UNIVERSIDAD DE CASTILLA-LA MANCHA
ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA

MÁSTER UNIVERSITARIO EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE MÁSTER

Clasificación automática de espacios utilizando información visual y de profundidad

Cristina Romero González

Septiembre, 2012

UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA

Departamento de Sistemas Informáticos

TRABAJO FIN DE MÁSTER

Clasificación automática de espacios utilizando información visual y de profundidad

Autor: Cristina Romero González

Directores: Ismael García Varea

Jesús Martínez Gómez

Septiembre, 2012

AGRADECIMIENTOS

En primer lugar, quiero agradecer a mis tutores en este proyecto, Ismael García Varea y Jesús Martínez Gómez, su inestimable ayuda y apoyo a lo largo de todo el proceso.

Gracias también a mi familia y amigos, por apoyarme cuando parecía que todo iba en contra y ayudarme en todo lo que han podido sin pedir nada a cambio.

RESUMEN

La clasificación consiste en asignar una clase o categoría a un elemento según sus características. La finalidad de dicha clasificación puede ser muy variada, desde la predicción de preferencias de usuarios hasta el diagnóstico médico, pasando por la visión artificial para robots, entre otros.

Durante este proyecto, se va a estudiar un caso concreto: la clasificación de espacios (habitaciones de un edificio de oficinas) utilizando información tanto visual como de profundidad. En este proceso hay varios pasos fundamentales, que pueden afectar en gran medida al resultado final.

La representación de la información es uno de estos pasos fundamentales. Es necesario seleccionar un descriptor que nos de datos robustos y fiables, tanto en el caso de información visual como de profundidad. Para ello, se va a realizar un breve repaso sobre los descriptores que se utilizan en la actualidad.

Una vez conseguida la información de la imagen es necesario detectar los elementos clave para poder clasificarla correctamente. Para ello, se estudiará como la distribución espacial de los puntos de la imagen puede ayudar a mejorar los datos de clasificación.

Los algoritmos de aprendizaje permiten crear sistemas que mejoran sus resultados con la experiencia. Existen diversos tipos de aprendizaje, pero el que más se ajusta a este caso es un aprendizaje supervisado, en el que se proporciona un conjunto de datos ya clasificados para que el algoritmo aprenda a reconocer nuevas instancias de los mismos.

Finalmente, es necesario realizar las pruebas pertinentes para poder determinar el correcto funcionamiento del sistema. En este caso, se va a comprobar en qué medida, la información de profundidad mejora los resultados de las características visuales cuando hay cambios de iluminación en al escena. Así como comprobar, como otros parámetros del proceso afectan a la solución final.

A lo largo de esta memoria se hará un breve repaso del estado del arte en clasificación automática. Se estudiarán las librerías y herramientas existentes que pueden ser de ayuda para el desarrollo de la misma y, por último, se llevarán a cabo la pruebas necesarias para determinar la eficacia de la propuesta que aquí se recoge.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

La clasificación consiste en asignar una clase o categoría a un elemento atendiendo a sus características. Es un campo con múltiples aplicaciones, desde el diagnóstico en medicina hasta el sistema de visión de un robot móvil.

Una de las aplicaciones de este proceso es la de clasificación de escenas visuales. La tarea consiste en, utilizando como entrada imágenes, asignarles una categoría que identifiquen qué representan. Existen en la actualidad diversas bases de datos con imágenes de distinta naturaleza para poder llevar a cabo esta tarea y comprobar sus resultados.

De este modo, se podría partir de una imagen con información de profundidad para intentar averiguar qué es. Los pasos a seguir para el tratamiento de dicha imagen serían los siguientes:

1. Captura de la imagen.
2. Identificación de las zonas de la imagen que tienen la información necesaria.
3. Representación de dicha información.
4. Usar la representación para determinar la clase de la imagen.

Para llevar a cabo este proceso son necesarios varios elementos adicionales. Por ejemplo, es necesario un dispositivo capaz de capturar información de profundidad. También se debe poder reconocer las zonas importantes o puntos de interés, y saber cómo utilizarlos para decidir ante qué imagen estamos.

El objetivo de este proyecto es precisamente, profundizar en dichos elementos y utilizarlos para la clasificación de imágenes de profundidad.

Así, se utilizarán descriptores que recogen la información relevante de los puntos clave de la imagen para representar los datos. Se estudiará su distribución geométrica y se utilizarán estos datos para enseñar a un algoritmo a reconocer las imágenes de ese tipo. Es decir, aquellas imágenes que representen la misma información y por tanto, los datos que se extraigan de ellas sean similares.

Finalmente, se debe comprobar que los resultados obtenidos son los esperados. Para ello se van a realizar diversas pruebas, que comprueben cómo afectan distintos factores involucrados en el proyecto en la clasificación final.

1.1. OBJETIVOS

Los principales objetivos de este proyecto se pueden resumir en los siguientes puntos:

- Realizar un estudio de todos los pasos del proceso necesarios para la clasificación de escenas visuales con información de profundidad. En concreto:

    a. Captura de imágenes.

    b. Extracción de información relevante de las mismas. En otras palabras, extracción de sus características.

    c. Procesamiento de dicha información haciendo uso de su información espacial.

    d. Algoritmos de aprendizaje. Es necesario disponer de un sistema que sea capaz de reconocer las partes relevantes de la información y recordarlas para identificar imágenes similares. En definitiva, que aprenda a clasificar.

- Obtener documentación sobre librerías y herramientas necesarias.

- Llevar a cabo la implementación que sea necesaria.

- Diseñar pruebas que permitan sacar conclusiones válidas.

1.2. ENFOQUE

La finalidad de este proyecto es el desarrollo de un sistema de clasificación con información de profundidad. Para ello se van a analizar y exponer distintos algoritmos y procesos que pueden ayudar con la tarea.

Primero, se hará un repaso del estado del arte en clasificación de imágenes. Se profundizará en los aspectos más relevantes como la extracción de características, el uso de información espacial de las mismas y los algoritmos de aprendizaje que se pueden utilizar.

Posteriormente se realizará una exposición más detallada de los objetivos de este proyecto.

A continuación, ser realizará un estudio de las herramientas y librerías disponibles. Haciendo uso de las mismas se procederá a la proposición e implementación de un sistema de clasificación con información visual y de profundidad.

Se realizarán las pruebas necesarias para comprobar la eficiencia de la clasificación, así como para comprobar qué factores son los que más afectan a los resultados.

Finalmente, se expondrán las conclusiones obtenidas y se sugerirán algunas mejoras que deberían ofrecer unos resultados más precisos y que pueden llevarse a cabo en futuros proyectos.

2. ESTADO DE LA CUESTIÓN

Es este capítulo se expondrá una descripción de los elementos clave implicados en la clasificación de imágenes de profundidad.

2.1. PROCESAMIENTO DE IMÁGENES

La visión por computador (o visión artificial) hace referencia a los procesos de obtención, caracterización e interpretación de la información contenida en imágenes, mediante el uso de un ordenador.

Una imagen es una función bidimensional de intensidad de luz $f(x, y)$, donde x e y representan las coordenadas espaciales y el valor de f en un punto cualquiera (x, y) es proporcional al brillo (o nivel de gris) de la imagen en ese punto. Una imagen digital es una imagen $f(x, y)$ que se ha discretizado tanto en las coordenadas espaciales como en el brillo. [1]. A su vez, las imágenes 3-dimensionales se representan mediante una función $f(x, y, z)$, donde z representa la coordenada adicional que indica el nivel de profundidad.

El procesamiento de imágenes es la disciplina del campo de la visión por computador encargada del tratamiento y análisis de las imágenes haciendo uso de computadoras.

Por lo general, para la clasificación de los distintos procesos asociados al tratamiento de imágenes se suelen considerar tres niveles.

- **Nivel bajo: Captación y preprocesado**. Incluye funciones para reducir ruido o aumentar el contraste, de modo que se facilite las posteriores etapas del procesamiento. En este caso, tanto las entradas como las salidas son imágenes.

- **Nivel medio: Segmentación, descripción y clasificación (reconocimiento)**. Se extraen características de las imágenes. Las entradas generalmente son imágenes mientras que las salidas suelen ser atributos de las imágenes.

- **Nivel alto: Reconocimiento e interpretación**. Utiliza los atributos obtenidos en el nivel medio para "dar sentido" a la imagen, en función de la aplicación final en la que se utilice.

A continuación, se van a exponer algunos métodos relacionados con el procesamiento de imágenes a cada uno de sus niveles. Desde la captura de imágenes tridimensionales, a la

extracción de características de las mismas y su posterior clasificación, todos ellos relevantes para la clasificación de imágenes.

2.1.1. Captura de imágenes

Una imagen se produce cuando diversas formas de radiación, las cuales han sido afectadas por objetos físicos, inciden y son registradas por un sensor. El sensor convierte la información visual en señales analógicas que son muestreadas espacialmente y cuantificadas en amplitud dando lugar a una señal digital.

La iluminación es uno de los factores más importantes en tratamiento de imágenes. Las técnicas más utilizadas son:

- *Iluminación difusa*. Incide en todas direcciones, de forma que al objeto le llegue la misma cantidad de luz en todos sus puntos. Se utiliza en aplicaciones que requieren analizar características de la superficie.

- *Iluminación a contraluz o retroiluminación*. Se emplea cuando la información a extraer reside en la silueta (contorno) del objeto.

- *Iluminación estructural*. Consiste en colocar un patrón delante de la fuente luminosa de forma que las características del patrón queden en el objeto. De este modo, se puede extraer información tridimensional.

- *Iluminación direccional*. Consiste en iluminar el objeto de forma rasante. Es útil para buscar desperfectos o irregularidades.

A diferencia de las imágenes visuales, las imágenes de profundidad captadas mediante sensores 3D contienen información de profundidad y pueden ser menos sensibles a condiciones de iluminación.

Para comprobar cómo las diferentes condiciones de iluminación afectan a la clasificación de imágenes visuales y de profundidad, se van a realizar pruebas con imágenes con distintos niveles de iluminación (ver apartado 5.2).

2.1.2. Extracción de características

Muchas de las aplicaciones de procesamiento de imágenes necesitan extraer la información básica de las mismas. Por ejemplo, en reconocimiento de objetos o registrado de imágenes, es indispensable poder encontrar información común.

Este proceso se suele realizar extrayendo una serie de *características* que describen, de la mejor forma posible, un segmento relevante de información de la imagen. De este modo, se pueden realizar comparaciones de forma eficiente.

La extracción de características suele centrarse en dos aspectos fundamentales:

1. Localizar los puntos que proporcionan la información necesaria. Estos puntos reciben el nombre de puntos clave (*key points*) o puntos de interés.

2. La información del entorno de vecindad de dichos puntos se codifica en un descriptor. En estos casos se habla de descriptores locales.

Algunos de los descriptores 2D más relevantes son:

- SIFT (*Scale-Invariant Feature Transform*) [2], un "método para extraer características invariantes distintivas de imágenes que pueden ser usadas para llevar a cabo un *matching* fiable entre distintas vistas de un objeto o escena". Estas características locales (o puntos) son invariantes a escalado y rotación, y parcialmente invariantes a cambios en iluminación y punto de vista (Figura 2.1).

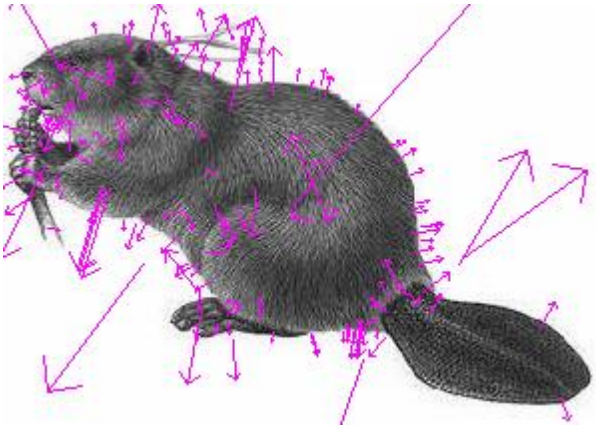


Figura 2.1: Características SIFT en una imagen

El algoritmo SIFT define cuatro etapas importantes para la generación del conjunto de puntos invariantes:

1. **Detección de extremos escala-espacio**. Inicialmente se realiza una búsqueda en todas las escalas y localizaciones de la imagen. Se puede implementar de forma eficiente utilizando una función diferencia de Gaussiana para identificar puntos potenciales que son invariantes a escala y orientación.

2. **Localización de puntos**. Para cada candidato se ajusta un modelo para determinar su localización y escala. Los puntos se seleccionan según medidas de su estabilidad [2].

3. **Asignación de la orientación**. A cada localización del punto se le asigna una o más orientaciones basadas en las direcciones de los gradientes locales. Las futuras operaciones se realizarán sobre datos que han sido transformados de forma relativa a la orientación, escala y localización asignadas. De este modo se proporciona invariancia ante estas transformaciones.

4. **Descriptor del punto**. Se genera el descriptor para cada punto dividiendo la región próxima al punto en una cuadrícula y computando un histograma de direcciones de gradientes locales en cada celda. El descriptor será el resultado de concatenar dichos histogramas.

- SURF (*Speed Up Robust Features*) [3], se trata de un descriptor basado en propiedades parecidas a las de SIFT, pero con menor nivel de complejidad, de modo que es capaz de detectar características en las imágenes en un tiempo más reducido que SIFT. Consta de dos pasos:

    1. Asignar una orientación basada en la información de una región circular alrededor del punto de interés.

    2. Seleccionar una región cuadrada alineada con la orientación y construir el descriptor a partir de ella.

- PHOG (*Pyramid of Histograms of Orientation Gradients*) [4], representa las imágenes mediante su forma local y la composición espacial de la forma. La forma local se define según la distribución de las orientaciones en los bordes de una región, mientras que la composición espacial se obtiene dividiendo la imagen en regiones a distintas resoluciones.

  El descriptor está formado por un histograma de gradientes de orientación de cada subregión a cada nivel de resolución (ver Figura 2.2).

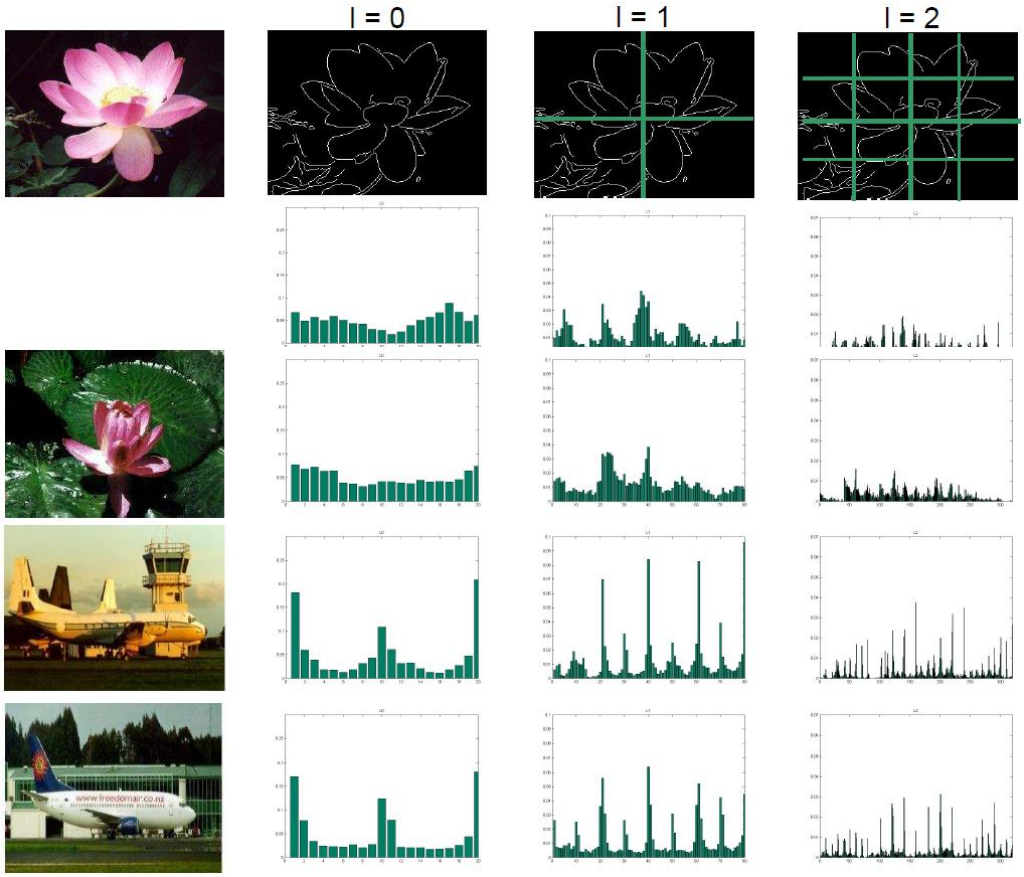


Figura 2.2: El descriptor PHOG será la concatenación ponderada de los histogramas de todos los niveles

Del mismo modo, se pueden definir descriptores 3D que utilizan la información de profundidad para extraer los datos relevantes de las imágenes. Algunos de ellos:

- *Spin-image* [5], utiliza una representación 2D de la superficie que rodea a un punto 3D.

