásicas como el KLT. A partir de la propuesta de Wang y Hancock (2006) se han realizado diversas baterías de pruebas, tanto con modelos rígidos como articulados. Se ha estudiado el efecto de la elección del tipo de función de kernel y sus parámetros sobre

el tamaño del subespacio de componentes principales. También se ha intentado inferir qué tipo de información atrapa el KPCA a través del cálculo de las preimágenes y si ésta era la adecuada para resolver el problema de la asignación de las correspondencias. En este sentido, se ha podido ver como los segmentos de mayor extensión del objeto articulado tienden a estar representados en las primeras dimensiones, las de mayores autovalores. Por tanto, para preservar la información de los segmentos de menor extensión —por ejemplo, las falanges de los dedos— es necesario utilizar subespacios de mayor número de dimensiones que lo estimado mediante el análisis de autovalores estándar.

También se ha comprobado que el subespacio de componentes principales obtenido mediante KPCA no depende del orden pero sí de los puntos concretos escogidos. Eso es un gran inconveniente, puesto que significa que para obtener siempre el mismo subespacio y comparar hay que seleccionar los mismos puntos en ambas imágenes. Es decir, que para utilizar esta técnica de correspondencia las imágenes deben tener información suficiente como para poder extraer puntos característicos que puedan ser identificados en ambas. Pero si ocurriera así con las imágenes capturadas de la mano, es muy probable que los resultados de las pruebas con el KLT hubieran sido muy diferentes. Para resolver estos problemas con los subespacios de componentes principales se propuso una variante del algoritmo utilizando KCCA.

La propuesta de Wang y Hancock (2006) introduce un proceso de relajación para ajustar las probabilidades de que cada punto pertenezca a un segmento del objeto. Estas probabilidades deben ser inicializadas con una estimación de las mismas. Las experiencias realizadas han puesto de manifiesto que para que la asignación se realice con éxito, las probabilidades de los puntos de una de las imágenes deben ser parecidas a las de la otra. Por tanto, es necesario disponer de una manera de asignar inicialmente las probabilidades a la segunda imagen, de forma que sean lo más parecidas posibles a las probabilidades finales. Se propuso hacerlo considerando inicialmente el objeto articulado como rígido, utilizando posteriormente la asignación de correspondencias resultante como fuente para generar las probabilidades iniciales. En todo caso se comprobó

que esta solución no daba resultados mucho mejores que asignando el etiquetado inicial en base a los vecinos más próximos en $\mathbb{R}^2$. Eso significa que la técnica no funciona si ambas imágenes difieren apreciablemente y que, seguramente, el KPCA no está atrapando información más allá de la relativa a las coordenadas de los puntos. En estas circunstancias el proceso de relajación poco puede hacer, aparte de suavizar los valores de probabilidad. En todo caso queda abierta la puerta de intentar aprender una matriz de coeficientes de compatibilidad adecuada que ayude a asignar las etiquetas de manera más inteligente, mitigando los problemas de una mala inicialización.

6.3. Líneas abiertas

Una vez concluido el trabajo son varias las líneas que han quedado abiertas:

- Utilizar otras características en lugar de las posiciones de puntos escogidos al azar. Teniendo en cuenta que la elección de los puntos en un factor determinante en el subespacio generado, podría ser interesante seleccionar otro tipo de características que, al menos parcialmente, se pudieran identificar entre imágenes. Por ejemplo, puntos de los bordes que tengan alguna propiedad concreta.

- La dificultad del problema se reduciría notablemente si se restringiera su generalidad. Por ejemplo, imponiendo el uso de guantes con marcadores, lo que permitiría utilizar incluso algoritmos clásicos como KLT.

- Un aspecto que no se ha explotado es la posibilidad de utilizar directamente las imágenes, en lugar de extraer puntos de interés de las mismas. En principio cada imagen se podría vectorizar para constituir un punto en el espacio de las imágenes. Dicho espacio podría ser muestreado mediante la generación de manos sintéticas en distintas posturas, para aplicar el KPCA sobre todo el conjunto de datos. En teoría el análisis debería de extraer

los factores subyacentes a la variabilidad de las imágenes, que deben estar de alguna manera relacionados con las posturas.

- Debido al coste que tiene hacer el KPCA sobre un gran conjunto de imágenes vectorizadas, se podría crear un modelo de apariencia que pudiera ser ajustado sobre cada una de las imágenes. Los modelos ajustados vivirían en un espacio de dimensión mucho menor que el de sus imágenes, por lo que hacer un KPCA o aplicar cualquier otro tipo de comparación tendría mucho menor coste.
- Una posibilidad interesante es sustituir el KPCA por el KCCA. En algunos artículos recientes se propone vectorizar las imágenes para hacer un KCCA junto con los parámetros de postura con los que se generaron. El KCCA devuelve un espacio donde imágenes y parámetros están máximamente correlados, lo que podría ser un buen punto de partida para el reconocimiento gestual.

Apéndice A

Cover-Tree toolbox

A continuación se resumen las operaciones soportadas por `covertree_call`, el ejecutable de `MATLAB` (MEX), y los métodos y propiedades de la clase `CoverTree` de la *toolbox* de Cover-Tree para `MATLAB`.

A.1. Operaciones soportadas por el MEX

batch_create Construye un árbol de Cover-Tree insertando los puntos especificados.

- **Argumentos de entrada:**
  Un punto individual o una matriz *cell* de puntos.
- **Argumentos de salida:**
  Un manejador que identifica el árbol de Cover-Tree creado.

insert Inserta nuevos puntos en el árbol de Cover-Tree.

- **Argumentos de entrada:**
  Un punto individual o una matriz *cell* de puntos.

k_nearest_neighbor Busca los k vecinos más próximos a los puntos consultados.

- **Argumentos de entrada:**
  1. El manejador del árbol de Cover-Tree que contiene los puntos a consultar.

2. El número de vecinos k a buscar.

- **Argumentos de salida:**
  1. Una matriz *cell* con un elemento para cada punto consultado. Cada uno de dichos elementos es otra matriz *cell* con $k+1$ elementos. El primero contiene el punto consultado y el resto los k vecinos más próximos.
  2. Una matriz *cell* con un elemento para cada punto consultado. Cada uno de dichos elementos es un vector de dimensión $k+1$ con las distancias entre los vecinos más próximos y el punto consultado, en el mismo orden que en el primer argumento de salida.

epsilon_nearest_neighbor Busca los ϵ-aproximados vecinos más próximos a los puntos consultados.