- *Detección de objetos basada en el contorno* [6], se trata de una aproximación basada en siluetas extraídas de las imágenes.

- NARF (*Normal Aligned Radial Feature*) [7], considera de forma explícita los bordes de los objetos identificados por transiciones del fondo a primer plano. A continuación, se explicará con más detalle el funcionamiento de este descriptor.

2.1.3. Características NARF (*Normal Aligned Radial Feature*)

Dos de los métodos más populares para la extracción de puntos de interés y la creación de descriptores estables en el área de visión por computador 2D son SIFT y SURF. Estos métodos se basan en gradientes locales y una orientación única en una región

de la imagen para lograr la invariancia a la rotación. Aunque SIFT y SURF no se pueden trasladar directamente al espacio 3D, muchos de sus conceptos son útiles.

Las características NARF [7] proporcionan un método de extracción de puntos así como un descriptor de puntos 3D. Las características NARF se generan en puntos donde la superficie es estable pero cambia de forma significativa en el entorno de vecindad más cercano. Además, hace un uso explícito de la información de bordes.

Un aspecto importante de la extracción de características es el tratamiento de los bordes. Los bordes suelen aparecer como líneas no continuas de primer plano al fondo. Existen tres tipos de bordes relevantes para este algoritmo (ver Figura 2.3):

- *Bordes de objetos* (*obstacle border*): Puntos visibles más exteriores que pertenecen al objeto.

- *Bordes de sombras* (*shadow borders*): Puntos en el fondo contiguos a oclusiones.

- *Puntos velo* (*veil points*): Puntos interpolados entre los bordes de los objetos y los bordes de sombra. El correcto tratamiento de estos puntos mejora claramente los resultados posteriores de comparación y clasificación.

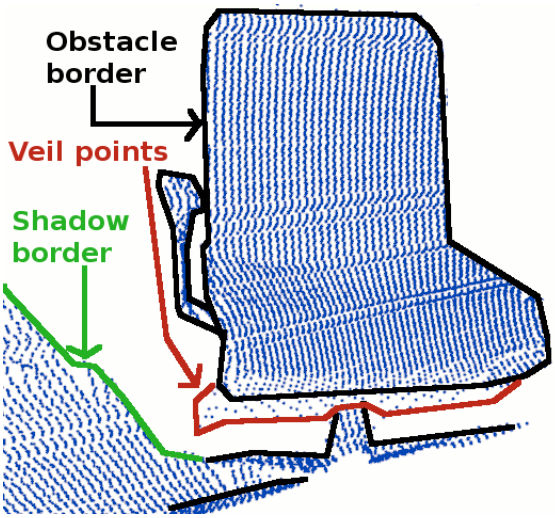


Figura 2.3: Tipos de puntos

Existen diversos indicadores para la detección de bordes. En este caso, se utiliza un cambio en la distancia entre puntos vecinos. Este indicador es además muy robusto ante ruido y cambios de resolución. Para clasificar los bordes se siguen estos pasos:

1. Utilizar una heurística para encontrar la distancia 3D euclídea entre puntos vecinos que no pertenecen a un borde.

2. Utilizar esta información para calcular una puntuación que determine si dicho punto forma parte de un borde.

3. Identificar la clase del punto del borde.

4. Utilizar una supresión no-máxima para encontrar la posición exacta del borde. Es decir, descartar aquellos puntos por debajo del máximo local y de un umbral preestablecido (0.8).

El siguiente paso es detectar los puntos de interés. Para ello se establecen una serie de requisitos:

- Debe utilizar información de los bordes y de la estructura de la superficie.

- Deben seleccionarse puntos que se puedan detectar incluso si el objetos se observa desde otra perspectiva.

- Los puntos deben encontrarse en áreas estables que permitan la estimación de la normal y el cálculo del descriptor, en general.

Los puntos de interés estables necesitan cambios de superficie considerables en una vecindad local para poder ser detectados de forma robusta independientemente de la perspectiva. Para esto, hay que:

1. Comprobar el entorno de vecindad de cada punto de la imagen y determinar una puntuación según cuánto cambia la superficie en dicha posición y la dirección dominante en ese cambio, también se debe incorporar la información de los bordes.

2. Consultar las direcciones dominantes alrededor de cada punto y calcular un valor de interés que represente:
    a. Cuánto difieren esas direcciones entre sí.
    b. Cuánto cambia la superficie del punto.

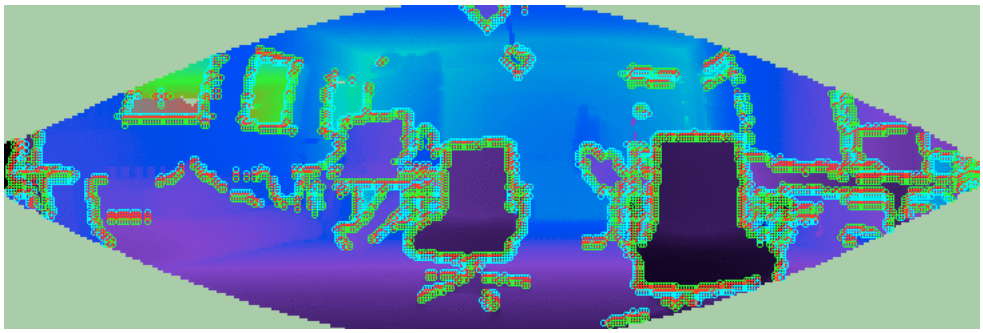
3. Ajustar los valores de interés.

4. Realizar una supresión no-máxima para encontrar los puntos de interés finales.

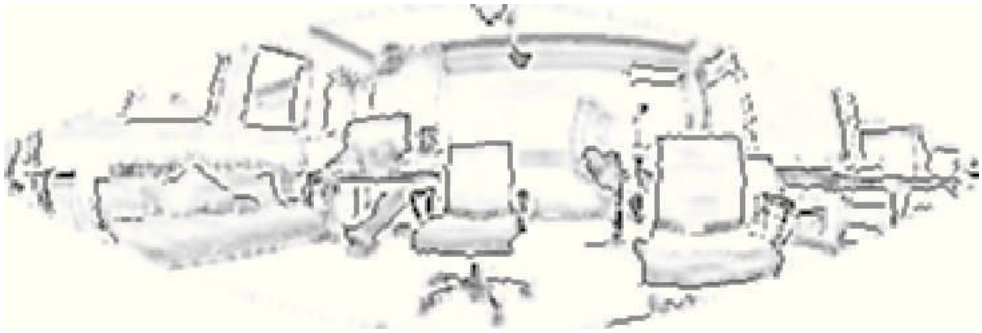
El parámetro más relevante en este proceso es el *tamaño de soporte* (*support size*) σ, que es el diámetro de la esfera alrededor del punto de interés que incluye a todos los puntos del entorno de vecindad cuyas direcciones dominantes se utilizan para el cálculo del valor de interés. Este mismo valor se utilizará posteriormente para determinar qué puntos se considerarán en el cálculo del descriptor.

En la Figura 2.4, se puede observar el procedimiento de extracción de puntos clave. En *a* podemos observar una imagen de profundidad con los bordes extraídos marcados, los distintos tipos de bordes aparecen marcados con diferentes colores. En *b* aparecen las puntuaciones de cambio de superficie según los bordes y el principio de curvatura. En *c* están señalados los valores de interés para un tamaño de soporte de 20cm, mientras que en *d* están señalados dichos valores para un tamaño de soporte de 1m.

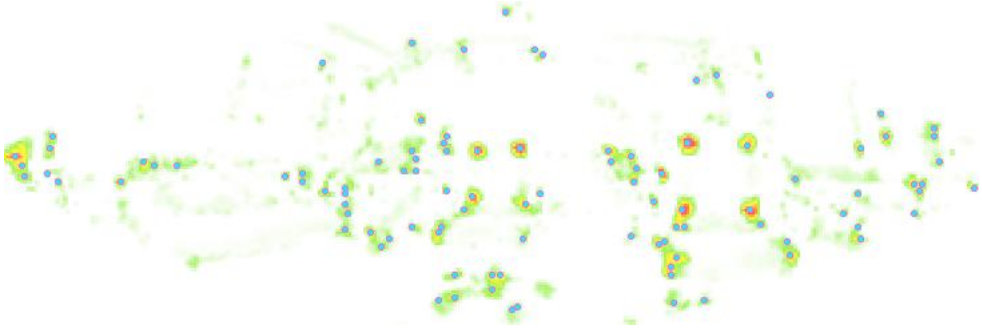
a.



b.



c.



d.

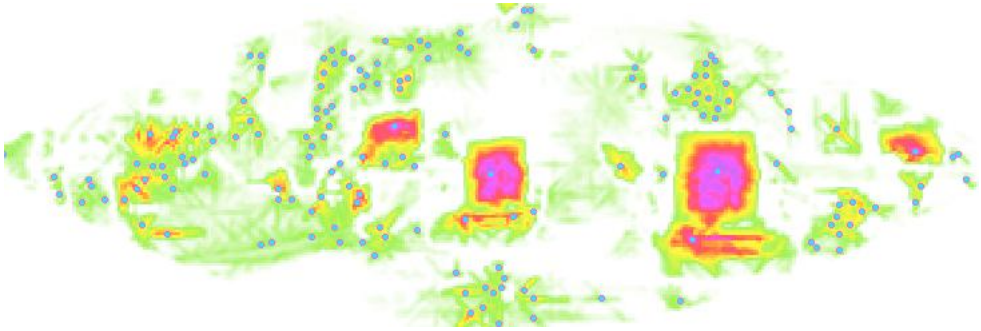


Figura 2.4: Procedimiento de extracción de puntos de interés

El último paso del proceso es el cálculo del descriptor. Los descriptores describen el área alrededor de un punto de interés de forma que su posterior comparación sea eficiente. Los descriptores NARF pretenden:

- Capturar la existencia de espacio libre y ocupado, de forma que se puedan describir partes de la superficie y la forma exterior de los objetos.

- Que sea robusto frente a ruido en el punto de interés.

- Que permita extraer un único espacio de coordenadas local.

De forma similar a SIFT y SURF, el descriptor NARF permite obtener una orientación única respecto a la normal.

En definitiva, para generar el descriptor NARF se siguen los siguientes pasos:

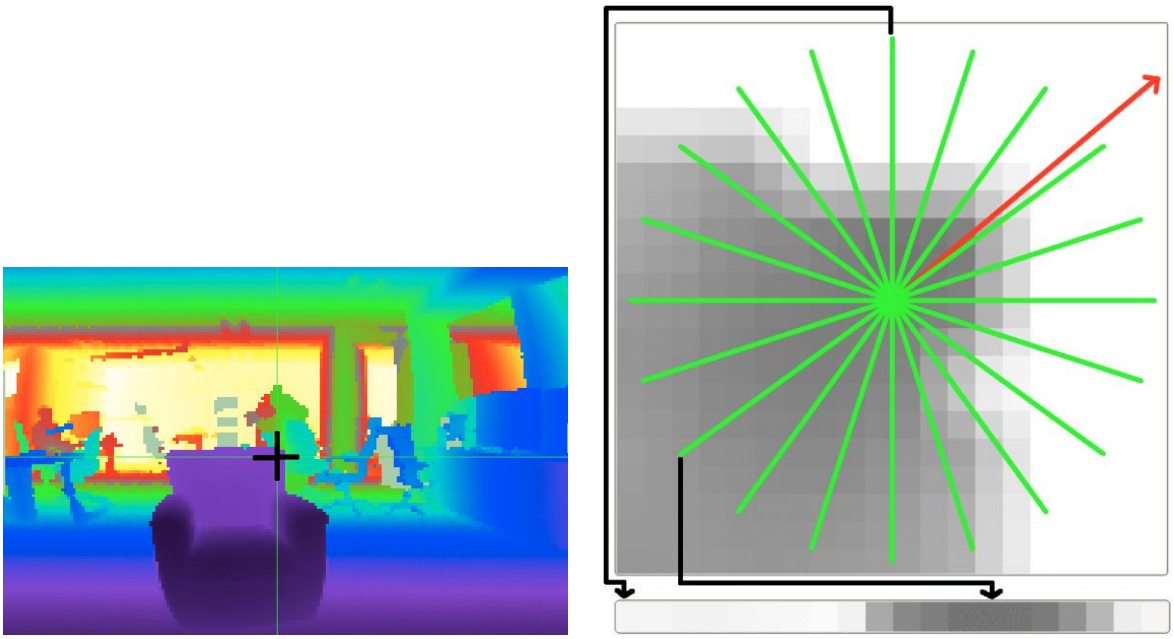
1. Calcular un rango alineado a la normal en el punto, es decir, una pequeña imagen con el observador mirando directamente al punto a través de la normal.

2. Superponer un patrón en estrella, de forma que cada rayo corresponda a un valor del descriptor final. De esa forma se captura cómo cambian los píxeles bajo el rayo.

3. Extraer una orientación única.

4. Desplazar el descriptor según su valor para que sea invariante a rotación.

En la Figura 2.5 aparecen los pasos de generación del descriptor NARF para un punto. En *a* se puede observar la imagen de profundidad de ejemplo con un sillón al frente, la cruz negra señala el punto de interés. En *b* se puede visualizar cómo el descriptor es calculado. Se muestra una región de la esquina del sillón y debajo el descriptor, cada una de las celdas corresponde a cada uno de los rayos (verde) y la flecha roja señala la orientación dominante extraída.



a. b.

Figura 2.5: Generación del descriptor NARF

2.2. SISTEMAS DE CLASIFICACIÓN

La *clasificación* consiste en asignar una categoría o clase a un conjunto de elementos, dependiendo de sus atributos o características [8]. Un algoritmo que implementa clasificación recibe el nombre de *clasificador*.

Existen multitud de ejemplos de problemas de clasificación: diagnóstico preliminar de pacientes en medicina, distribución del correo según el código postal, asignación de crédito en función de la información financiera, etc.

Los clasificadores pueden ser binarios o multi-clase, dependiendo del número de salidas esperadas. Hablamos de clasificación multi-clase siempre que tengamos más de dos clases.

Aunque algunos clasificadores tienen naturaleza binaria, se pueden utilizar para una clasificación multi-clase mediante diversas estrategias. Por ejemplo, se puede entrenar un clasificador binario por cada clase, de modo que distinga esa clase de las demás. Posteriormente se elegirá aquella clase que proporcione un mayor nivel de confianza. Esta estrategia se conoce como *todos contra uno*.

2.2.1. Tipos de aprendizaje

Tom Mitchell definió formalmente el aprendizaje del siguiente modo: "Se dice que una aplicación aprende de la experiencia E con respecto a una clase de tarea T y una medida de eficiencia P, si su eficiencia en tareas en T, medidas por P, mejora con la experiencia E" [9].

Es decir, el aprendizaje consiste en aumentar el conocimiento respecto a un problema concreto a base de experiencia u observaciones. Estas observaciones suelen un conjunto de datos no estructurados, a partir de los cuales se debe encontrar un patrón o comportamiento.

Una vez entrenado el algoritmo de aprendizaje, lo bien que prediga la salida correcta para nuevos casos se llama generalización [10]. Para obtener la mejor generalización debemos comparar la complejidad de la hipótesis con la función que representa los datos. Si la hipótesis es menos compleja que la función, estamos ante un caso de subajuste o *underfitting*. En el otro extremo, si la hipótesis es más compleja que la función de los datos, hablamos de sobreajuste u *overfitting* (ver Figura 2.6). Para medir la generalización de una hipótesis se puede utilizar un subconjunto del entrenamiento como conjunto de validación para probar el resultado.

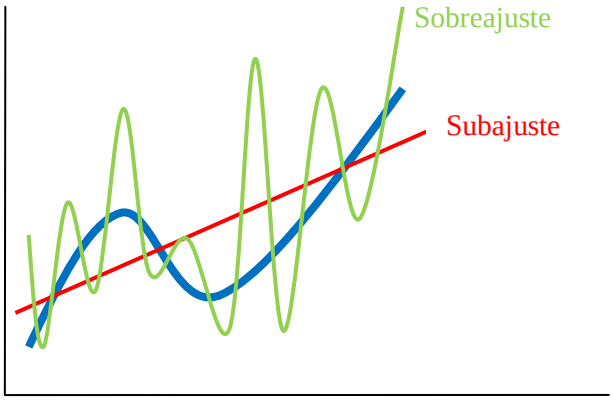


Figura 2.6: Generalización del aprendizaje

El tipo de realimentación disponible para el aprendizaje suele ser el factor más importante a la hora de determinar la naturaleza del problema. Atendiendo a este factor, se distinguen tres tipos de aprendizaje [11]:

- **Aprendizaje supervisado**. Consiste en aprender una función a partir de ejemplos de sus entradas y salidas. Partimos de un número conocido de clases y el objetivo

es establecer una regla que permita clasificar nuevas observaciones en una de las clases existentes.

- **Aprendizaje no supervisado**. Aprender a partir de patrones de entradas para los que no se especifican los valores. En clasificación, partiríamos de un conjunto de observaciones con la intención de establecer clases o grupos de datos.

- **Aprendizaje por refuerzo**. Consiste en usar recompensas observadas para aprender un comportamiento óptimo.

La clasificación suele considerarse un problema de aprendizaje supervisado. El algoritmo no-supervisado correspondiente recibe el nombre de *clustering* y consiste agrupar elementos similares en un mismo grupo.

2.2.2. Algoritmos de aprendizaje

Existe una gran variedad de algoritmos de aprendizaje que se pueden aplicar al problema de clasificación. A continuación explicaremos las características de algunos de ellos.

- **Redes neuronales (ANN, *Artificial Neural Networks*)**. Las redes neuronales proporcionan un método práctico y general para aprender funciones con valores reales, discretos o vectores. Este aprendizaje es robusto a errores y se ha utilizado con éxito en problemas como la interpretación de imágenes visuales, reconocimiento del habla, etc. [9]

  Este algoritmo es adecuado para problemas con las siguientes características:

  - Las instancias están representadas por muchos pares atributo-valor.

  - La función objetivo puede tener valores discretos, reales o vectores de valores reales o discretos.

  - Los ejemplos de entrenamiento pueden contener errores.

  - Es aceptable un tiempo de entrenamiento largo.

  - Puede ser necesaria una evaluación rápida de la función objetivo aprendida.

  - La interpretabilidad de la función de evaluación no es importante.

Las redes neuronales están compuestas de nodos o unidades conectadas a través de conexiones dirigidas [11]. Una conexión de la unidad j a la unidad i sirve para propagar la activación de a_j de j a i. Además cada conexión tiene un peso numérico $W_{j,i}$ asociado, que determina la fuerza y el signo de la conexión. Cada unidad i primero calcula la suma ponderada de sus entradas:

$$in_i = \sum_{j=0}^{n} W_{j,i} a_j \tag{2.1}$$

Luego aplica una función de activación g a esa suma para producir la salida:

$$a_i = g(in_i) = g\left(\sum_{j=0}^{n} W_{j,i} a_j\right) \tag{2.2}$$

Las redes neuronales normalmente se organizan en capas, de forma que cada unidad recibe entradas únicamente de las unidades de la capa que la precede inmediatamente. La primera capa la forman las unidades de entrada y la última las unidades de salida, el resto son capas ocultas. (Ver Figura 2.7)

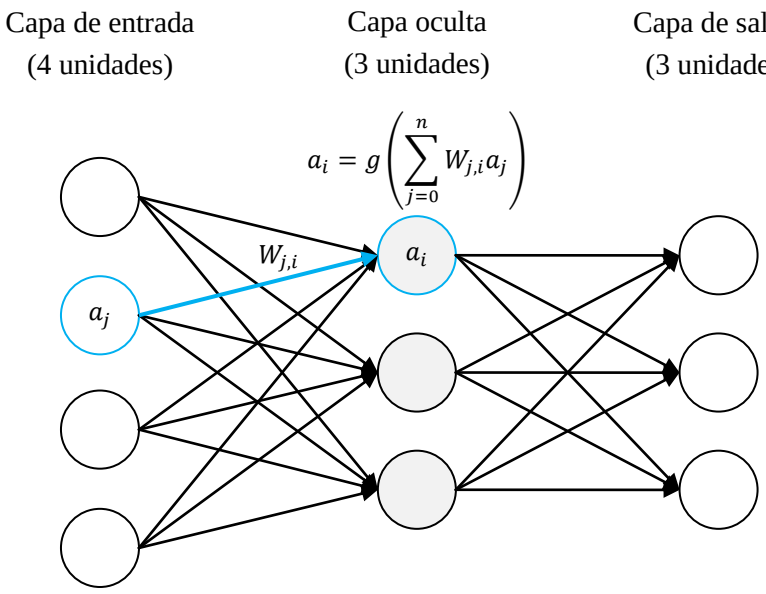


Figura 2.7: Ejemplo de red neuronal

Las redes de una única capa tienen un algoritmo de aprendizaje más simple y eficiente, pero tienen un poder de expresividad muy limitado. Por otro lado, las redes multicapa son más expresivas, pero son muy difíciles de entrenar debido a la abundancia de mínimos locales y la gran dimensión del espacio de pesos.

- **SVM (*Support Vector Machines*)** [12]. Las SVMs son uno de los algoritmos más populares en clasificación. Además, a diferencia de las redes neuronales, tienen una fuerte base teórica y se han utilizado con éxito en varias aplicaciones. Estos métodos usan un algoritmo de aprendizaje eficiente y pueden representar funciones no lineales complejas.

  Si se parte de un conjunto de datos que no es linealmente separable, se puede hacer corresponder los datos a un espacio con una dimensión suficientemente alta, de modo que sean siempre linealmente separables. En general, si tenemos N elementos en los datos, excepto en casos especiales, serán siempre separables en un espacio de dimensión $N + 1$ o mayor [11].

  Las SVMs mapean los vectores de entrada a un espacio de mayor dimensión, donde el problema consiste en encontrar un hiperplano que separe los datos de entrenamiento. El hiperplano óptimo que divide clases separables es aquél que proporciona un margen mayor entre ellas. Los vectores soporte quedan definidos por los puntos más cercanos al separador (ver Figura 2.8).

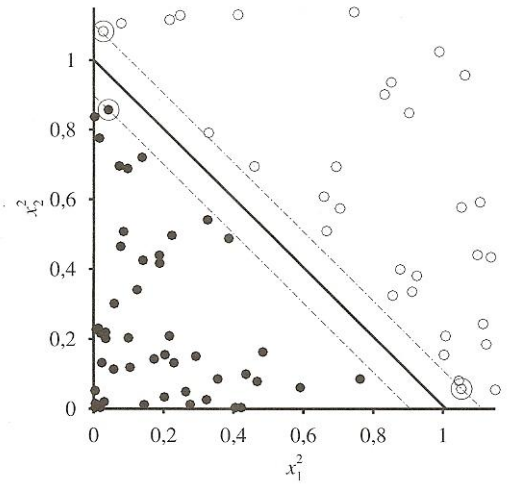


Figura 2.8: Separador en una SVM. Los puntos más cercanos (vectores soporte) están marcados con círculos

2.2.3. OISVM (*On-line Independent Support Vector Machines*)

Las SVMs son uno de los algoritmos de clasificación con más éxito. Pero debido a sus requisitos de espacio y rendimiento, no son aconsejables para aprendizaje on-line, es decir, cuando hay un flujo infinito de observaciones de entrenamiento y debe actualizarse con conocimiento con cada nueva muestra. En este caso, el tamaño de la solución SMVs crece de forma lineal con el número de muestras de entrenamiento.

El algoritmo *On-line Independent Support Vector Machines* (OISVMs) [13] converge de forma aproximada a la solución estándar obtenida con SVMs, cada vez que se añade una nueva observación.

Este método utiliza un conjunto de observaciones linealmente independientes e intenta proyectar cada nueva observación en el conjunto acumulado hasta entonces. Esto reduce considerablemente el tiempo y el espacio necesarios, a costa de una pérdida asumible en precisión. El tamaño de la solución obtenida mediante OISVM es siempre fijo, lo que implica un tiempo de pruebas también fijo.

De forma similar a otros algoritmos online discriminativos basados en kernel, Las OISVMs construyen la hipótesis mediante un subconjunto de muestras llamadas *base*. Las nuevas muestras se añaden a la base sólo si son linealmente independientes en el espacio de características de las muestras base actuales.

Las OISVMs producen modelos más pequeños si los comparamos con las SVMs estándar. Además, logran un rendimiento casi óptimo a la vez que mantiene el poder de generalización de las SVMs.

En concreto, el algoritmo OISVM propone, en lugar de entrenar y posteriormente simplificar la solución como en las SVMs, construir la solución directamente con un número pequeño de vectores base. El proceso puede resumirse como:

1. Comprobar si la muestra actual es linealmente independiente de las muestras base en el espacio de características. Si lo es, se añade a la base.

   Para determinar si una muestra es linealmente independiente se establece un factor de tolerancia η, cuanto mayor sea este valor la precisión disminuirá pero también se obtendrán bases más pequeñas. Si η es cero, la solución obtenida será la misma que con las SVMs clásicas.

2. Optimizar de forma incremental.

Este algoritmo está implementado en Matlab en la librería DOGMA [14].

2.3. PIRÁMIDE ESPACIAL

A continuación, se va a describir un método para reconocer la categoría semántica o clase de una imagen basado en correspondencias geométricas globales. Esta técnica divide la imagen en subregiones cada vez más precisas y computa histogramas de características locales dentro de cada subregión. La *pirámide espacial* resultante es una extensión sencilla

y computacionalmente eficiente de un conjunto de características desordenadas que representan una imagen [15].

A pesar de la simplicidad de este método, y a pesar de que no construye modelos de objetos explícitos sino que usa información global como prueba indirecta de la presencia de un objeto, es capaz de conseguir mejorar los resultados de clasificación respecto a una representación desordenada de la imagen. En definitiva, puede mejorar las prestaciones de métodos más sofisticados basados en partes y relaciones.

2.3.1. Kernels de comparación de pirámides (*Pyramid Match Kernels*)

Sean X e Y dos conjuntos de vectores en un espacio de características d-dimensional. Grauman and Darrel [16] proponen una *comparación de pirámides* para encontrar una correspondencia adecuada entre estos dos conjuntos.

La comparación de pirámides funciona estableciendo una serie de regiones en forma de cuadrícula sobre el espacio de características y seleccionando el número de coincidencias de características a cada nivel de resolución (Figura 2.9), ponderado por la precisión de dicho nivel. A una resolución determinada, se dice que dos puntos coinciden si están en la misma celda de la cuadrícula. Las características encontradas a resoluciones más precisas se ponderan más que aquellas encontradas en resoluciones más generales.

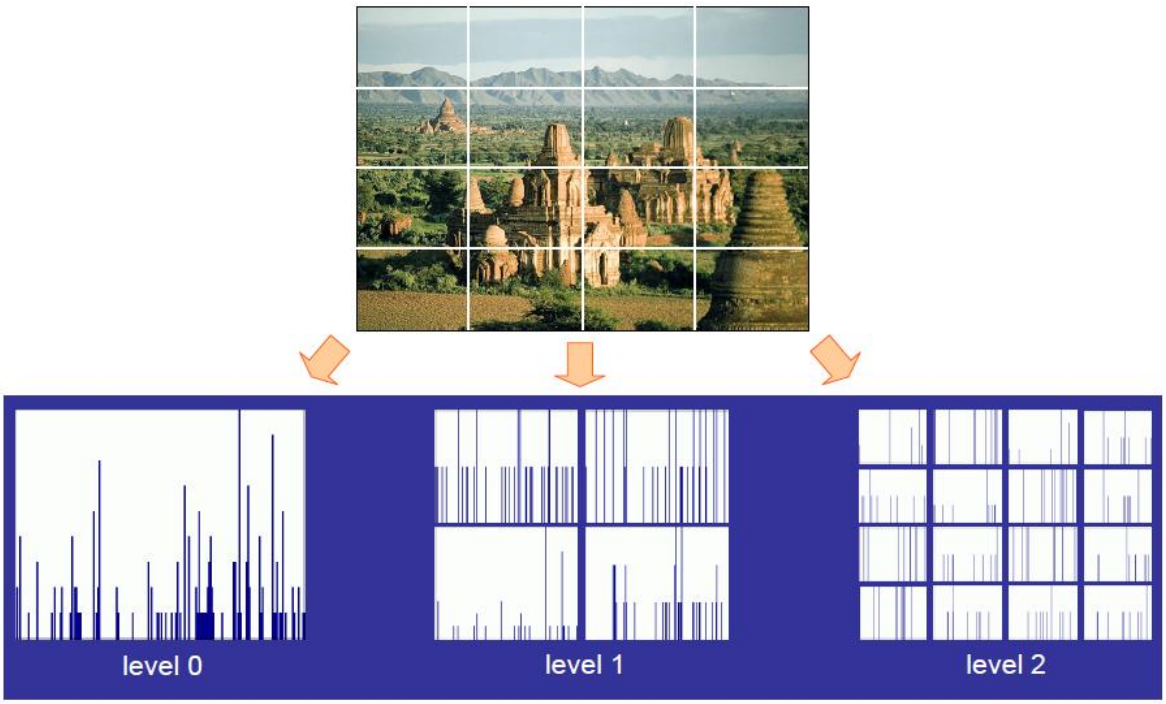


Figura 2.9: Representación de una pirámide espacial

De forma más específica, se construye una secuencia de L niveles de cuadrículas, de forma que el nivel l tenga 2^l celdas por dimensión y un total de $D = 2^{dl}$ celdas. H_X^l y H_Y^l representan los histogramas de X e Y a dicho nivel l, siendo $H_X^l(i)$ y $H_Y^l(i)$ el número de puntos de X e Y en la i-ésima celda de la cuadrícula. De este modo, el número de coincidencias al nivel l viene dado por la función de intersección de histogramas:

$$I^l = I\left(H_X^l, H_Y^l\right) = \sum_{i=1}^{D} \min(H_X^l(i), H_Y^l(i)) \tag{2.3}$$

El número de coincidencias encontrado en el nivel l incluye las del nivel $l+1$. Por eso, el número de nuevas coincidencias en el nivel l es $I^l - I^{l+1}$, para $l = 0, \dots, L-1$. Además, el peso asociado a cada nivel es $\frac{1}{2^{L-l}}$, que es inversamente proporcional al ancho de la celda. Al establecer este peso se penaliza aquellos puntos encontrados en celdas de mayor tamaño, como se puede observar en la Figura 2.10.