- **Argumentos de entrada:**
  1. El manejador del árbol de Cover-Tree que contiene los puntos a consultar.
  2. El valor de ϵ, que define el radio alrededor del punto consultado en el que tienen que encontrarse los vecinos más próximos.
- **Argumentos de salida:**
  1. Una matriz *cell* con un elemento para cada punto consultado. Cada uno de dichos elementos es otra matriz *cell*. El primer elemento de dicha matriz contiene el punto consultado y el resto los ϵ-aproximados vecinos más próximos.
  2. Una matriz *cell* con un elemento para cada punto consultado. Cada uno de dichos elementos es un vector con las distancias entre los vecinos más próximos y el punto consultado, en el mismo orden que en el primer argumento de salida.

unequal_nearest_neighbor Busca el vecino más próximo aunque no igual a los puntos consultados.

- **Argumentos de entrada:**
  El manejador del árbol de Cover-Tree que contiene los puntos a consultar.
- **Argumentos de salida:**
  1. Una matriz *cell* con un elemento para cada punto consultado. Cada uno de dichos elementos es otra matriz *cell* de dos elementos. El primero contiene el punto consultado y el otro al vecino más próximo.
  2. Una matriz *cell* con un elemento para cada punto consultado. Cada uno de dichos elementos es un vector con las distancias entre el vecino más próximo y el punto consultado, en el mismo orden que en el primer argumento de salida.

breadth_dist Devuelve el número de hijos para cada nodo del árbol de Cover-Tree

- **Argumentos de salida:**
  Un vector con tantos elementos como nodos que contiene el número de hijos de cada uno.

depth_dist Devuelve el número de niveles implícitos entre cada nodo y el nodo raíz del árbol de Cover-Tree

- **Argumentos de salida:**
  Un vector con tantos elementos como nodos con la distancia en niveles implícitos desde cada nodo explícito al nodo raíz.

height_dist Devuelve el número máximo de niveles explícitos por debajo de cada nodo del árbol de Cover-Tree

- **Argumentos de salida:**
  Un vector con tantos elementos como nodos con el número máximo de niveles explícitos por debajo de cada nodo.

load Carga el árbol de Cover-Tree desde el archivo especificado.

- **Argumentos de entrada:**
  1. Nombre del archivo desde el que cargar la estructura de datos.
  2. `"text"` para cargar la estructura en formato texto o `"binary"` para hacerlo en formato binario, que es la opción por defecto.

save Guardar el árbol de Cover-Tree en un archivo.

- **Argumentos de entrada:**
  1. Nombre del archivo en el que guardar la estructura de datos.
  2. `"text"` para guardar la estructura en formato texto o `"binary"` para hacerlo en formato binario, que es la opción por defecto.

delete Libera la memoria y el resto de los recursos reservados por el árbol de Cover-Tree.

A.2. Métodos de la clase CoverTree

CoverTree Construye un nuevo objeto `CoverTree`, insertando los puntos especificados en el primer argumento y configurando el objeto en base a las restante opciones.

- **Argumentos de entrada:**
  1. Un punto individual o una matriz *cell* de puntos (opcional).
  2. `'DistanceFcn'` seguido de la función de distancia a utilizar (opcional).
  3. `'PreSerializeFcn'` seguido de la función para preprocesar los puntos antes de la serialización (opcional).
  4. `'PostDeserializeFcn'` seguido de la función para posprocesar los puntos después de la deserialización (opcional).

- **Argumentos de salida:**
  Un objeto de la clase `CoverTree`.

insert Inserta nuevos puntos en el árbol del objeto `CoverTree`.

- **Argumentos de entrada:**
  1. Un punto individual o una matriz *cell* de puntos.

kNN Busca los k vecinos más próximos a los puntos consultados.

- **Argumentos de entrada:**
  1. Un objeto `CoverTree` que contiene los puntos a consultar.
  2. El número de vecinos k a buscar.
- **Argumentos de salida:**
  1. Una matriz *cell* con un elemento para cada punto consultado. Cada uno de dichos elementos es otra matriz *cell* con $k+1$ elementos. El primero contiene el punto consultado y el resto los k vecinos más próximos.
  2. Una matriz *cell* con un elemento para cada punto consultado. Cada uno de dichos elementos es un vector de dimension $k+1$ con las distancias entre los vecinos más próximos y el punto consultado, en el mismo orden que en el primer argumento de salida.

epsilonNN Busca los ϵ-aproximados vecinos más próximos a los puntos consultados.

- **Argumentos de entrada:**
  1. Un objeto `CoverTree` que contiene los puntos a consultar.
  2. El valor de ϵ, que define el radio alrededor del punto consultado en el que tienen que encontrarse los vecinos más próximos.
- **Argumentos de salida:**

1. Una matriz *cell* con un elemento para cada punto consultado. Cada uno de dichos elementos es otra matriz *cell*. El primer elemento de dicha matriz contiene el punto consultado y el resto los ϵ-aproximados vecinos más próximos.
2. Una matriz *cell* con un elemento para cada punto consultado. Cada uno de dichos elementos es un vector con las distancias entre los vecinos más próximos y el punto consultado, en el mismo orden que en el primer argumento de salida.

unequalNN Busca el vecino más próximo aunque no igual a los puntos consultados.

- **Argumentos de entrada:**
  Un objeto `CoverTree` que contiene los puntos a consultar.
- **Argumentos de salida:**
  1. Una matriz *cell* con un elemento para cada punto consultado. Cada uno de dichos elementos es otra matriz *cell* de dos elementos. El primero contiene el punto consultado y el otro al vecino más próximo.
  2. Una matriz *cell* con un elemento para cada punto consultado. Cada uno de dichos elementos es un vector con las distancias entre el vecino más próximo y el punto consultado, en el mismo orden que en el primer argumento de salida.

load Carga el árbol del Cover-Tree desde el archivo especificado. Se debe invocar sobre un objeto `CoverTree` inicializado sin puntos.

- **Argumentos de entrada:**
  1. Nombre del archivo desde el que cargar la estructura de datos.
  2. `'text'` para cargar la estructura en formato texto o `'binary'` para hacerlo en formato binario, que es la opción por defecto (opcional).

save Guardar el árbol del Cover-Tree del objeto en un archivo.

- **Argumentos de entrada:**
  1. Nombre del archivo en el que guardar la estructura de datos.
  2. `'text'` para guardar la estructura en formato texto o `'binary'` para hacerlo en formato binario, que es la opción por defecto (opcional).

delete Destruye el objeto `CoverTree`.

A.3. Propiedades de la clase CoverTree

Todas las propiedades públicas de la clase `CoverTree` son de sólo lectura.

BreadthDistances Contiene un vector con tantos elementos como nodos que contiene el número de hijos de cada uno.

DepthDistances Contiene un vector con tantos elementos como nodos con la distancia en niveles implícitos desde cada nodo explícito al nodo raíz.

HeightDistances Contiene un vector con tantos elementos como nodos con el número máximo de niveles explícitos por debajo de cada nodo.