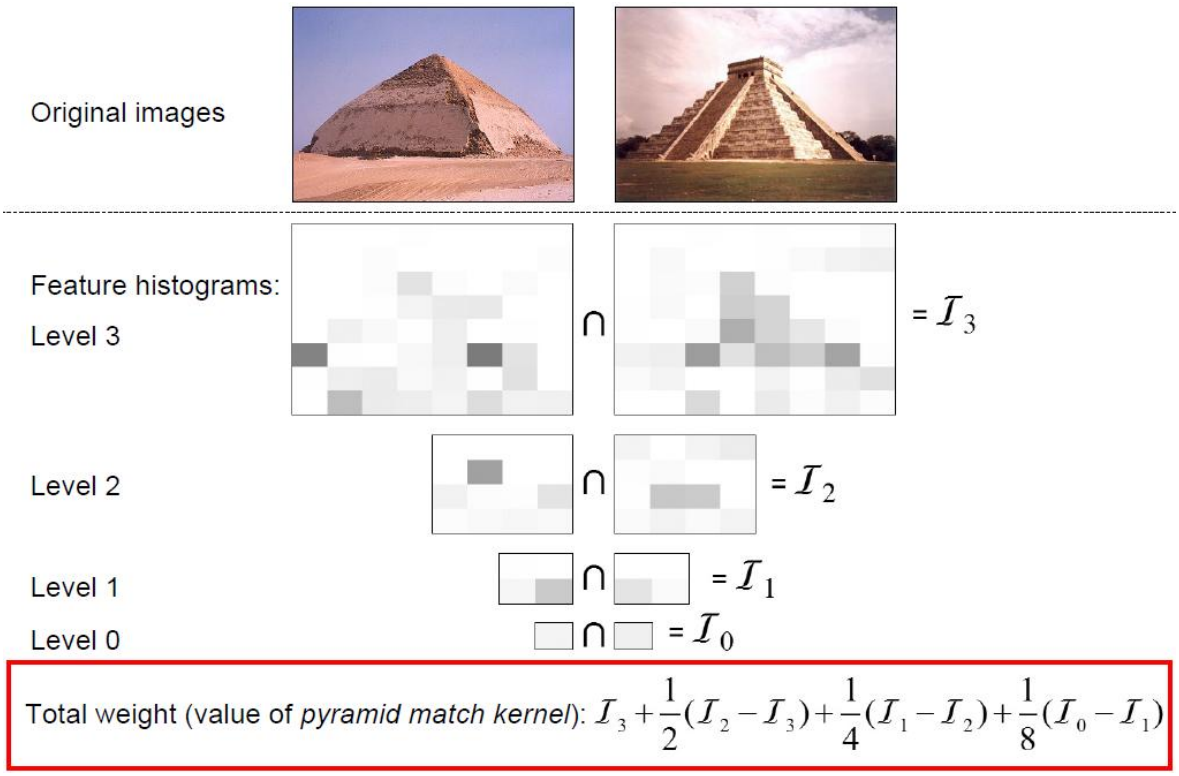


Figura 2.10: Ponderación de niveles

Combinando todo lo anterior obtenemos el *kernel de comparación de pirámides*:

$$\kappa^L(X,Y) = I^L + \sum_{l=0}^{L-1} \frac{1}{2^{L-l}}\left(I^l - I^{l+1}\right) = \frac{1}{2^L} I^0 + \sum_{l=0}^{L-1} \frac{1}{2^{L-l+1}} I^l \tag{2.4}$$

Tanto la intersección de histogramas como el kernel de comparación de pirámides son kernels Mercer [16].

2.3.2. Esquema de coincidencias espaciales

La propuesta de pirámide espacial realiza una comparación de pirámides en el espacio de imágenes de 2-dimensiones y utiliza técnicas tradicionales de clustering en el espacio de características [15]. En concreto, se agrupan todos los vectores de características en M tipos discretos y se asume que sólo las características del mismo tipo pueden coincidir entre sí.

Cada canal m proporciona dos conjuntos de vectores de 2-dimensiones (X_m y Y_m) que representan las coordenadas de las características del tipo m encontradas. El kernel final es la suma de los kernels de cada canal:

$$\mathrm{K}^L(X,Y) = \sum_{m=1}^{M} \kappa^L(X_m, Y_m) \tag{2.5}$$

Finalmente, se implementa K^L como un único histograma formado concatenando los histogramas ponderados de todos los canales en todas las resoluciones (Figura 2.11). Para L niveles y M canales, el vector resultante tiene un tamaño de $M\sum_{l=0}^{L} 4^l = M\frac{1}{3}(4^{L+1}-1)$. El último paso es la normalización de los histogramas.

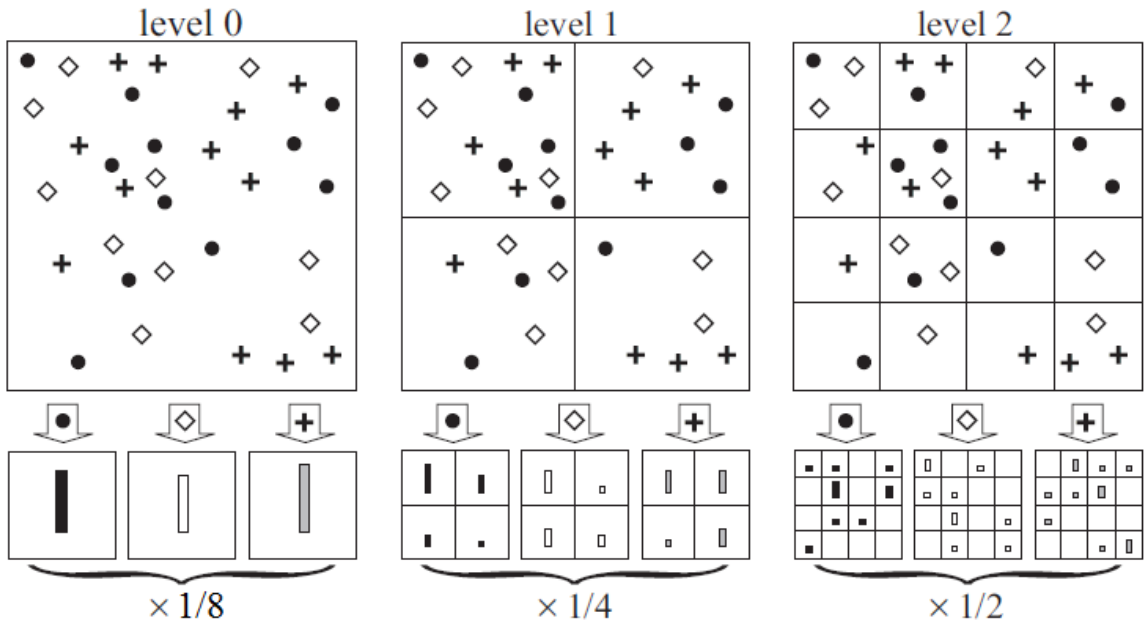


Figura 2.11: Construcción de pirámide de 3 niveles con 3 tipos de características (círculos, rombos y cruces)

En resumen, para cada nivel de la pirámide se construye una serie de histogramas, tantos como subdivisiones haya en dicho nivel. Cada uno de estos histogramas contendrá el número de apariciones de las palabras visuales del diccionario en la correspondiente región espacial. Finalmente, se construye la pirámide ponderando los histogramas según su nivel y concatenándolos.

3. OBJETIVOS DEL PROYECTO

Clasificar es asignar una categoría a un elemento de acuerdo a sus características. Para ello, es necesario un sistema capaz de reconocer qué datos diferencian a unos elementos de otros y a qué categoría o clase corresponden.

La pirámide espacial es un algoritmo sencillo y efectivo, que permite procesar las características extraídas de una serie de imágenes para, posteriormente, obtener unos mejores resultados de clasificación.

Sin embargo, estos resultados se basan en información visual, que es muy susceptible a cambios en el entorno como, por ejemplo, cambios de iluminación. Este problema no afecta tanto a las imágenes de profundidad. Es por este motivo que, en este trabajo, se pretende desarrollar un método similar para ese tipo de imágenes.

Partiendo de la información que se dispone de la pirámide espacial se va a incorporar una tercera dimensión. Así las imágenes quedarán divididas en cubos de igual tamaño, en lugar de en una cuadrícula. Para ello también es necesario disponer de un descriptor de características capaz de manejar información de profundidad. En este caso, vamos a utilizar características NARF.

Puesto que se va a utilizar imágenes de profundidad, no va a ser posible comparar los resultados directamente, con los obtenidos con el algoritmo original de pirámide espacial, pues las imágenes de prueba en ese caso eran 2D.

Una vez generada la pirámide es necesario un sistema capaz de reconocer las clases de las imágenes a partir de los histogramas que ésta contiene. En este caso, se va a utilizar un sistema de aprendizaje supervisado, en concreto un algoritmo OISVM.

Estamos ante un problema de clasificación multi-clase y la implementación del algoritmo que vamos a utilizar es binaria, por ello, vamos a utilizar una estrategia de todos-contra-uno para llevar a cabo la clasificación. En concreto, se entrenará el algoritmo de forma independiente para cada una de las clases y después se asignará aquella que devuelva un valor mayor.

Para comprobar la eficiencia del algoritmo se van desarrollar dos tipos de pruebas:

1. *Estudio de resultados utilizando distintos parámetros y condiciones de iluminación*.

Utilizando imágenes con unas condiciones de iluminación concretas, se va a generar un clasificador y los resultados del mismo se van a comparar con:

a. Distintos niveles de la pirámide.
b. Distintos tamaños de diccionario.
c. Distintas condiciones de iluminación.

De este modo, se podrá comparar como, cada uno de estos factores, afecta a la efectividad de la clasificación.

2. *Comprobar la efectividad del proceso*.

En este caso, se va entrenar el algoritmo OISVM ante los distintos tipos de iluminación, para comprobar los resultados que se obtendrían en una situación más real en la que pueden encontrarse situaciones de todo tipo.

Finalmente, considerando los resultados obtenidos en cada una de estas pruebas, se propondrán una serie de mejoras orientadas a obtener una mejor representación de la información proporcionada por las imágenes de profundidad y en definitiva, mejores resultados en la clasificación.

En concreto, los objetivos de este proyecto son:

- Definir los cambios necesarios en el algoritmo de pirámide espacial para poder gestionar información de imágenes tridimensionales. Esto implica:
    - Utilizar características que representen de forma adecuada la información de profundidad.
    - Realizar la división espacial en las tres dimensiones.
    - Generar la pirámide final de acuerdo a las nuevas dimensiones y, en consecuencia, a los nuevos requisitos de tamaño de la misma.
- Analizar y comprender las librerías auxiliares que servirán de apoyo a la implementación del proyecto.
- Analizar los resultados para determinar el interés de la mejora realizada en el algoritmo de *pirámide espacial*, comparando los resultados obtenidos utilizando únicamente información visual con los obtenidos mediante información de profundidad y con los obtenidos combinando ambos tipos de datos.

- Comprobar de qué forma afecta la iluminación a la clasificación de imágenes visuales y de imágenes de profundidad.

- Estudiar si el uso de un diccionario mayor y/o un mayor número de niveles en la pirámide ayuda a mejorar los resultados.

4. HIPÓTESIS DE TRABAJO

En este capítulo se va a realizar un estudio de las librerías y herramientas que existen en la actualidad y que van a ser de ayuda en la implementación la propuesta recogida en esta memoria.

4.1. LIBRERÍAS Y HERRAMIENTAS

El objetivo de este proyecto es clasificar imágenes haciendo uso de su información visual y de profundidad. Para ello, se utilizan diversas herramientas en todos y cada uno de los elementos que componen el proceso de tratamiento de imágenes. A continuación, se detallan las características básicas de dichos elementos.

4.1.1. Kinect y PCL (*Point Cloud Library*)

El primer paso para poder clasificar una imagen es la captura de la misma. Para ello en necesario un dispositivo que pueda detectar la información de profundidad necesaria, así como un software que gestione los datos recibidos.

El sensor Kinect de Microsoft es un controlador de juego desarrollado por Microsoft para la videoconsola Xbox 360. También es posible conectar el dispositivo al ordenador para realizar capturas con el mismo.

Kinect permite al usuario interactuar con la aplicación sólo con sus movimientos, sin ayuda de ningún controlador, mediante una interfaz que reconoce gestos, comandos de voz y objetos e imágenes. Es capaz de rastrear hasta seis personas e identificar los movimientos de dos de ellas. Además, identifica 20 articulaciones si el usuario está de pie y 10 si está sentado.

El sensor de Kinect incluye los componentes que se pueden ver en la Figura 4.1 [17]:

1. **Sensores de profundidad 3D**: Los sensores tridimensionales realizan un seguimiento del cuerpo en el área de juego.

2. **Cámara RGB**: La cámara RGB (rojo, verde, azul) permite identificarle y sacar fotos y vídeos en el juego.

3. **Micrófono multi matriz**: En la parte inferior delantera del sensor Kinect hay una serie de micrófonos que se usan para reconocimiento de órdenes y charla.

4. **Inclinación motorizada**: Una unidad mecánica en la base del sensor de Kinect lo inclina automáticamente hacia arriba o hacia abajo según sea necesario.

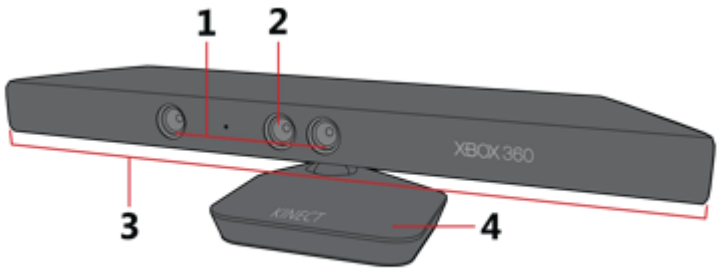


Figura 4.1: Componentes de Kinect

Las especificaciones técnicas que afectan a la captura de imágenes son [18]:

- Ángulos de visión (en profundidad y RGB). Ver Figura 4.2.
    - Campo de visión horizontal: 57.5 grados.
    - Campo de visión vertical: 43.5 grados con ±27 grados de inclinación hacia arriba y hacia abajo.

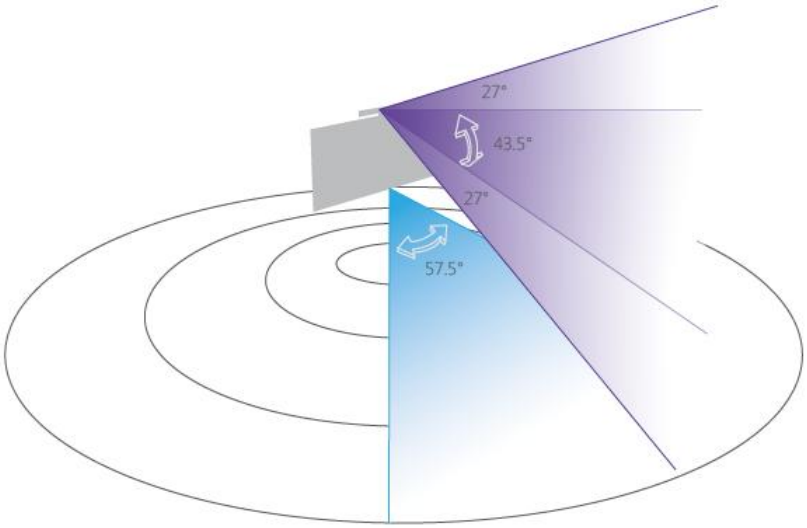


Figura 4.2: Campo de visión de Kinect

- Rango de distancias en profundidad (modo por defecto). Ver Figura 4.3.
    - Límites físicos: de 0.8 a 4m.
    - Límites reales: de 1.2 a 3.5m.