Apéndice

B

Modelo cinemático de la mano en Blender

En el capítulo 3 en la página 35 se comentó la estructura de la mano humana y sus grados de libertad, así como las propuestas de los distintos autores para modelarla (Albrecht *et al.*, 2003; Dragulescu *et al.*, 2007; Vardy, 1998; Wu y Huang, 1999a; Yasumuro *et al.*, 1997). En el mismo capítulo se explicó como un modelo de mano completo fue extraído del programa `MakeHuman`—que está basado en el de Dragulescu *et al.* (2007) y Sturman (1992)— importado en el programa `Blender` de modelado 3D y modificado en base a las propuestas comentadas para adaptarlo a las necesidades concretas del problema del reconocimiento gestual (véase el apartado 3.1.2 en la página 42). A continuación se exponen algunos detalles complementarios del modelo cinemático que finalmente fue utilizado con `Blender` para la síntesis de las imágenes de la mano, con el objetivo de tener una descripción completa del mismo.

B.1. Convenciones de Blender

Entre los diversos tipos de elementos que pueden conformar un modelo de `Blender`, la *armadura* es el que se encarga de definir el modelo cinemático de los objetos no rígidos. Las armaduras actúan como un esqueleto. Están compuestas por huesos que pueden ser configurados de diferentes maneras. Cada hueso tie-

ne una *raíz* y una *cola* que son los puntos extremos del hueso. De manera que todos los huesos comienza por su raíz, que define su posición, y terminan en su cola. Aparte del sistema de referencia global de la escena, a partir del cual se puede indicar la ubicación de cualquier elemento, cada hueso tiene su propio sistema de referencia local centrado en su raíz. Así, un hueso en una posición arbitraria $\boldsymbol{v}$ respecto al sistema de referencia global está ubicado en el origen $\mathbf{0}$ respecto a su sistema de referencia local.

Por convención, el sistema de referencia local situado en la raíz del primer hueso de la cadena cinemática de una armadura es el sistema de referencia global. Cada hueso tiene una matriz 3×3 que se utiliza para rotar su sistema de referencia, de manera que el eje Y se alinee longitudinalmente con el hueso, es decir, apunte en la dirección de la cola. Entonces el nuevo sistema de referencia, resultante de aplicar la rotación sobre el sistema de referencia local, es trasladado a la cola del hueso, definiéndose como el sistema de referencia local del siguiente hueso de la cadena cinemática. De esta manera se pueden ir recorriendo los huesos y calcular sus sistemas de referencia a lo largo de toda la cadena cinemática. Un ejemplo de esto puede observarse en la figura 3.2 en la página 44 para el caso concreto del modelo de la mano. Generalmente la raíz de un hueso se sitúa sobre la cola del anterior, en el origen de su sistema de referencia local. Sin embargo en ocasiones esto no ocurre, por lo que cada hueso tiene un vector de dimensión 3 que indica la localización de la raíz respecto a dicho sistema de referencia.

En general `Blender` sigue la convención[1] X-Y-Z para componer matrices de rotación a partir de los ángulos de Euler, por lo que cada hueso puede ser descrito en base a la siguiente transformación en coordenadas homogéneas desde el $(n-1)$-ésimo al n-ésimo sistema de referencia

$$^{n-1}\mathbf{A}_n = \mathbf{R}(\phi, \theta, \psi)\mathbf{T}_Y(d) = \mathbf{R}_X(\phi)\mathbf{R}_Y(\theta)\mathbf{R}_Z(\psi)\mathbf{T}_Y(d) \tag{B.1}$$

[1] Aparte de la convención comentada, `Blender` sigue la de utilizar vectores fila en lugar de vectores columna. Sin embargo, para mantener la consistencia de la notación se han adaptado las expresiones para seguir utilizando vectores columna.

donde $\mathbf{R}_X(\phi)$, $\mathbf{R}_Y(\theta)$, $\mathbf{R}_Z(\psi)$ y $\mathbf{T}_Y(d)$ son las matrices de rotación en el eje X, en el eje Y, en el eje Z y traslación en el eje Y, respectivamente

$$\mathbf{R}_X(\phi) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi & 0 \\ 0 & \sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.2}$$

$$\mathbf{R}_Y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.3}$$

$$\mathbf{R}_Z(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 & 0 \\ \sin\psi & \cos\psi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.4}$$

$$\mathbf{T}_Y(d) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & d \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.5}$$

Esto permite obtener las componentes de la matriz de transformación ${}^{n-1}\mathbf{A}_n$ sustituyendo (B.2), (B.3), (B.4) y (B.5) en (B.1)

$$a_{11} = \cos\theta\cos\psi \tag{B.6}$$

$$a_{12} = -\cos\theta\sin\psi \tag{B.7}$$

$$a_{13} = \sin\theta \tag{B.8}$$

$$a_{14} = -d\cos\theta\sin\psi \tag{B.9}$$

$$a_{21} = \sin\phi\sin\theta\cos\psi + \cos\phi\sin\psi \tag{B.10}$$

$$a_{22} = -\sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi \tag{B.11}$$
$$a_{23} = -\sin\phi\cos\theta \tag{B.12}$$
$$a_{24} = d(-\sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi) \tag{B.13}$$
$$a_{31} = -\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi \tag{B.14}$$
$$a_{32} = \cos\phi\sin\theta\sin\psi + \sin\phi\cos\psi \tag{B.15}$$
$$a_{33} = \cos\phi\cos\theta \tag{B.16}$$
$$a_{34} = d(\cos\phi\sin\theta\sin\psi + \sin\phi\cos\psi) \tag{B.17}$$
$$a_{41} = 0,\ a_{42} = 0,\ a_{43} = 0,\ a_{44} = 1$$

Además, puesto que para ganar flexibilidad `Blender` permite que la raíz de un hueso no esté siempre sobre la cola del anterior, puede ser necesario trasladar el sistema de referencia de la cola del hueso previo a la raíz del posterior

$$\mathbf{T}(x,y,z) = \begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dx \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.18}$$

Aparte de las matrices que definen la posición y orientación, cada hueso tiene una matriz 4×4 de postura en coordenadas homogéneas respecto al sistema de referencia local. Las matrices vistas hasta el momento definen la posición de reposo de los huesos, mientras que cada una de estas matrices de postura definen como un hueso debe moverse, rotar y escalarse respecto a la postura de reposo. Puesto que en el modelo cinemático de la mano sintética desarrollada todos los movimientos posibles son rotaciones, sólo es necesario encadenar el producto de las matrices (B.2), (B.3), (B.4) para calcular la matriz de la postura a partir de los ángulos de Euler correspondientes

$$\mathbf{P}_{n-1} = \mathbf{P}_{n-1}(\phi',\theta',\psi') = \mathbf{R}_X(\phi')\mathbf{R}_Y(\theta')\mathbf{R}_Z(\psi') \tag{B.19}$$

Esta expresión debe ser aplicada después de la rotación y antes de la traslación en (B.1), por lo que la matriz de transformación completa para pasar del $(n-1)$-ésimo al n-ésimo sistema de referencia ${}^{n-1}\mathbf{A}_n$ sería

$$ {}^{n-1}\mathbf{A}_n = \mathbf{R}(\phi,\theta,\psi)\mathbf{P}_{n-1}(\phi',\theta',\psi')\mathbf{T}_Y(d) \tag{B.20} $$

B.2. Ecuaciones del modelo cinemático

A continuación se exponen las ecuaciones del modelo cinemático de la mano sintética desarrollada en base a las convenciones de `Blender` comentadas en el apartado B.1. Estas ecuaciones, junto con el esquema de la figura 3.2 en la página 44 y los detalles comentados en el apartado 3.1.2 en la página 42 acerca de las restricciones impuestas, describen el modelo cinemático por completo.

En el modelo desarrollado existen distintas cadenas cinemáticas, cada una de las cuales termina en las falanges distales de los dedos. Todas tienen su origen en el codo, donde se sitúa el sistema de referencia global —que se denota por O—. Realmente, el modelo incluye tanto la articulación del codo como la del hombro pero, puesto que no fueron utilizadas, no serán incluidas en esta descripción. Por tanto, las ecuaciones para pasar del sistema de referencia global al sistema de referencia de la muñeca —o articulación radiocarpal (RC)— son

$$ \begin{aligned} {}^{O}\mathbf{A}_{RC} &= \mathbf{R}_O\mathbf{T}_O \\ \mathbf{R}_O &= \mathbf{R}_X(-2^\circ)\mathbf{R}_Y(2^\circ)\mathbf{R}_Z(-88^\circ) \\ \mathbf{T}_O &= \mathbf{T}_Y(2{,}53) \end{aligned} \tag{B.21} $$

donde no aparece la matriz de postura $\mathbf{P}_{RC}$ puesto que se está considerando que no hay articulación en el codo. Como se puede apreciar en la figura 3.2 en la página 44, unido a la articulación de la muñeca hay un único hueso W que hace de metacarpo para toda la palma de la mano. Por lo que la matriz de transformación para pasar del sistema de referencia de la articulación RC a la del extremo

de W es

$$\begin{aligned} {}^{RC}\mathbf{A}_W &= \mathbf{R}_{RC}\mathbf{P}_{RC}\mathbf{T}_{RC} \\ \mathbf{R}_{RC} &= \mathbf{R}_X(20^\circ)\mathbf{R}_Y(-14^\circ)\mathbf{R}_Z(117^\circ) \\ \mathbf{P}_{RC} &= \mathbf{R}_X(\phi_{AA})\mathbf{R}_Y(\theta_C)\mathbf{R}_Z(\psi_F) \\ \mathbf{T}_{RC} &= \mathbf{T}_Y(0{,}89) \end{aligned} \tag{B.22}$$

donde se ha utilizado en la matriz de postura el subíndice F para hacer referencia al grado de libertad del movimiento de flexión-extensión —en el eje Z— el subíndice AA para el de abducción-aducción —en el eje X— y el subíndice C para el de circunducción —en el eje Y— que son los tres grados de libertad de la articulación esférica de la muñeca. El hueso W está enlazado con el interfalangeal proximal (IFP) de cada dedo, excepto en el caso del pulgar que lo está con el metacarpo. Éste es uno de los aspectos en los que el modelo difiere de que cabría esperar de una mano humana real. En lugar de que cada dedo tenga su propio metacarpo, el hueso W hace de metacarpo para todos los dedos menos para el pulgar, que tiene el suyo propio aunque no está únido a la muñeca sino al hueso W, haciendo de articulación trapeciometacarpal (TM). En la figura 3.2 se puede observar que en la cola de W se define el sistema de referencia local de todos los huesos que están conectados con él. Pero, puesto que estos mismos huesos no tienen su raíz en la cola de W, es necesario trasladar el sistema de referencia antes de continuar con las transformaciones definidas para dichos huesos. Por ejemplo, la matriz de transformación para pasar del sistema de referencia del extremo de W a la articulación TM del pulgar sería

$${}^{W}\mathbf{A}_{TM} = T(0{,}19, -0{,}76, 0{,}32) \tag{B.23}$$

mientras que las matrices para pasar a los sistemas de referencia de las articulaciones metacarpofalangeal (MF) de cada uno de los dedos se serían

$${}^{W}\mathbf{A}_{MF(2)} = T(0{,}10, -0{,}10, 0{,}55) \tag{B.24}$$

$$
\begin{aligned}
{}^{W}\mathbf{A}_{MF(3)} &= T(0{,}07, -0{,}02, 0{,}39) && \text{(B.25)}\\
{}^{W}\mathbf{A}_{MF(4)} &= T(0, -0{,}01, 0{,}20) && \text{(B.26)}\\
{}^{W}\mathbf{A}_{MF(5)} &= T(0, 0, 0) && \text{(B.27)}
\end{aligned}
$$

donde se han utilizado los números asignados a los dedos en la figura 3.1 en la página 36 para indicar la articulación MF concreta a la que hace referencia cada expresión. Igualmente se calcula la matriz de transformación desde el sistema de referencia de la articulación TM al de la articulación MF del pulgar

$$
\begin{aligned}
{}^{TM}\mathbf{A}_{MF(1)} &= \mathbf{R}_{TM}\mathbf{P}_{TM}\mathbf{T}_{TM}\\
\mathbf{R}_{TM} &= \mathbf{R}_X(77^\circ)\mathbf{R}_Y(29^\circ)\mathbf{R}_Z(-30^\circ)\\
\mathbf{P}_{TM} &= \mathbf{R}_X(\phi_{AA})\mathbf{R}_Z(\psi_F)\\
\mathbf{T}_{TM} &= \mathbf{T}_Y(0{,}38) && \text{(B.28)}
\end{aligned}
$$

Cada dedo, con la excepción del pulgar, tienen 3 huesos denominados falanges proximal, media y distal. Éstos están unidos mediante las articulaciones interfalangeal proximal (IFP) e interfalangeal distal (IFD). En el caso del pulgar no existe la falange media, por lo que sólo existe una articulación interfalangeal (IF). En B.1 se proporcionan las expresiones para las matrices **R**, **P** y **T** para todos los sistemas de referencia locales y todos los dedos, desde el de las articulaciones MF hasta el de los extremos de los dedos.

Tabla B.1. Matrices de transformación **R**, **P** y **T** del modelo cinemático de las articulaciones de los dedos.