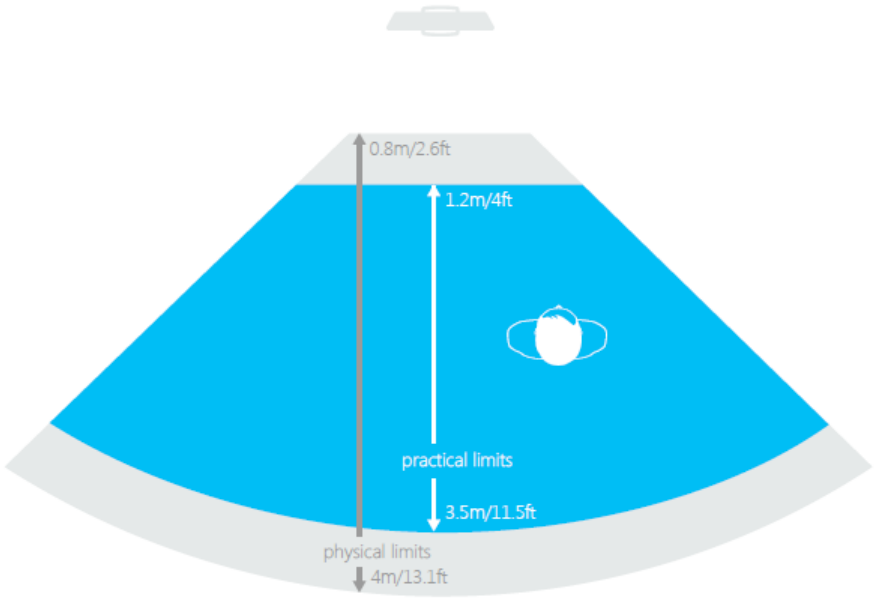


Figura 4.3: Rango de distancias en profundidad (modo por defecto)

- Rango de distancias en profundidad (modo cercano). Ver Figura 4.4.
  - Límites físicos: de 0.4 a 3m.
  - Límites reales: de 0.8 a 2.5m.

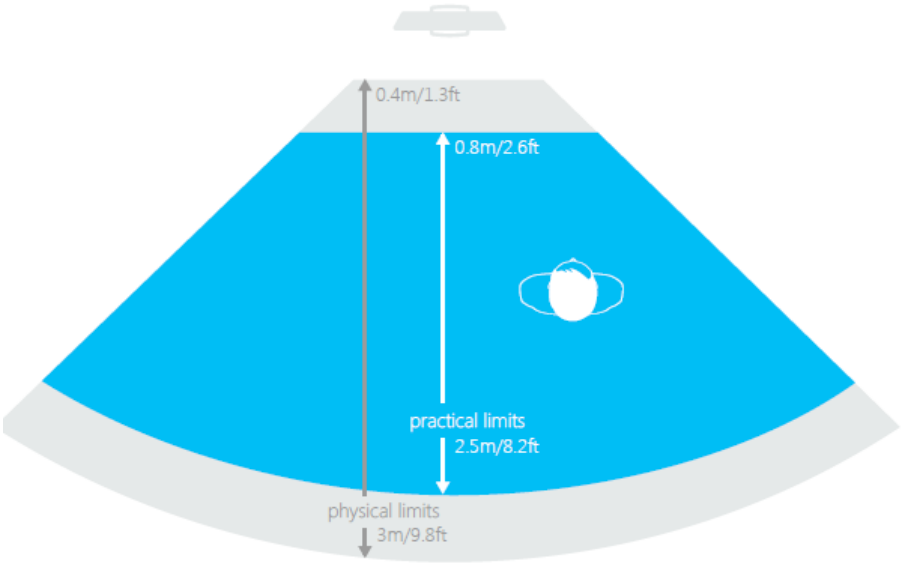


Figura 4.4: Rango de distancias en profundidad (modo por cercano)

- Flujo de datos
  - 320x240 a 16-bit en profundidad (30 fps).
  - 640x480 a 32-bit en color (30 fps).
  - 16-bit de audio (16 kHz).

Por otro lado, existe un proyecto en código abierto de gran escala para el procesamiento de nubes de puntos n-dimensionales (n-D) y el tratamiento de imágenes 2D/3D. Este proyecto proporciona una librería llamada *Point Cloud Library* (PCL) [19], la cual se distribuye bajo licencia BSD y es, por tanto, gratis para uso comercial y de investigación. Además, está completamente integrada con ROS (*Robot Operating System*, Sistema operativo de robots) y ha sido utilizada en varios proyectos de robótica.

PCL es multi-plataforma y ha sido implementada con éxito en Windows, MacOS, Linux y Android. Está implementado en C++ y está divida es una serie de librerías modulares que se pueden compilar por separado (ver Figura 4.5). Algunos de los más importantes son: filtros, características, registrado, kdtree, octree, segmentación, superficie, entrada/salida, etc.

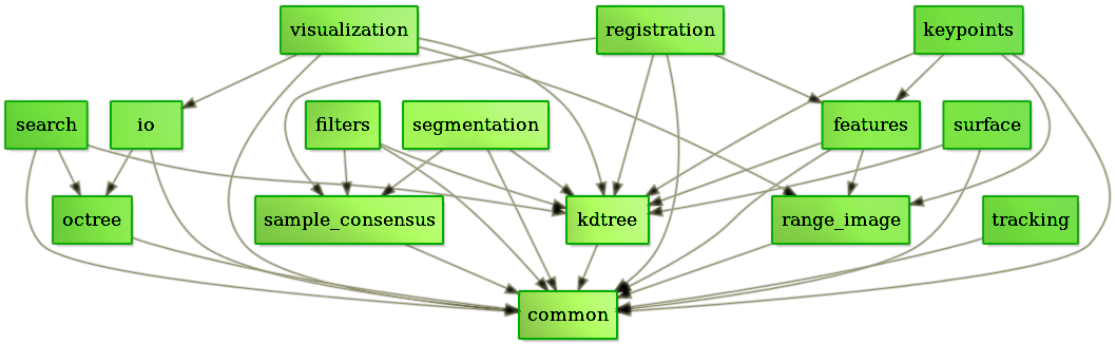


Figura 4.5: Diseño modular de PCL

PCL soporta de forma nativa la interfaz 3D OpenNI[20] y, por ello, puede adquirir y procesar datos de este tipo de dispositivos como las cámaras 3D PrimeSensor, el Microsoft Kinect o el Asus XTionPRO.

4.1.2. Pirámide espacial

Existe una implementación en Matlab de la pirámide espacial descrita anteriormente (ver punto 2.3), desarrollada por Joe Tighe y Svetlana Lazebnik. Este código lleva a cabo todos los pasos necesarios para construir una pirámide espacial basada en características SIFT.

Primero, extrae los descriptores SIFT de cada imagen. Después realiza un clustering con k-medias para generar el diccionario. A cada descriptor SIFT se le asigna una etiqueta que corresponde a la palabra más cercana en el diccionario. Finalmente, se genera la pirámide espacial a partir de dichas etiquetas.

Cada uno de estos pasos está dividido en funciones individuales que se pueden llamar de forma independiente, siempre que los datos del paso anterior estén disponibles. Las funciones son:

- `GenerateSiftDescriptors`: Genera los descriptores SIFT para cada imagen.
- `CalculateDictionary`: Carga los descriptores SIFT de un conjunto de imágenes. Después se ejecuta el algoritmo de *clustering* k-medias en los descriptores para encontrar N centros, donde N es el tamaño del diccionario.
- `BuildHistograms`: Para cada imagen, se cargan sus descriptores SIFT y se etiqueta cada uno de ellos. Entonces se calcula el histograma global de la imagen.
- `CompilePyramid`: Genera la pirámide a partir de las etiquetas del diccionario.

Durante este proceso se utilizan dos tipos de datos. En la Figura 4.6, se pueden ver los campos de cada uno de ellos.

```
features:
  data: matriz NxM de características de la imagen, donde N es el
        número de características en la imagen y M es el tamaño de
        del vector características
  x: vector Nx1 de coordenadas x de la imagen, donde N es el
     número de características de la imagen
  y: vector Nx1 de coordenadas y de la imagen, donde N es el
     número de características de la imagen
  wid: ancho de la imagen
  hgt: alto de la imagen

texton_ind:
  data: vector Nx1 de las etiquetas correspondientes a la palabra
        del diccionario más cercana para cada característica de la
        imagen, donde N es el número de características
  x: vector Nx1 de coordenadas x de la imagen, donde N es el
     número de características de la imagen
  y: vector Nx1 de coordenadas y de la imagen, donde N es el
     número de características de la imagen
  wid: ancho de la imagen
  hgt: alto de la imagen
```

Figura 4.6: Estructuras de datos en el código para generar la pirámide espacial

4.1.3. DOGMA

Finalmente, el último elemento necesario para llevar a cabo las pruebas es el clasificador.

DOGMA (*Discriminative Online (Good?) Matlab Algorithms*) es una herramienta en MATLAB, desarrollada por Francesco Orabona, para el aprendizaje online discriminativo [14]. Implementa los algoritmos actuales en un único framework. Su principal objetivo es la simplicidad, todos los algoritmos implementados son fáciles de usar, entender y modificar. Es por este motivo, que todas las implementaciones están realizadas en MATLAB, limitando el uso de archivos mex (permiten integrar código escrito en otros lenguajes de programación) sólo cuando es estrictamente necesario.

La librería se centra en algoritmos online de kernel y lineales. Algunos ejemplos son: *Perceptron*, *Passive-Aggresive*, *ALMA*, *NORMA*, *SILK*, *Projecton*, *RBP*, *Banditron*, *OISVM*, etc.

En definitiva, DOGMA proporciona un conjunto de herramientas que permiten usar y comparar de forma sencilla distintos tipos de algoritmos de aprendizaje, y así poder determinar cuál de ellos se ajusta mejor al problema concreto. Aunque en este proyecto se utilizará únicamente uno de los algoritmos. En concreto, se ha elegido esta herramienta para clasificar las muestras porque incluye la implementación de OISVM.

4.1.4. Base de datos: ImageCLEF 2012

Para poder realizar las pruebas de forma exhaustiva en necesario contar con una buena base de imágenes. Existen varias bases de datos de las que obtener esta información.

Un ejemplo es la base de datos KTH-IDOL2 [21]. El nombre IDOL es un acrónimo de *Image Database for rObot Localization* (Base de datos de imágenes para localización en robots). La finalidad de la misma es evaluar la robustez y adaptabilidad de los algoritmos de reconocimiento de lugares basados en visión, a los cambios ambientales del mundo real.

Esta base de datos contiene 24 secuencias de imágenes acompañadas de escáneres láser y datos odométricos adquiridos mediante dos plataformas robóticas móviles. Las imágenes corresponden a un laboratorio de interior con cinco habitaciones con diferente funcionalidad (oficina de una persona, oficina de dos personas, pasillo, cocina y sala de impresión) y bajo distintas condiciones de iluminación (tiempo nublado, soleado y de noche).

En la Figura 4.7 se pueden observar ejemplos de imágenes de dos de las habitaciones con cambios de iluminación.

Figura 4.7: Ejemplos de imágenes de la base de datos KTH-IDOL2 con diferentes tipos de iluminación

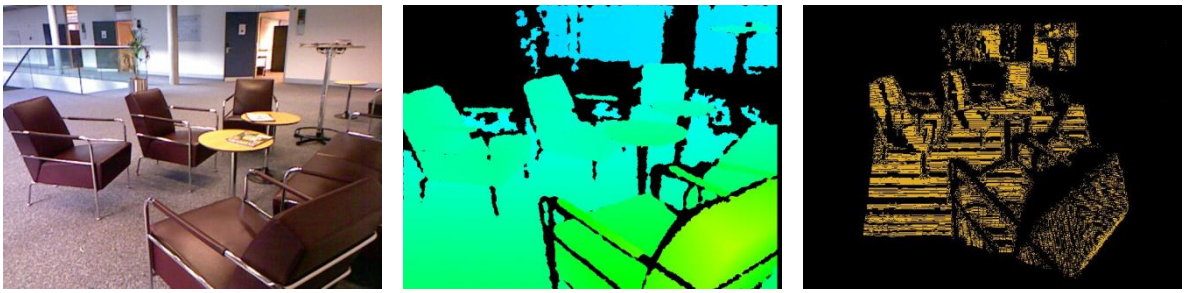
Sin embargo, para realizar las pruebas pertinentes en este proyecto se van a utilizar las imágenes del desafío de Visión Robótica (Robot Vision) del ImageCLEF 2012 [22].

ImageCLEF (*Image Retrieval in CLEF*) se lanzó en 2003 como parte del Foro de Evaluación Multi-Lenguaje (CLEF, *Cross Language Evaluation Forum*) [23] con el objetivo de proporcionar apoyo para la evaluación de:

- Métodos independientes del lenguaje para anotar imágenes con conceptos.
- Métodos de captura de información multimodal basados en la combinación de características visuales y textuales.
- Métodos de captura de imágenes multi-lenguaje para poder compararlos.

El desafío de Visión Robótica del ImageCLEF 2012 propone la clasificación de lugares utilizando imágenes obtenidas con el sensor de profundidad de Kinect.

Para esto, proporcionan diversas secuencias de imágenes capturadas con cámaras de perspectiva y un robot móvil con un sensor Kinect incorporado. Se proporcionan imágenes visuales (RGB) e imágenes de profundidad generadas a partir de nubes de puntos 3D (Figura 4.8).



Visual Profundidad Nube de puntos 3D

Figura 4.8: Tipos de imágenes

Estas secuencias se han obtenido en el mismo entorno, un edificio de oficinas, pero con distintos parámetros en las condiciones de iluminación (soleado, nublado, de noche) o del procedimiento de adquisición (en sentido de las agujas del reloj o en sentido contrario). Las secuencias son:

- **Training 1**: 2667 imágenes (nublado).
- **Training 2**: 2532 imágenes (nublado).
- **Training 3**: 1913 imágenes (de noche).

- **Test 1**: 2445 imágenes (de noche).
- **Test 2**: 4023 imágenes (nublado).

Las imágenes se pueden clasificar en una de estas nueve categorías:

- Pasillo
- Ascensor
- Sala de impresión
- Sala de descanso
- Oficina de profesor
- Oficina de estudiante
- Sala de conferencias
- Sala técnica
- Baño

En la Figura 4.9 se puede observar la diferencia entre las habitaciones de las categorías, ya que se muestra un ejemplo de imagen visual de cada una de ellas.