Articulación	$\mathbf{R} =$	$\mathbf{P} =$	$\mathbf{T} =$
$MF(1)$	$\mathbf{R}_X(-23°)\mathbf{R}_Y(-21°)\mathbf{R}_Z(-21°)$	$\mathbf{R}_Z(\psi_F)$	$\mathbf{T}_Y(0{,}24)$
$MF(2)$	$\mathbf{R}_X(31°)\mathbf{R}_Y(14°)\mathbf{R}_Z(-37°)$	$\mathbf{R}_X(\phi_{AA})\mathbf{R}_Z(\psi_F)$	$\mathbf{T}_Y(0{,}33)$
$MF(3)$	$\mathbf{R}_X(19°)\mathbf{R}_Y(11°)\mathbf{R}_Z(-36°)$	$\mathbf{R}_X(\phi_{AA})\mathbf{R}_Z(\psi_F)$	$\mathbf{T}_Y(0{,}34)$
$MF(4)$	$\mathbf{R}_X(16°)\mathbf{R}_Y(11°)\mathbf{R}_Z(-37°)$	$\mathbf{R}_X(\phi_{AA})\mathbf{R}_Z(\psi_F)$	$\mathbf{T}_Y(0{,}36)$
$MF(5)$	$\mathbf{R}_X(15°)\mathbf{R}_Y(10°)\mathbf{R}_Z(-29°)$	$\mathbf{R}_X(\phi_{AA})\mathbf{R}_Z(\psi_F)$	$\mathbf{T}_Y(0{,}28)$
$IFP(2)$	$\mathbf{R}_X(-7°)\mathbf{R}_Y(-7°)\mathbf{R}_Z(7°)$	$\mathbf{R}_Z(\psi_F)$	$\mathbf{T}_Y(0{,}25)$
$IFP(3)$	$\mathbf{R}_X(-2°)\mathbf{R}_Y(-2°)\mathbf{R}_Z(8°)$	$\mathbf{R}_Z(\psi_F)$	$\mathbf{T}_Y(0{,}26)$
$IFP(4)$	$\mathbf{R}_X(0°)\mathbf{R}_Y(0°)\mathbf{R}_Z(11°)$	$\mathbf{R}_Z(\psi_F)$	$\mathbf{T}_Y(0{,}25)$
$IFP(5)$	$\mathbf{R}_X(-1°)\mathbf{R}_Y(-1°)\mathbf{R}_Z(3°)$	$\mathbf{R}_Z(\psi_F)$	$\mathbf{T}_Y(0{,}21)$
$IF(1)$	$\mathbf{R}_X(0°)\mathbf{R}_Y(0°)\mathbf{R}_Z(0°)$	$\mathbf{R}_Z(\psi_F)$	$\mathbf{T}_Y(0{,}24)$
$IFD(2)$	$\mathbf{R}_X(0°)\mathbf{R}_Y(0°)\mathbf{R}_Z(0°)$	$\mathbf{R}_Z(\psi_F)$	$\mathbf{T}_Y(0{,}25)$
$IFD(3)$	$\mathbf{R}_X(0°)\mathbf{R}_Y(0°)\mathbf{R}_Z(0°)$	$\mathbf{R}_Z(\psi_F)$	$\mathbf{T}_Y(0{,}26)$
$IFD(4)$	$\mathbf{R}_X(0°)\mathbf{R}_Y(0°)\mathbf{R}_Z(0°)$	$\mathbf{R}_Z(\psi_F)$	$\mathbf{T}_Y(0{,}25)$
$IFD(5)$	$\mathbf{R}_X(0°)\mathbf{R}_Y(0°)\mathbf{R}_Z(0°)$	$\mathbf{R}_Z(\psi_F)$	$\mathbf{T}_Y(0{,}21)$

Bibliografía

Las siguientes referencias bibliográficas se presentan en orden alfabético por autor. Las referencias con más de un autor aparecen ordenadas en base al primero de los mismos.

A. Aizerman, E. Braverman y L. Rozoner. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821 – 837, 1964.

S. Akaho. A kernel method for canonical correlation analysis. En *Proceedings of International Meeting on Psychometric Society*, Osaka, 2001. URL `http://uk.arxiv.org/abs/cs/0609071v2`.

I. Albrecht, J. Haber y H.-P. Seidel. Construction and animation of anatomically based human hand models. En *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, págs. 98 – 109, San Diego, California, 2003. Eurographics Association.

R. R. Anderson, J. Hu y J. A. Parrish. Optical radiation transfer in the human skin and applications in in vivo remittance spectroscopy. En *Bioengineering and the Skin*, eds. R. Marks y P. A. Payne, cap. 28, págs. 253–265. MTP Press Limited, 1981.

M. Arulampalam, S. Maskell, N. Gordon y T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions*

on Signal Processing, 50:174–188, 2002. ISSN 1053587X. DOI: 10.1109/78.978374. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=978374.

F. R. Bach y M. I. Jordan. Kernel independent component analysis. *The Journal of Machine Learning Research*, 2003. ISSN 1533-7928.

G. H. Bakir, J. Weston y B. Schölkopf. Learning to find pre-images. En *Advances in Neural Information Processing Systems*, eds. S. Thrun, L. Saul y B. Schölkopf, págs. 449–456, Cambridge, 2004. MIT Press. DOI: 10.1.1.68.5164. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.68.5164.

M. Bastioni y M. Flerackers. Makehuman, 2009. URL http://www.makehuman.org/.

P. Beckmann y A. Spizzichino. *The Scattering of Electromagnetic Waves from Rough Surfaces*. Artech Print on Demand, 1987.

M. Belkin y P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. En *Advances in Neural Information Processing Systems 14*, págs. 585–591. MIT Press, 2001. DOI: 10.1.1.19.9400. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.19.9400.

J. L. Bentley. Multidimensional divide-and-conquer. *Communications of the ACM*, 23:214 – 229, 1980. ISSN 0001-0782.

A. Beygelzimer, S. Kakade y J. Langford. Cover trees for nearest neighbor. En *Proceedings of the 23rd International Conference on Machine Learning*, págs. 97–104, Pittsburgh, Pennsylvania, 2006a. ACM.

A. Beygelzimer, S. Kakade y J. Langford. Cover trees for nearest neighbor, 2006b. URL http://hunch.net/~jl/projects/cover_tree/paper/paper.ps.

A. Beygelzimer, S. Kakade y J. Langford. Cover tree, 2006c. URL http://hunch.net/~jl/projects/cover_tree/cover_tree.html.

F. Blender. Blender, 2009. URL http://www.blender.org/.

G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

L. Bretzner, I. Laptev y T. Lindeberg. Hand gesture recognition using multi-scale colour features, hierarchical models and particle filtering. En *Proceedings of Fifth IEEE International Conference on Automatic Face Gesture Recognition*, vol. 32, págs. 423–428. IEEE, 2002. ISBN 0-7695-1602-5. DOI: 10.1109/AFGR.2002.1004190. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1004190.

W. A. Burkhard y R. M. Keller. Some approaches to best-match file searching. *Communications of the ACM*, 16:230–236, 1973. ISSN 0001-0782. DOI: http://doi.acm.org/10.1145/362003.362025.

R. Cattell. The scree test for the number of factors. *Multivariate Behavioral Research*, 1:245–276, 1966. DOI: 10.1207/s15327906mbr0102_10. URL http://dx.doi.org/10.1207/s15327906mbr0102_10.

R. L. Cook y K. E. Torrance. A reflectance model for computer graphics. En *International Conference on Computer Graphics and Interactive Techniques*, vol. 15, págs. 307–316, Dallas, Texas, United States, 1981. ACM.

T. F. Cootes, C. J. Taylor, D. H. Cooper y J. Graham. Training models of shape from sets of examples. En *Proceedings of British Machine Vision Conference*, págs. 9–18. Springer-Verlag, 1992. DOI: 10.1.1.18.1478. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.18.1478.

T. F. Cootes, C. J. Taylor, D. H. Cooper y J. Graham. Active shape models—their training and application. *Computer Vision and Image Understanding*, 61:38–59, 1995. ISSN 1077-3142.

Z. Daoqiang, Z. Zhi-Hua y C. Songcan. Adaptive kernel principal component analysis with unsupervised learning of kernels. En *Sixth International Conference on Data Mining (ICDM'06)*, págs. 1178–1182, Hong Kong, 2006. IEEE.

ISBN 0-7695-2701-7. DOI: 10.1109/ICDM.2006.14. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4053175.

D. Dragulescu, V. Perdereau, M. Drouin, L. Ungureanu y K. Menyhardt. 3D active workspace of human hand anatomical model. *Biomedical engineering online*, 6:15, 2007. ISSN 1475-925X. DOI: 10.1186/1475-925X-6-15. URL http://www.ncbi.nlm.nih.gov/pubmed/17472756.

G. D. Finlayson y G. Schaefer. Solving for colour constancy using a constrained dichromatic reflection model. *International Journal of Computer Vision*, 42:127 – 144, 2001. DOI: 10.1023/A:1011120214885. URL http://www.springerlink.com/content/u7523r74x847537x.

J. H. Friedman, J. L. Bentley y R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software (TOMS)*, 3:209 – 226, 1977. ISSN 0098-3500.

K. Fukumizu, F. R. Bach y A. Gretton. Statistical consistency of kernel canonical correlation analysis. *The Journal of Machine Learning Research*, 2007. ISSN 1533-7928.

H. Gray. *Anatomy of the Human Body*. Lea & Febiger, 20 ed$^{\text{ón}}$., 1918. URL http://www.bartleby.com/107/.

A. Guttman. R-trees: A dynamic index structure for spatial searching. En *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data*, vol. 14, págs. 47 – 57, Boston, Massachusetts, 1984. ACM.

R. Gvili. *Depth keying*, vol. 5006. SPIE, 2003. DOI: 10.1117/12.474052. URL http://link.aip.org/link/?PSI/5006/564/1&Agg=doi.

R. M. Haralick y L. G. Shapiro. The consistent labeling problem: Part I. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1:173–184, 1979. ISSN 0162-8828. DOI: 10.1109/TPAMI.1979.4766903. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4766903.

R. M. Haralick, J. L. Mohammed y S. W. Zucker. Compatibilities and the fixed points of arithmetic relaxation processes. *Computer Graphics and Image Processing*, 13:242–256, 1980.

D. R. Hardoon y J. Shawe-Taylor. KCCA for different level precision in content-based image retrieval. En *Third International Workshop on Content-Based Multimedia Indexing, IRISA*, 2003. DOI: 10.1.1.13.6122. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.13.6122`.

D. R. Hardoon, J. Shawe-Taylor y O. Friman. KCCA for fMRI analysis. En *Proceedings of Medical Image Understanding and Analysis*, London, 2004. URL `http://eprints.ecs.soton.ac.uk/10658/`.

R. Hartley y A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, 2004.

T. Hofmann, B. Schölkopf y A. J. Smola. Kernel methods in machine learning. *The Annals of Statistics*, 36:1171–1220, 2008. ISSN 0090-5364. DOI: 10.1214/009053607000000677. URL `http://projecteuclid.org/euclid.aos/1211819561`.

G. Hongyu, A. Rangarajan, S. Joshi y L. Younes. A new joint clustering and diffeomorphism estimation algorithm for non-rigid shape matching. *Computer Vision and Pattern Recognition Workshop, 2004. CVPRW '04. Conference on*, 2004. DOI: 10.1109/CVPR.2004.9. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1384805`.

R. W. G. Hunt. *The Reproduction of Colour*. Wiley, 6 ed$^{\underline{\text{ón}}}$., 2004.

M. Isard y A. Blake. Condensation - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29:5–28, 1998. DOI: 10.1.1.36.8357. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.36.8357`.

T. Jaeggli, E. Koller-Meier y L. Gool. Learning generative models for multi-activity body pose estimation. *International Journal of Computer Vision*, 83: 121–134, 2008. ISSN 0920-5691. DOI: 10.1007/s11263-008-0158-0. URL http://www.springerlink.com/index/10.1007/s11263-008-0158-0.

I. Jolliffe. *Principal Component Analysis*. Springer-Verlag, 2 ed$^{\underline{ón}}$., 2002.

G. S. Kimeldorf y G. Wahba. Some results on Tchebycheffian spline functions. *Journal of Mathematical Analysis and Applications*, 33:82–95, 1971.

J. Kittler y J. Illingworth. Relaxation labelling algorithms-a review. *Image and Vision Computing*, 3, 1986. ISSN 0262-8856.

G. J. Klinker, S. A. Shafer y T. Kanade. The measurement of highlights in color images. *International Journal of Computer Vision*, 2:7–32, 1988. DOI: 10.1.1.3.7531. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.3.7531.

M. Kolsch y M. Turk. Robust hand detection. En *Proceedings of Sixth IEEE International Conference on Automatic Face and Gesture Recognition*, vol. 30, págs. 614–619. IEEE, 2004. ISBN 0-7695-2122-3. DOI: 10.1109/AFGR.2004.1301601. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1301601.

J. Kuch y T. Huang. Vision based hand modeling and tracking for virtual teleconferencing and telecollaboration. En *Proceedings of IEEE International Conference on Computer Vision*, págs. 666–671. IEEE Comput. Soc. Press, 1995. ISBN 0-8186-7042-8. DOI: 10.1109/ICCV.1995.466875. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=466875.

T. Kunii. Model-based analysis of hand posture. *IEEE Computer Graphics and Applications*, 15:77–86, 1995. ISSN 02721716. DOI: 10.1109/38.403831. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=403831.