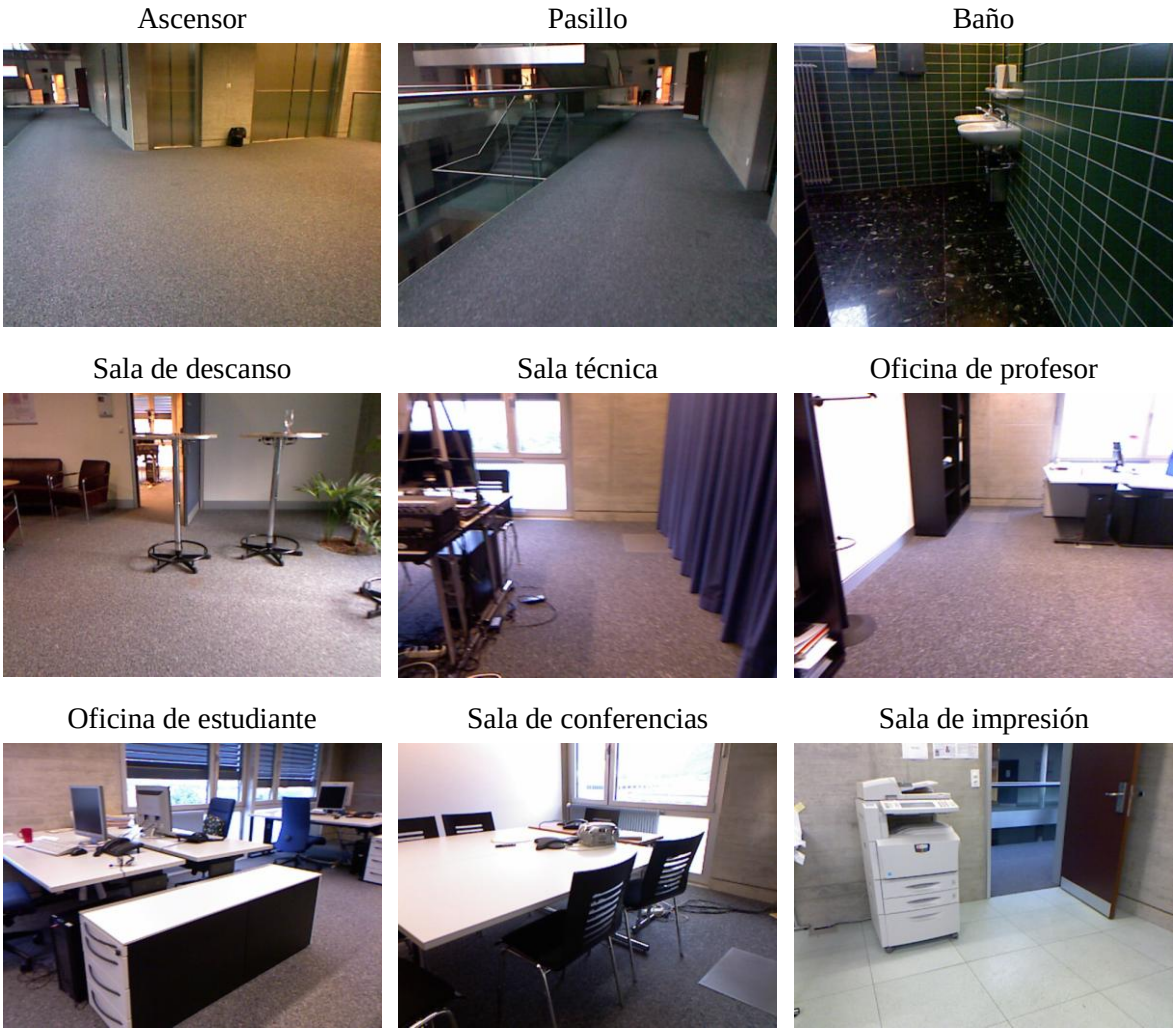


Figura 4.9: Imágenes de ejemplo de la base de datos de Robot Visio 2012

La distribución de estas categorías en los conjuntos de entrenamiento se puede observar en la Tabla 4.1.

	Número de frames		
Categorías	Training 1	Training 2	Training 3
Pasillo	438	498	4444
Ascensor	140	152	84
Sala de impresión	119	80	65
Sala de descanso	421	452	376
Oficina de profesor	408	336	247
Oficina de estudiante	664	599	388
Sala de conferencias	126	79	60
Sala técnica	156	96	118
Baño	198	240	131
Todas	2667	2532	1913

Tabla 4.1: Distribución de categorías en los conjuntos de entrenamiento

Para todas estas imágenes se tiene dos tipos de descriptores que permitirán comparar resultados con información visual y de profundidad:

- *Características visuales*: Descriptores PHOG (ver apartado 2.1.2).

- *Características de profundidad*: Descriptores NARF (ver apartado 2.1.3).

Los conjuntos de imágenes (*datasets*) tanto de entrenamiento como de prueba están incluidos en el DVD que acompaña a esta memoria.

5. METODOLOGÍA Y RESULTADOS

En este capítulo se va a detallar cómo se han utilizado las librerías y herramientas para desarrollar un sistema de clasificación de imágenes con información de profundidad. Posteriormente se explicará qué pruebas se han realizado y se expondrán los resultados obtenidos.

5.1. SISTEMA DE CLASIFICACIÓN CON IMÁGENES DE PROFUNDIDAD

El sistema de clasificación diseñado se puede dividir en los siguientes pasos:

1. Obtención de las imágenes.
2. Extracción de características NARF.
3. Generación de una pirámide espacial con la información de las imágenes.
4. Utilizar dicha para pirámide para entrenar un algoritmo de aprendizaje, en este caso OISVM.

El paso siguiente sería probar el algoritmo entrenado para comprobar la efectividad del sistema ante nuevas imágenes. Dichos resultados se expondrán en el apartado 5.2.

5.1.1. Captura de imágenes y extracción de características

El primer paso del procesamiento de imágenes es la captura de las mismas. Para este proyecto nos interesan las imágenes de profundidad, por eso es necesario un sensor 3D. Kinect para Xbox 360, ofrece unas buenas características técnicas a un precio accesible. Es por eso que es uno de los dispositivos más utilizados.

Además, es posible gestionar la captura de imágenes con Kinect mediante PCL. De este modo, es posible:

1. Capturar imágenes con Kinect.
2. Procesarlas con PCL para extraer las características NARF de las mismas [24].
3. Generar un archivo de Matlab con la información de las características.

Las imágenes utilizadas serán las que se proporcionan en el desafío de Visión Robótica de ImageCLEF 2012 [22], que se han obtenido haciendo uso de Kinect y PCL.

5.1.2. Pirámide espacial con información de profundidad

El código original de la pirámide espacial llevaba a cabo cuatro pasos para generar el resultado final:

1. Generar los descriptores SIFT.
2. Construir el diccionario.
3. Construir los histogramas.
4. Construir la pirámide.

En la implementación actual se van a realizar distintos cambios que se enumeran a continuación:

- *Uso de características NARF*. Con el uso de Kinect y PCL se obtiene directamente la estructura con la información de las características para utilizar en la generación de la pirámide. Puesto que es posible proporcionar los datos necesarios para el siguiente paso en todos los puntos del proceso, se va a eliminar el primer paso del mismo, la generación de descriptores SIFT. Es decir, la extracción de características pasará a ser un paso previo a la construcción de la pirámide, en vez del paso inicial.

- *Modificación de las estructuras de datos para añadir una tercera coordenada*. Como ya se ha indicado anteriormente (ver punto 4.1.2), se utilizan una serie de estructuras para almacenar la información necesaria. En dichas estructuras están incluidas las coordenadas de las características, así como el ancho y alto de la imagen.

  En concreto, es necesario añadir la coordenada *z* de las características y el valor máximo de profundidad, que en este caso es 10 debido a las características de Kinect. Al añadir los nuevos datos, las estructuras resultantes serán las que se pueden ver en la Figura 5.1.

- *Uso de la nueva información para generar la pirámide*. Es necesario tener en cuenta los nuevos datos para generar la pirámide. En concreto, se debe prestar especial atención al tamaño de la pirámide y a la división espacial de las

características. Estas modificaciones, que son las más importantes puesto que son las que generan la pirámide 3D, se efectúan en la función `CompilePyramid`.

```
features:
  data: matriz NxM de características de la imagen, donde N
        es el número de características en la imagen y M es
        el tamaño de del vector características
  x: vector Nx1 de coordenadas x de la imagen, donde N es el
     número de características de la imagen
  y: vector Nx1 de coordenadas y de la imagen, donde N es el
     número de características de la imagen
  z: vector Nx1 de coordenadas z de la imagen, donde N es el
     número de características de la imagen
  wid: ancho de la imagen
  hgt: alto de la imagen
  prf: profundidad de la imagen

texton_ind:
  data: vector Nx1 de las etiquetas correspondientes a la
        palabra del diccionario más cercana para cada
        característica de la imagen, donde N es el número de
        características
  x: vector Nx1 de coordenadas x de la imagen, donde N es el
     número de características de la imagen
  y: vector Nx1 de coordenadas y de la imagen, donde N es el
     número de características de la imagen
  z: vector Nx1 de coordenadas z de la imagen, donde N es el
     número de características de la imagen
  wid: ancho de la imagen
  hgt: alto de la imagen
  prf: profundidad de la imagen
```

Figura 5.1: Estructuras de datos para generar la pirámide espacial con información de profundidad. Los nuevos elementos están resaltados en negrita

Así el algoritmo final estaría compuesto por los siguientes pasos:

- `CalculateDictionary`: Carga los descriptores NARF de un conjunto de imágenes. Después se ejecuta k-medias en los descriptores para encontrar N centros, donde N es el tamaño del diccionario.
- `BuildHistograms`: Para cada imagen, se cargan sus descriptores NARF y se etiqueta cada uno de ellos. Entonces se calcula el histograma global de la imagen.
- `CompilePyramid`: Genera la pirámide a partir de las etiquetas del diccionario y teniendo en cuenta las coordenadas de las características, incluida la coordenada z.

De todo este proceso es muy importante el diccionario generado. Ya que contendrá las palabras que mejor representan la información contenida en las imágenes. En concreto, incluirá agrupaciones de descriptores similares, de modo que todos aquéllos descriptores que pertenezcan a un mismo grupo o palabra contendrán información similar y, en definitiva, representarán información parecida de la imagen.

Dicho diccionario se genera con los datos de entrenamiento y sirve para construir la pirámide de los mismos así como las pirámides de los conjuntos de pruebas.

5.1.3. Clasificación

Partiendo de la librería DOGMA, se va a utilizar una implementación OISVM que incluye una estrategia de reemplazo para los vectores base [25]. En concreto, añade un umbral para el número de vectores de entrenamiento y cada vez que se pasa esa cantidad se descartan vectores, que quedan reemplazados por los nuevos. Así se consigue controlar los requerimientos de memoria del algoritmo.

Al ser este un problema de clasificación multi-clase con estrategia todos-contra-uno, es necesario entrenar una OISVM por cada clase que se quiera poder identificar. Puesto que estos entrenamientos son independientes entre sí se ha preparado el código para poder realizarlos por separado y así, si se tienen los recursos necesarios, poder ejecutarlos en paralelo para acelerar el proceso.

Las entradas de este algoritmo serán las diferentes pirámides generadas según sus niveles y tamaño del diccionario. Además, también están disponibles las características visuales, por lo que se pueden realizar tres experimentos con cada configuración de la pirámide:

- Utilizando únicamente *características visuales*.

- Utilizando únicamente la pirámide espacial con *información de profundidad*.

- *Combinando ambos datos*. Esta combinación se realizará concatenando el vector de características visual al vector de características de profundidad que contiene la pirámide.

La salida de este proceso será una OISVM por cada clase, que posteriormente, se utilizará para clasificar los vectores de dos conjuntos de test.

La etapa de pruebas también se puede realizar de forma individual para cada clase. El único requisito para hacerlo de este modo es unir todos los resultados al final, ya que

estamos utilizando un método todos-contra-uno. Los resultados de cada clase nos indicarán únicamente, si el vector o pirámide de pruebas pertenece o no a la clase. Para determinar cuál es la categoría final que se asignará se debe comparar con sus resultados en el resto de clasificadores y escoger aquella clase con mayor valor de confianza, es decir, con mayor distancia al margen correspondiente.

5.2. RESULTADOS

Como ya se ha explicado anteriormente se han realizado dos experimentos: uno para poder evaluar como los distintos parámetros afectan al resultado final y otro para comprobar cuál sería el resultado obtenido al usar el clasificador en un caso más general.

Existen diferentes parámetros que son necesarios configurar para cada prueba:

- *Conjunto de entrenamiento*: Puede ser training1, training2, training3 o cualquier combinación entre ellos.

- *Combinación del conjunto de entrenamiento*: Se pueden utilizar características visuales (*visual*), de profundidad (*depth*) o una combinación de ambas (*both*).

- *Niveles de la pirámide (L)*: Número total de niveles. Hay que tener en cuenta que tenemos tres dimensiones y el tamaño de la pirámide crece exponencialmente con este parámetro, en concreto el total regiones será:

$$R = \sum_{l=1}^{L} 2^{3l} \tag{5.1}$$

- *Tamaño del diccionario (K)*: Número de palabras en que se van a agrupar los distintos descriptores. Este factor también afecta considerablemente al tamaño de la pirámide final, pues se concatena un histograma de tamaño K para cada región de la imagen. El tamaño total de la pirámide será:

$$T = K \cdot R = K \sum_{l=1}^{L} 2^{3l} \tag{5.2}$$

- *Conjunto de test*: Puede ser Test1 o Test2. Hay que tener en cuenta que ambos conjuntos tienen características distintas.

- *Combinación del conjunto de test*: Debe ser la misma que la del conjunto de entrenamiento, ya sea visual, de profundidad o ambos.

- *Subsampling*: Número máximo de vectores que se van a utilizar para el entrenamiento. La situación ideal sería poder entrenar con todo el conjunto de entrenamiento, pero por cuestiones de rendimiento no se ha podido abordar. Por este motivo es necesario establecer un valor máximo.

- *Número máximo de vectores de soporte (numMaxSV)*: Número máximo de vectores que formarán parte de los vectores base de la OISVM. Si se supera este número, se producirá un reemplazo de un vector almacenado.

Estos dos últimos parámetros afectan a la ejecución de la SVM, el resto definen la prueba que se va a realizar.

Por cuestiones de rendimiento las pruebas se han realizado con un conjunto reducido de vectores de entrenamiento, es por este motivo, que en todas las pruebas se ha podido establecer un número máximo de vectores soporte lo suficientemente alto para que no sea necesario ningún reemplazo.

5.2.1. Prueba 1: Múltiples parámetros de la pirámide con profundidad

En esta prueba se quiere comprobar el efecto de todos los parámetros de generación de la pirámide que pueden afectar a los resultados de clasificación. También se va a comparar los resultados que se obtienen con información visual y de profundidad cuando cambian las condiciones de iluminación entre el conjunto de entrenamiento y el de test.

Los parámetros que se han utilizado para las pruebas son:

- *Entrenamiento*: Training1 que tiene condiciones de iluminación ‘nublado’. Se han utilizado todas sus posibles combinaciones de características (*visual*, *depth* y *both*).

- *Pirámide*: 1, 2 y 3 niveles con 50, 100 y 200 palabras.

- *Pruebas*: Se van a realizar las pruebas con test1 y test2, ya que presentan distintas condiciones de iluminación.

- *Subsampling*: El tamaño de training1 es de 2667 elementos, por lo que no se considera necesario un subsampling.

Los resultados obtenidos para esta prueba son los que aparecen en la Tabla 5.1.

Entrenamiento	Test	Combinación	Niveles de la pirámide	Tamaño del diccionario	Tasa de acierto (%)
training1	test1	visual	0	0	19,02
training1	test1	depth	1	50	23,27
training1	test1	both	1	50	18,61
training1	test1	depth	1	100	23,80
training1	test1	both	1	100	22,21
training1	test1	depth	1	200	28,59
training1	test1	both	1	200	24,17
training1	test1	depth	2	50	22,70
training1	test1	both	2	50	20,29
training1	test1	depth	2	100	24,66
training1	test1	both	2	100	22,62
training1	test1	depth	2	200	28,88
training1	test1	both	2	200	24,05
training1	test1	depth	3	50	25,03
training1	test1	both	3	50	23,72
training1	test1	depth	3	100	27,73
training1	test1	both	3	100	25,69
training1	test1	depth	3	200	30,35
training1	test1	both	3	200	25,07
training1	test2	visual	0	0	49,89
training1	test2	depth	1	50	43,18
training1	test2	both	1	50	61,42
training1	test2	depth	1	100	42,88
training1	test2	both	1	100	61,50
training1	test2	depth	1	200	44,17
training1	test2	both	1	200	60,38
training1	test2	depth	2	50	46,36
training1	test2	both	2	50	64,35
training1	test2	depth	2	100	46,56
training1	test2	both	2	100	62,39
training1	test2	depth	2	200	46,66
training1	test2	both	2	200	60,65
training1	test2	depth	3	50	50,01
training1	test2	both	3	50	62,44
training1	test2	depth	3	100	49,42
training1	test2	both	3	100	61,60
training1	test2	depth	3	200	48,60
training1	test2	both	3	200	59,58

Tabla 5.1: Resultados de training1 con distintas opciones. Está resaltado en rojo el peor resultado de cada conjunto de prueba y en verde el mejor

Análisis de los resultados de test2

En primer lugar se analizará los resultados obtenidos con test2. En este caso, como se puede observar, la tasa de aciertos es buena, superando varias configuraciones el 60% de aciertos.