J. T.-Y. Kwok e I. W.-H. Tsang. The pre-image problem in kernel methods. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 15:1517–25, 2004. ISSN 1045-9227. DOI: 10.1109/TNN.2004.837781. URL http://www.ncbi.nlm.nih.gov/pubmed/15565778.

Y. Li y J. Shawe-Taylor. Using KCCA for Japanese—english cross-language information retrieval and document classification. *Journal of Intelligent Information Systems*, 27, 2006. ISSN 0925-9902.

J. Lin, Y. Wu y T. Huang. Modeling the constraints of human hand motion. En *Proceedings Workshop on Human Motion*, págs. 121–126, Los Alamitos, CA, USA, 2000. IEEE Comput. Soc. ISBN 0-7695-0939-8. DOI: 10.1109/HUMO.2000.897381. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=897381.

B. D. Lucas y T. Kanade. An iterative image registration technique with an application to stereo vision (ijcai). En *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI '81)*, págs. 674–679, 1981.

B. Luo y E. Hancock. *Procrustes Alignment with the EM Algorithm*, págs. 837. Springer Berlin / Heidelberg, 1999. DOI: 10.1007/3-540-48375-6_74. URL http://www.springerlink.com/content/helrcxveupxeg6yt.

T. Melzer, M. Reiter y H. Bischof. *Nonlinear Feature Extraction Using Generalized Canonical Correlation Analysis*, vol. 2130 de *Lecture Notes in Computer Science*, págs. 353 – 360. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001. ISBN 978-3-540-42486-4. DOI: 10.1007/3-540-44668-0. URL http://www.springerlink.com/index/10.1007/3-540-44668-0.

T. Melzer, M. Reiter y H. Bischof. Appearance models based on kernel canonical correlation analysis. *Pattern Recognition*, 36:1961–1971, 2003. ISSN 00313203. DOI: 10.1016/S0031-3203(03)00058-X. URL http://linkinghub.elsevier.com/retrieve/pii/S003132030300058X.

S. Mika, B. Schölkopf, A. Smola, K.-R. Müller, M. Scholz y G. Rätsch. Kernel PCA and de-noising in feature spaces. En *Advances in Neural Information Processing Systems*, eds. M. S. Kearns, S. S. A. y D. A. Cohn. MIT Press, 1999. URL citeseer.ist.psu.edu/mika99kernel.html.

T. Mita, T. Kaneko, B. Stenger y O. Hori. Discriminative feature co-occurrence selection for object detection. *IEEE transactions on pattern analysis and machine intelligence*, 30:1257–69, 2008. ISSN 0162-8828. DOI: 10.1109/TPAMI.2007.70767. URL http://www.ncbi.nlm.nih.gov/pubmed/18550907.

S. Mitra y T. Acharya. Gesture recognition: A survey. *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, 37:311–324, 2007. ISSN 1094-6977. DOI: 10.1109/TSMCC.2007.893280. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4154947.

T. Ohtsuki y G. Healey. Using color and geometric models for extracting facial features. *Journal of Imaging Science and Technology*, 42:554–561, 1999.

D. P. O'Leary y S. Peleg. Analysis of relaxation processes: The two-node two-label case. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13:618–623, 1983.

M. Oren y S. K. Nayar. Generalization of the lambertian model and implications for machine vision. *International Journal of Computer Vision*, 14:227–251, 1995. ISSN 0920-5691. DOI: 10.1007/BF01679684. URL http://www.springerlink.com/index/10.1007/BF01679684.

M. Pelillo y M. Refice. Learning compatibility coefficients for relaxation labeling processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16, 1994. ISSN 0162-8828.

B. T. Phong. Illumination for computer generated pictures. *Communications of the ACM*, 18:311–317, 1975. ISSN 0001-0782. DOI: http://doi.acm.org/10.1145/360825.360839.

D. Ramanan, D. A. Forsyth y A. Zisserman. Tracking people by learning their appearance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29: 16, 2007. ISSN 0162-8828.

Y. Rathi, S. Dambreville y A. Tannenbaum. Statistical shape analysis using kernel PCA. En *Proceedings of SPIE*, págs. 60641B–60641B–8. SPIE, 2006. DOI: 10.1117/12.641417. URL http://link.aip.org/link/PSISDG/v6064/i1/p60641B/s1&Agg=doi.

A. Rosenfeld, R. A. Hummel y S. W. Zucker. Scene labeling by relaxation operations. *IEEE Transactions on Systems, Man, and Cybernetics*, 6:420–433, 1976. ISSN 0018-9472. DOI: 10.1109/TSMC.1976.4309519. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4309519.

S. T. Roweis y L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science (New York, N.Y.)*, 290:2323–6, 2000. ISSN 0036-8075. DOI: 10.1126/science.290.5500.2323. URL http://www.ncbi.nlm.nih.gov/pubmed/11125150.

H. Sahbi. Affine invariant shape description using the triangular kernel and its application to leaf recognition. En *Proceedings of the 4th International Workshop on Content-Based Multimedia Indexing*, Riga, 2005.

J. Schäfer y K. Strimmer. A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics. *Statistical applications in genetics and molecular biology*, 4:Article32, 2005. ISSN 1544-6115. DOI: 10.2202/1544-6115.1175. URL http://www.ncbi.nlm.nih.gov/pubmed/16646851.

B. Schölkopf. The kernel trick for distances. En *NIPS*, págs. 301–307, 2000. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.20.327.

B. Schölkopf y A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond (Adaptive Computation and Machine Learning)*. The MIT Press, 2001. URL http://www.amazon.com/

Learning-Kernels-Regularization-Optimization-Computation/dp/0262194759.

B. Schölkopf, A. Smola y K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Comp.*, 10:1299 – 1319, 1998.

S. A. Shafer. Optical phenomena in computer vision. *Journal of Canadian Society for Computational Studies of Intelligence*, 1984.

S. A. Shafer. Using color to separate reflection components. *Color Research & Application*, 10:210–218, 1985. ISSN 03612317. DOI: 10.1002/col.5080100409. URL http://doi.wiley.com/10.1002/col.5080100409.

L. S. Shapiro y J. M. Brady. Feature-based correspondence: an eigenvector approach. *Image and Vision Computing*, 10, 1992. ISSN 0262-8856.

J. Shi y C. Tomasi. Good features to track. En *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, págs. 593–600. IEEE Comput. Soc. Press, 1994. ISBN 0-8186-5825-8. DOI: 10.1109/CVPR.1994.323794. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=323794.

K. Shoemake. Animating rotation with quaternion curves. En *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, págs. 245–254, New York, NY, USA, 1985. ACM. ISBN 0-89791-166-0. DOI: http://doi.acm.org/10.1145/325334.325242.

L. Sigal, A. O. Balan y M. J. Black. HumanEva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. *International Journal of Computer Vision*, 2009. ISSN 0920-5691. DOI: 10.1007/s11263-009-0273-6. URL http://www.springerlink.com/index/10.1007/s11263-009-0273-6.

M. Störring. Physics-based modelling of human skin colour under mixed illuminants. *Robotics and Autonomous Systems*, 35:131–142, 2001. ISSN 09218890.

DOI: 10.1016/S0921-8890(01)00122-1. URL http://linkinghub.elsevier.com/retrieve/pii/S0921889001001221.

M. Störring, H. J. Andersen y E. Granum. Skin colour detection under changing lighting conditions. En *7th Symposium on Intelligent Robotics Systems*, págs. 187–195, 1999. DOI: 10.1.1.28.7702. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.28.7702.

M. Störring, H. J. Andersen y E. Granum. Estimation of the illuminant colour from human skin colour. En *Proceedings of the International Conference on Automatic Face and Gesture Recognition*, págs. 64–69, 2000. DOI: 10.1.1.21.5499. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.21.5499.

D. J. Sturman. Whole-hand input, 1992.

H. Suetani, Y. Iba y K. Aihara. Detecting generalized synchronization between chaotic signals: A kernel-based approach. *Journal of Physics A: Mathematical and General*, 39:10723–10742(20), 2006. DOI: 10.1088/0305-4470/39/34/009. URL http://uk.arxiv.org/abs/nlin/0507006v2.

X. Sun, L. Wang y J. Feng. Further results on the subspace distance. *Pattern Recognition*, 40:328–329, enero 2007. ISSN 00313203. DOI: 10.1016/j.patcog.2006.06.002. URL http://linkinghub.elsevier.com/retrieve/pii/S0031320306002731.

J. B. Tenenbaum, V. De Silva y J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science (New York, N.Y.)*, 290:2319–23, 2000. ISSN 0036-8075. DOI: 10.1126/science.290.5500.2319. URL http://www.ncbi.nlm.nih.gov/pubmed/11125149.

C. Tomasi y T. Kanade. Shape and motion from image streams under orthography: A factorization method. *International Journal of Computer Vision*, 9:137–154, 1992. ISSN 0920-5691. DOI: 10.1007/BF00129684. URL http://www.springerlink.com/index/10.1007/BF00129684.

K. E. Torrance y E. M. Sparrow. Theory for off-specular reflection from roughened surfaces. En *Physics-Based Vision: Principles And Practice: Radiometry*, eds. L. B. Wolff, S. A. Shafer y G. Healey, págs. 32–41. Jones and Bartlett Publishers, Inc., 1992.

M. A. Turk y A. P. Pentland. Face recognition using eigenfaces. En *Proceedings of Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, págs. 586–591, 1991. DOI: 10.1109/CVPR.1991.139758. URL `http://dx.doi.org/10.1109/CVPR.1991.139758`.

A. Vardy. Articulated human hand model with inter-joint dependency constraints, 1998. URL `http://www.scs.carleton.ca/$\sim$avardy/software/hand/hand_doc.pdf`.

P. Viola, M. J. Jones y D. Snow. Detecting pedestrians using patterns of motion and appearance. *International Journal of Computer Vision*, 63:153–161, 2005. ISSN 0920-5691. DOI: 10.1007/s11263-005-6644-8. URL `http://www.springerlink.com/index/10.1007/s11263-005-6644-8`.

H. Wang. Non-rigid motion behaviour learning: A spectral and graphical approach, 2007. URL `http://www.cs.york.ac.uk/ftpdir/reports/2008/YCST/12/YCST-2008-12.pdf`.

H. Wang y E. Hancock. Correspondence matching using kernel principal components analysis and label consistency constraints. *Pattern Recognition*, 39:1012–1025, junio 2006. ISSN 00313203. DOI: 10.1016/j.patcog.2005.05.013.

L. Wang, X. Wang y J. Feng. Subspace distance analysis with application to adaptive bayesian algorithm for face recognition. *Pattern Recognition*, 39: 456–464, 2006. ISSN 00313203. DOI: 10.1016/j.patcog.2005.08.015. URL `http://linkinghub.elsevier.com/retrieve/pii/S003132030500381X`.

A. D. Wilson y A. F. Bobick. Parametric hidden markov models for gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(9):

884–900, 1999. ISSN 0162-8828. DOI: http://doi.ieeecomputersociety.org/10.1109/34.790429.

L. Wolf y A. Shashua. Learning over sets using kernel principal angles. *Journal of Machine Learning Research*, 4:913–931, enero 2003a. ISSN 1532-4435. DOI: 10.1162/jmlr.2003.4.6.913. URL `http://www.crossref.org/jmlr_DOI.html`.

L. Wolf y A. Shashua. Kernel principal angles for classification machines with applications to image sequence interpretation. En *Proceedings of 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, págs. 635–640, 2003b. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1211413`.

Y. Wu y T. S. Huang. Human hand modeling, analysis and animation in the context of HCI. En *Proceedings of International Conference on Image Processing*, págs. 6–10, Kobe, Japan, 1999a. DOI: 10.1.1.42.4572. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.42.4572`.

Y. Wu y T. S. Huang. Capturing articulated human hand motion: A divide-and-conquer approach. En *Proc. 7th Int. Conf. on Computer Vision, volume I*, págs. 606–611, 1999b. DOI: 10.1.1.38.2855. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.38.2855`.

O. Yamaguchi, K. Fukui y K.-I. Maeda. Face recognition using temporal image sequence. En *Proceedings of Third IEEE International Conference on Automatic Face and Gesture Recognition*, vol. 16, págs. 318–323, Nara, 1998. IEEE Comput. Soc. ISBN 0-8186-8344-9. DOI: 10.1109/AFGR.1998.670968. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=670968`.

Y. Yamanishi, J.-P. Vert, A. Nakaya y M. Kanehisa. Extraction of correlated gene clusters from multiple genomic data by generalized kernel canonical correlation analysis. *Bioinformatics*, 19:323i–330, julio 2003. ISSN 1460-2059. DOI: 10.1093/bioinformatics/btg1045. URL `http://www.bioinformatics.oupjournals.org/cgi/doi/10.1093/bioinformatics/btg1045`.

S. Yan, D. Xu, B. Zhang, H.-J. Zhang, Q. Yang y S. Lin. Graph embedding and extensions: A general framework for dimensionality reduction. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29:40 – 51, 2007.

Y. Yasumuro, Q. Chen y K. Chihara. 3D modeling of human hand with motion constraints. En *Proceedings of the International Conference on Recent Advances in 3-D Digital Imaging and Modeling*, págs. 275–282. IEEE Computer Society, 1997.

P. N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. En *Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*, págs. 311 – 321, Austin, Texas, United States, 1993. Society for Industrial and Applied Mathematics.