La tasa de acierto más alta (64.35%) se consigue con 2 niveles de pirámide y 50 palabras en el diccionario, utilizando una combinación de información visual y de profundidad. Por otro lado, la más baja (42.88%) se genera al utilizar 1 nivel con 100 palabras y únicamente información de profundidad.

Tanto training1 como test2 tienen las mismas condiciones de iluminación. Por ese motivo, las características visuales ofrecen buenos resultados que, las características de profundidad no pueden alcanzar por ellas mismas. Sin embargo, es necesario resaltar la importante mejora de utilizar los dos tipos de datos de forma combinada.

Teniendo en cuenta estas consideraciones, se puede concluir que:

- Utilizar sólo las características de profundidad empeora los resultados de las visuales, pero al combinar las dos opciones mejora la eficiencia de ambas por separado.

- Como se puede ver en la Figura 5.2, aumentar el número de niveles mejora la eficacia de la información de profundidad. Sin embargo, añadir un tercer nivel utilizando ambos tipos de características no mejora los resultados obtenidos para el mismo tipo de información con dos niveles.

  En este aspecto, hay que señalar que, con información de profundidad y tres niveles en la pirámide, los resultados se aproximan a los obtenidos con las características visuales. Es posible que añadir un nivel más mejorase estos datos, pero en ese caso habría que considerar si dichos resultados compensan los requisitos computacionales que necesitaría, pues el tamaño de la pirámide crece de forma exponencial.

- Por otro lado, en la Figura 5.3, se puede observar que, ante las mismas condiciones de iluminación, aumentar el tamaño del diccionario, por lo general, no mejora los resultados, llegando a empeorar ligeramente en la mayoría de los casos.

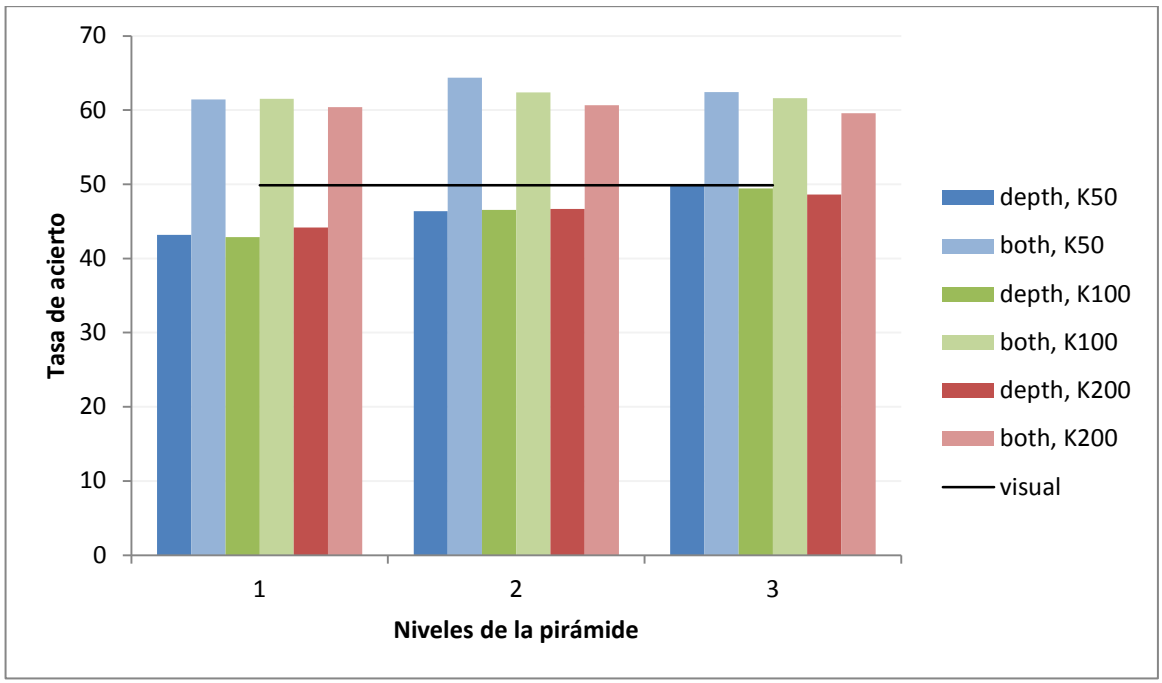


Figura 5.2: Evolución de la tasa de aciertos según los niveles de la pirámide con test2

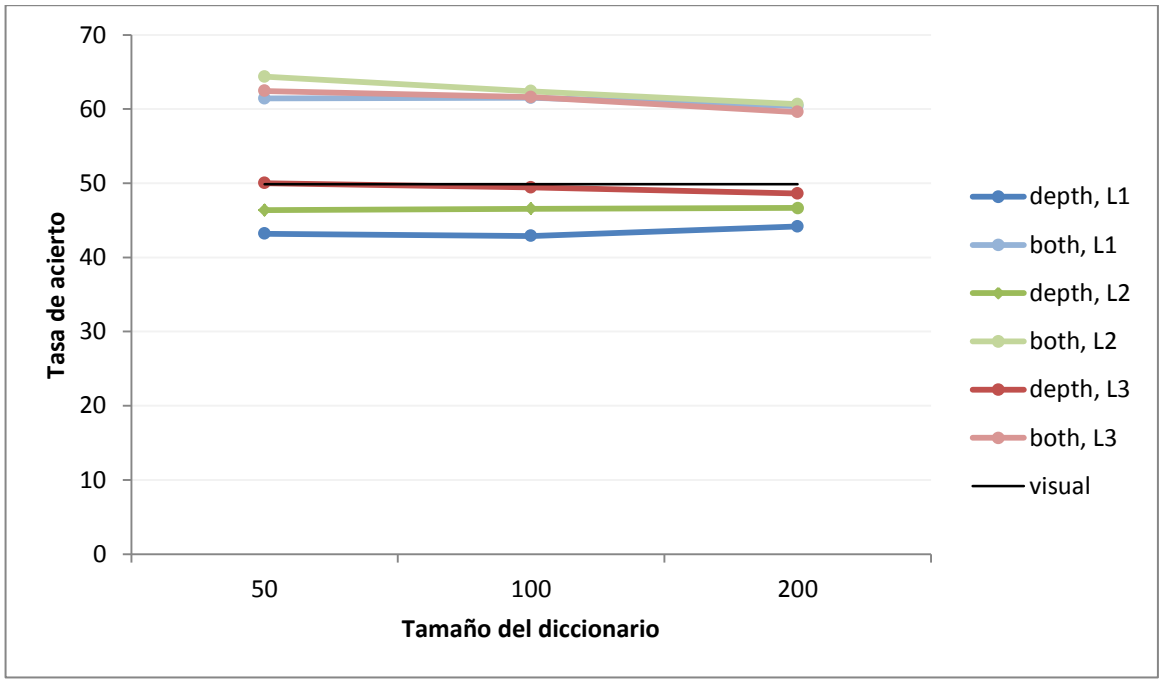


Figura 5.3: Evolución de la tasa de aciertos según el tamaño del diccionario con test2

Análisis de los resultados de test1

Por otro lado, los resultados de test1 conducen a unas conclusiones distintas. En este caso, el conjunto de prueba tiene unas condiciones de iluminación distintas a las del conjunto de entrenamiento. Este hecho es el que provoca unas tasas de acierto tan bajas, se está clasificando ejemplos para los que no había muestras de entrenamiento.

En este caso, la tasa de aciertos más alta es del 30.35% y se consigue con características de profundidad, a 3 niveles y con 200 palabras. La tasa más baja (18.61%) se obtiene con un nivel y 50 palabras, utilizando información visual y de profundidad.

En este caso, la información de profundidad, que con las mismas condiciones de iluminación no ofrecía mejoras por sí misma, mejora los resultados de las características visuales, ya que este tipo de datos no son tan susceptibles a cambios de iluminación.

Es también destacable, sobre todo comparándolo con el caso anterior, que la combinación de características visuales y de profundidad no mejora los resultados de las de profundidad solas. En principio, al tener más información se debería mejorar la identificación de los puntos, pero los pobres resultados de la información visual en este caso impide dicha mejora.

En definitiva, a partir de los resultados de test1 se puede ver que:

- Los resultados obtenidos con test2 son notablemente mejores que con test1, porque test2 tiene las mismas condiciones de iluminación que training1.

- Se mejoran los resultados de división espacial si se considera también la información de profundidad. Esto es debido a que las condiciones de iluminación son distintas en el conjunto de entrenamiento y en el de pruebas. Como resultado, las características visuales, que son muy susceptibles a cambios de iluminación, consiguen unos resultados peores.

  Este hecho también se evidencia en que, en la mayoría de los casos, el utilizar una combinación de información visual y de profundidad empeora los resultados.

- En este caso, la información adicional sí que mejora la clasificación y, como se puede ver en la Figura 5.4, al aumentar los niveles de la pirámide se mejoran los resultados obtenidos, tanto para características de profundidad como para su combinación con visuales.

- Del mismo modo, aumentar el tamaño del diccionario mejora la clasificación por lo general (ver Figura 5.5).

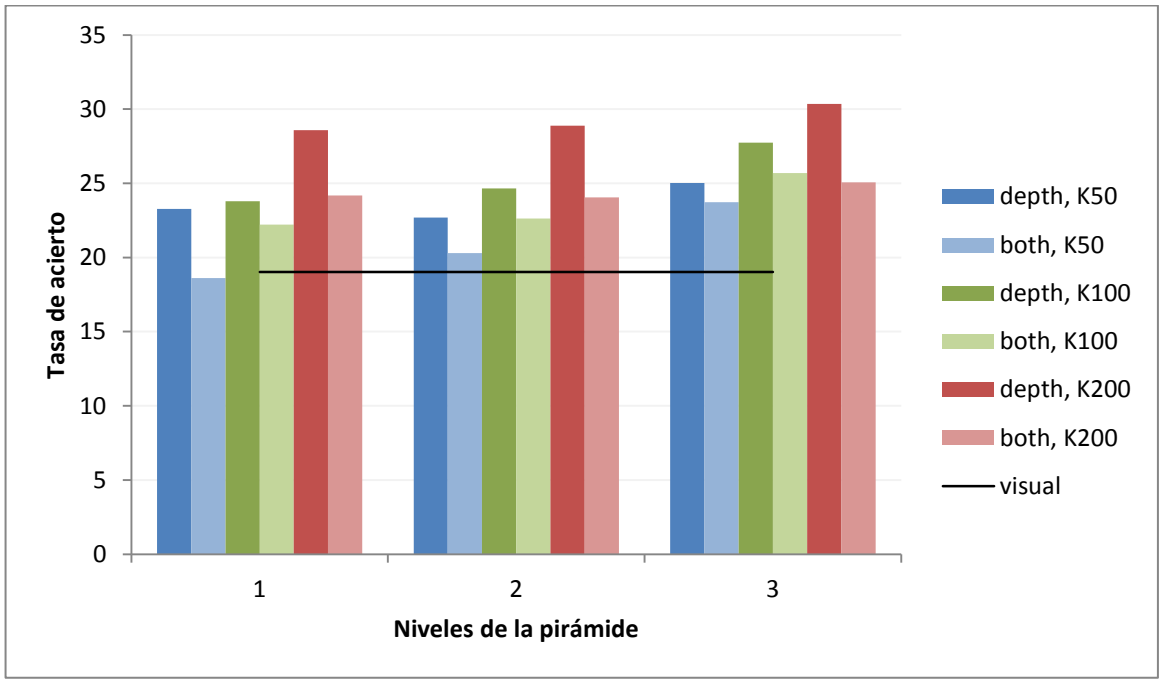


Figura 5.4: Evolución de la tasa de aciertos según los niveles de la pirámide con test1

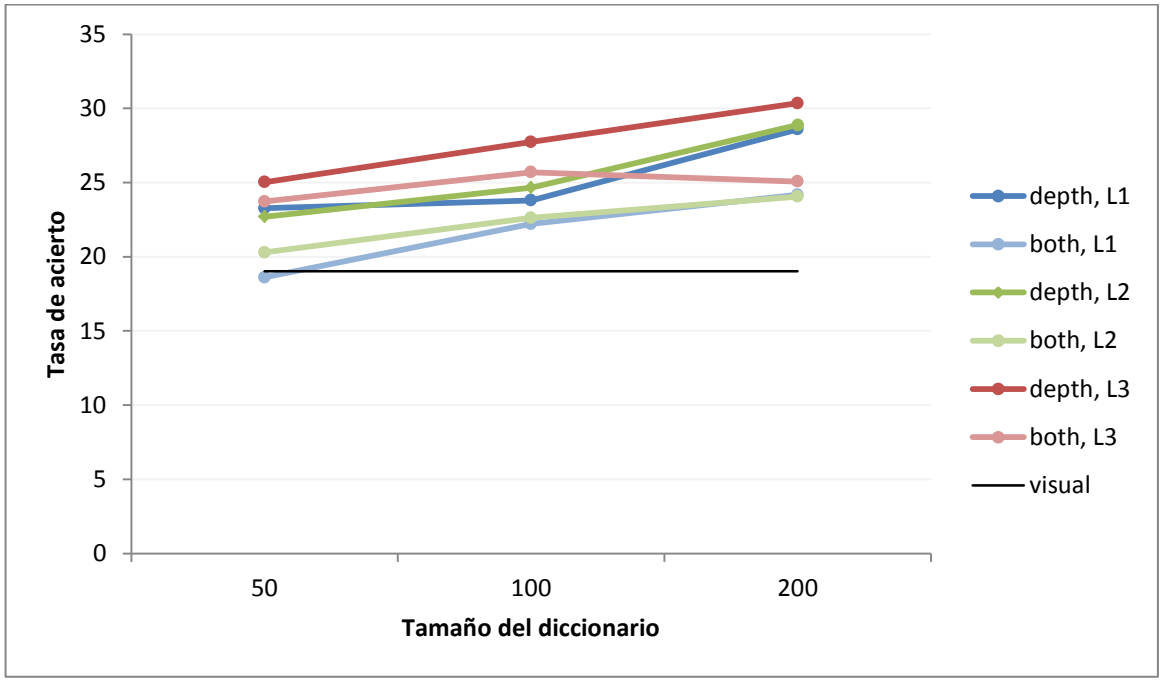


Figura 5.5: Evolución de la tasa de aciertos según el tamaño del diccionario con test1

5.2.2. Prueba 2: Resultados globales

En esta prueba queremos comprobar el comportamiento global de la propuesta, partiendo de información que se puede aplicar a los dos conjuntos de pruebas. En este caso, los parámetros de las pruebas son:

- *Entrenamiento*: Todos los conjuntos de entrenamiento (training1, training2 y training3) con todas sus posibles combinaciones de características (*visual*, *depth* y *both*).

- *Pirámide*: 100 palabras y 2 niveles.

- *Pruebas*: Se van a realizar las pruebas con test1 y test2.

- *Subsampling*: El uso de los tres conjuntos de entrenamiento genera 7112 elementos. Este número es demasiado grande para realizar un entrenamiento completo, por ese motivo se selecciona un subconjunto de 2370 vectores, un tercio del conjunto inicial.

  El subsampling se realiza seleccionando los vectores de entrenamiento de forma uniforme ya que las secuencias están ordenadas. De este modo nos aseguramos de que habrá muestras de todas las clases.

Los resultados obtenidos son los que se pueden observar en la Tabla 5.2:

	Resultados (% aciertos) (Subsampling: 2370)	
Train1,2,3 vs	Test1	Test2
Visual	44.16%	50.56%
Depth	37.34%	44.64%
Visual + Depth	51.57%	61.62%

Tabla 5.2: Tasa de aciertos en un caso general con distintos tipos de iluminación durante el enternamiento

De estos resultados se pueden sacar varias conclusiones:

- Con test2 se obtienen mejores resultados en todos los casos. En esta ocasión, puede deberse a dos motivos:

a. El conjunto de entrenamiento contiene más ejemplos de imágenes con las mismas características de iluminación que test2, por lo que el entrenamiento ha sido más exhaustivo.

b. La secuencia de imágenes de test2 incluye secuestros, esto es que las imágenes pasan de forma brusca de una clase a la siguiente. En términos de un robot que está escaneando una habitación sería impedir que capte imágenes con el sensor, llevarlo a otra habitación y permitir que vuelva a escanear el entorno, en otras palabras "secuestrarlo" (ver Figura 5.6).

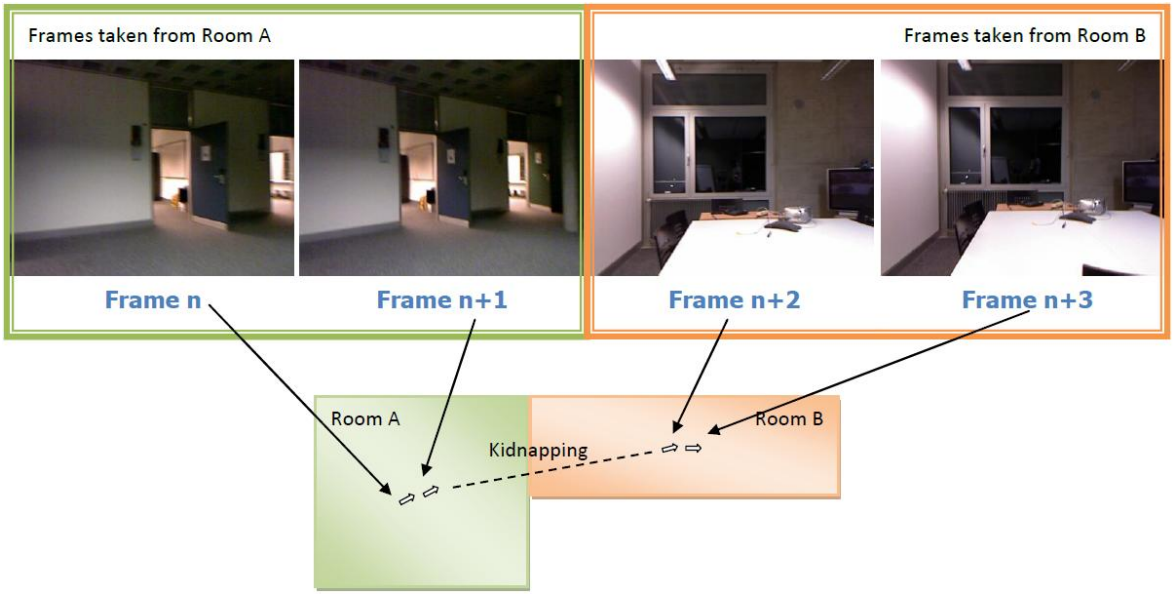


Figura 5.6: Ejemplo de secuestro en test2

Sin embargo, en test1 el cambio se produce de forma gradual (ver Figura 5.7). En este caso, un pequeño número de *frames* muestran una transición.

- A pesar del aparente empeoramiento que produce el uso de información de profundidad, cabe destacar la gran mejora que se produce al combinar esta información con la visual.

Tanto en los ejemplos de esta prueba, como en la anterior, los resultados obtenidos con ambos tipos de datos mejoran sustancialmente los obtenidos por cada uno de ellos de forma individual, incluidos aquellos que sólo utilizan información visual.

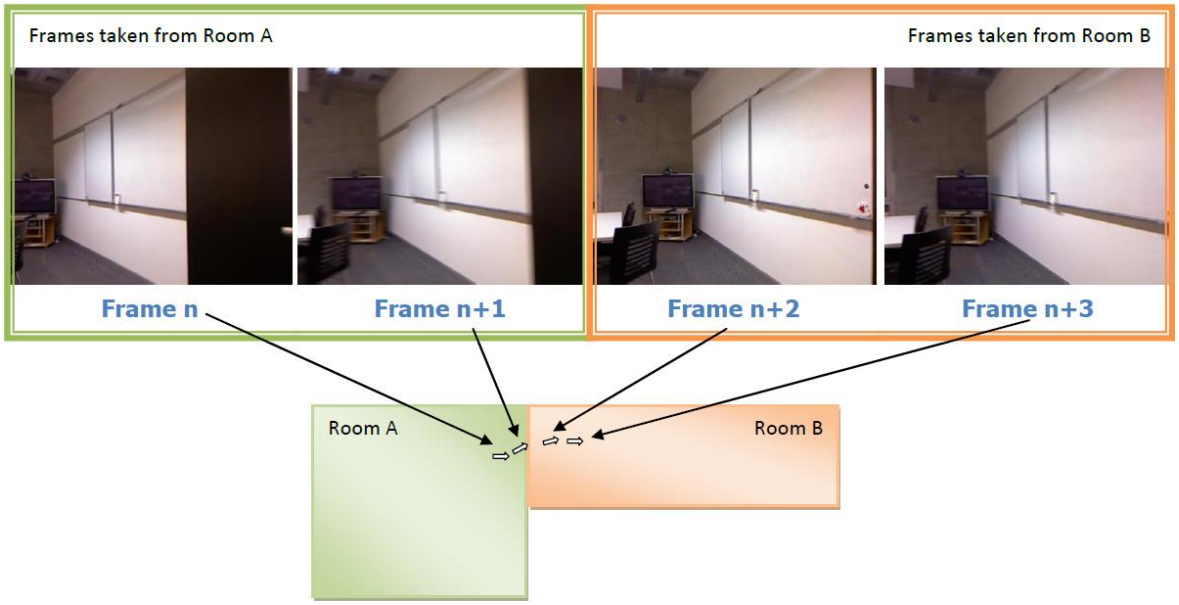


Figura 5.7: Transición entre habitaciones en test1

6. CONCLUSIONES Y PROPUESTAS

Finalmente, en este capítulo se va detallar las conclusiones que las distintas pruebas realizadas han permitido realizar. Además, se propondrán diversas mejoras que se podrían llevar a cabo en el futuro.

6.1. CONCLUSIONES

La clasificación de escenas visuales es un campo ampliamente investigado en el mundo de la visión por computador. Sin embargo, la mayoría de propuestas hasta ahora se centran en la clasificación de espacios visuales.

A lo largo de este proyecto, se ha integrado la información de profundidad en un proceso de clasificación de escenas de interior. Se ha utilizado un proceso que ha demostrado buenos resultados subdividiendo imágenes y aprovechando así la geometría de las mismas. Posteriormente, se ha utilizado dicha información para clasificar las escenas que representaban.

Los resultados obtenidos demuestran que este enfoque mejora los conseguidos con el uso exclusivo de información visual. En concreto, se ha demostrado que el uso de información de profundidad mejora considerablemente la clasificación si nos encontramos ante distintas condiciones de iluminación.

Esta mejora es muy importante de cara a su utilización en sistemas de clasificación que deben responder en tiempo real, como sería el mecanismo de visión de un robot. Un robot móvil se enfrenta a numerosas situaciones donde las condiciones del entorno cambian, cambia la iluminación, el ángulo, etc. así que es importante contar con sistema que ofrezca cierta robustez ante ese tipo de modificaciones.

Sin embargo, el punto más destacable de los resultados obtenidos es la diferencia que se produce al utilizar una combinación de información visual y de profundidad. Siendo este caso, por lo general, el que ofrece mejores tasas de acierto.

6.2. TRABAJOS FUTUROS

Aunque la propuesta ha permitido mostrar los resultados deseados en cuando a clasificación de escenas con distintas condiciones de iluminación. Todavía existe un amplio margen de mejora en la tasa global de aciertos. Es por este motivo, que a

continuación se detallan algunas posibles mejoras y las consideraciones que deben tenerse en cuenta para cada una de ellas.

- **Pirámide n-D**. Una posible mejora de esta aproximación sería generar una pirámide espacial multi-dimensional, de modo que, además de subdividir la imagen y asignar las características según las coordenadas geométricas, se pueden añadir nuevos valores de división, como por ejemplo, niveles de color. De este modo la subdivisión tendría en cuenta más información.

  El principal problema de esta propuesta sería que el tamaño de la pirámide crece de manera exponencial con el número de dimensiones.

- ***Clustering* supervisado**. En la actualidad, el *clustering* de los descriptores para obtener las palabras del diccionario se realiza con k-medias. Los resultados muestran que, dependiendo de las condiciones de iluminación, al aumentar el tamaño del diccionario no tienen porqué obtenerse mejores resultados.

  Una posible mejora sería poder establecer un *clustering* guiado, de modo que, con un número más reducido de palabras se puedan obtener resultados más significativos, pues dichas palabras serían las más relevantes para el problema concreto.

- **Distintas divisiones de la pirámide**. Aunque la división espacial geométrica ofrece buenos resultados puede que si se consideran otro tipo de divisiones para generar la pirámide se consigan mejores resultados.

  Es posible que, debido a las características de las imágenes sea interesante realizar la división sólo en algunas de las dimensiones o con distinto tamaño (ver Figura 6.1). Con esta aproximación se podría conseguir mantener la división espacial centrándose en las zonas de más interés. Además, al reducir el número de divisiones, se reducirá el tamaño de la pirámide final.

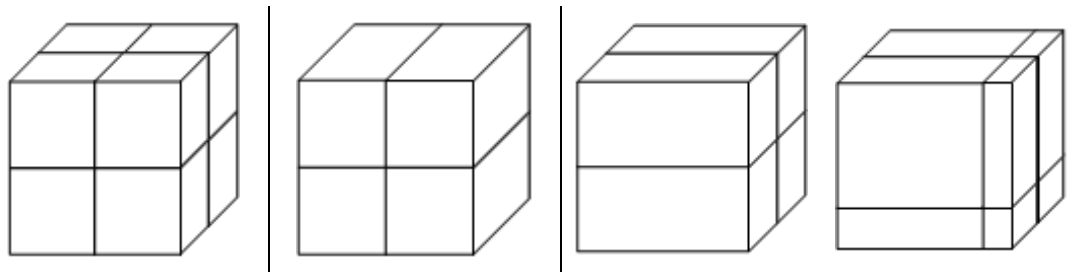


Figura 6.1: Posibles divisiones espaciales

- **Modificación de las características**. Resulta evidente que para obtener buenos resultados del clasificador es necesario tener una buena representación de la información. Es por este motivo, que una posible mejora sería cambiar las características NARF por otro tipo de características que se ajusten mejor a los datos que se van a utilizar.

7. BIBLIOGRAFÍA

[1] R. C. González y R. E. Woods, Tratamiento digital de imágenes, Wilmington, Delaware: Addison-Wesley/Díaz de Santos, 1996.

[2] D. G. Lowe, «Distinctive Image Features from Scale-Invariant Keypoints,» *International Journal of Computer Vision,* nº 60, pp. 91-110, 2004.

[3] H. Bay, T. Tuytelaars y L. Van Gool, «SURF: Speeded Up Robust Features,» *Computer Vision and Image Understanding (CVIU),* vol. 110, nº 3, pp. 346-359, 2008.

[4] A. Bosch, A. Zisserman y X. Munoz, «Representing shape with a spatial pyramid kernel,» de *ACM International Conference on Image and Video Retrieval*, 2007.

[5] A. E. Johnson y M. Hebert, «Using Spin-Images for Efficient Object Recognition in Cluttered 3-D Scenes,» *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 21, nº 5, pp. 433-449, 1999.

[6] S. Stiene, K. Lingemann, A. Nüchter y J. Hertzberg, «Contour-based Object Detection in Range Images,» de *Third International Symposium on 3D Data Processing, Visualization, and Transmission*, 2006.

[7] B. Steder, R. B. Rusu, K. Konolige y W. Burgard, «Point Feature Extraction on 3D Range Scans Taking into Account Object Boundaries,» de *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[8] D. Michie, D. J. Spiegelhalter y C. C. Taylor, Machine Learning, Neural and Statistical Classification, 1994.

[9] T. M. Mitchell, Machine Learning, McGraw-Hill, 1997.

[10] E. Alpaydin, Introduction to Machine Learning, The MIT Press, 2004.

[11] S. Russell y P. Norvig, Inteligencia Artificial: Un Enfoque Moderno, 2ª ed., Madrid: Prentice Hall, 2004.

[12] C. Cortes y V. Vapnik, «Support-Vector Networks,» *Machine Leaming,* vol. 20, nº 3, pp. 273-297, 1995.

[13] F. Orabona, C. Castellini, B. Caputo, L. Jie y G. Sandini, «On-line Independent Support Vector Machines,» *Pattern Recognition,* vol. 43, nº 4, pp. 1402-1412, 2010.

[14] F. Orabona, «DOGMA: a MATLAB toolbox for Online Learning,» 2009. [En línea]. Available: http://dogma.sourceforge.net/. [Último acceso: Agosto 2012].

[15] S. Lazebnik, C. Schmid y J. Ponce, «Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories,» de *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[16] K. Grauman y T. Darrell, «Pyramid Match Kernels: Discriminative Classification with Sets of Image Features,» de *IEEE International Conference on Computer Vision (ICCV)*, 2005.

[17] «Componentes del sensor de Kinect,» [En línea]. Available: http://support.xbox.com/es-ES/kinect/setup-and-playspace/kinect-sensor-components. [Último acceso: Agosto 2012].

[18] Microsoft, «Human Interface Guidelines. Kinect for Windows v1.5.0,» [En línea]. Available: http://download.microsoft.com/download/B/0/7/B070724E-52B4-4B1A-BD1B-05CC28D07899/Human_Interface_Guidelines_v1.5.0.pdf. [Último acceso: Agosto 2012].

[19] R. B. Rusu y S. Cousins, «3D is here: Point Cloud Library (PCL),» de *IEEE International Conference on Robotics and Automation (ICRA)*, Shangai, China, 2011.

[20] «OpenNI,» [En línea]. Available: http://www.openni.org/. [Último acceso: Septiembre 2012].

[21] J. Luo, A. Pronobis, C. Barbara y P. Jensfelt, «The KTH-IDOL2 Database,» Kungliga Tekniska Hoegskolan, 2006.

[22] ImageCLEF 2012, «Robot Vision 2012,» [En línea]. Available: http://www.imageclef.org/2012/robot. [Último acceso: Agosto 2012].

[23] «The CLEF Initiative (Conference and Labs of the Evaluation Forum),» [En línea].

Available: http://www.clef-initiative.eu/. [Último acceso: Septiembre 2012].

[24] PCL (Point Cloud Library), «How to extract NARF Features from a range image,» [En línea]. Available: http://pointclouds.org/documentation/tutorials/narf_feature_extraction.php. [Último acceso: Agosto 2012].

[25] J. Martinez-Gomez y B. Caputo, «Towards Semi-Supervised Learning of Semantic Spatial Concepts for Mobile Robots,» *Journal of Physical Agents,* vol. 4, nº 3, pp. 19-31, 2010.