Proyecto Fin de Carrera

Seguimiento de objetos basado en características y estructura

Autor

Elena Pollo Albéniz

Director

Carlos Orrite Uruñuela

Escuela de Ingeniería y Arquitectura

2016

Seguimiento de objetos basado en características y estructura

Resumen

Este proyecto pretende implementar un algoritmo capaz de realizar el seguimiento de un objeto dentro de una secuencia de vídeo, superando las carencias que presentan algunos de los métodos utilizados en la actualidad.

El procedimiento se basa en tres elementos fundamentales: extractor de características, algoritmo de seguimiento y modelado estructural del objeto. Como extractor de características del objeto a seguir planteamos la utilización de histogramas de gradientes orientados (HOGs) así como descriptores del tipo SIFT (Scale Invariant Feature Transform) o SURF (Speeded-Up Robust Features). El algoritmo de seguimiento contemplado es Mean shift teniendo en cuenta que se realizará un seguimiento en posición, orientación y escala. Finalmente, para hacer una mejor aproximación y corregir las carencias del algoritmo basado en apariencia nombrado, se hace un estudio de los puntos corregidos en cada una de las escalas, mediante un método estructural que se basa en la Triangulación de Delaunay y las coordenadas baricéntricas. Éste otro algoritmo, crea una red de triángulos para cada fotograma y escala, en la que se podrán modificar las posiciones de cada uno de los puntos en base a sus coordenadas baricéntricas respecto a alguno de los triángulos contenidos en la red.

La propuesta se comparará con uno de los algoritmos considerados en el estado del arte, cuya principal limitación es el alto coste computacional que conlleva, consistente en la selección previa de un conjunto de puntos representativos del objeto en el primer fotograma, para los cuales se obtendrán una matriz de características mediante el descriptor de características SIFT, que realiza la búsqueda de descriptores en distintas escalas. El objeto es reconocido en una nueva imagen comparando las características de cada punto del objeto con los almacenados en el primer fotograma y encontrando el candidato más apropiado basándose en la distancia Euclídea y el método RANSAC que elige emparejamientos de puntos y calcula la transformada a la que han sido expuestos para posteriormente aplicarla al resto de puntos característicos de la imagen.

Con todo ello se pretende crear un único método capaz de realizar el seguimiento de un objeto que sufre transformaciones afines, así como oclusiones, minimizando el tiempo.

Agradecimientos

En primer lugar a mi tutor Carlos, por marcarme el camino y tener paciencia en todo el tiempo que ha costado sacar esto adelante.

A mi familia, padres, hermanas, tías, cuñados, sobrinos e incluso Turca. En lo bueno y en lo malo somos una piña, pase lo que pase. Gracias en especial a mi madre, que siempre dirige el barco y es capaz de hacerse cargo de todos y cada uno.

A Mateo, por "ayudarme con mi trabajo", aunque sólo sea sacándome una sonrisa en momentos de agobio.

A mis amigas de siempre, que celebraran conmigo el final tanto como lo haré yo.

A mis compañeros de carrera, en especial aquellos que se han convertido en compañeros de vida. Gracias Paloma por estar siempre ahí, sin tí habría sido más difícil.

A las personas que han aparecido en este nuevo periodo de mi vida pero que parece que se van a quedar por largo tiempo, por apoyarme como si fuéramos amigos de toda la vida y seguir aguantándome a pesar de haber mostrado tan pronto mis momentos malos.

A ti abuela, que estas presente siempre aunque pase el tiempo.

Índice general

Índice de figuras

Parte I

Memoria

Capítulo 1

Introducción

1.1. Antecedentes

La combinación de nuevas tecnologías, como la Visión por Computador, la Realidad Aumentada o la generación de imágenes 3D puede permitir la reconstrucción virtual de elementos desaparecidos o deteriorados. En el proyecto TAA (Tecnologías Audiovisuales Avanzadas), llevado a cabo por el I3A se han propuesto diversos demostradores de Realidad Aumentada, como es el caso de la aplicación desarrollada para el Festival Asalto, patrocinado por el Ayuntamiento de Zaragoza. La App añade contenido virtual a los grafitis. Los ciudadanos que acudieron al Solar "Circo Social" de la Calle Las Armas 83, de Zaragoza, pudieron usar la aplicación, enfocar cada uno de los cuatro grafitis, y en ese momento observar un contenido superpuesto, que proporciona una nueva manera de ver el grafiti y de interactuar con él.

Figura 1.1: Grafitis con realidad aumentada.

Para hacer posible esta aplicación se debe determinar cuándo se está enfocando al grafiti en cuestión y seguirlo de forma fiable cuando el sujeto se mueve explorando las distintas partes del mismo. Para ello se utilizó una herramienta comercial denominada Vuforia.

Vuforia es un Kit de Desarrollo Software (SDK) de Realidad Aumentada para dispositivos móviles que permite la creación de aplicaciones de

Realidad Aumentada. Utiliza tecnología de Visión por Computador para reconocer y seguir imágenes planas (Image Targets) y objetos 3D simples, como cajas, en tiempo real. Esta capacidad de registro de imágenes permite a los desarrolladores posicionar y orientar objetos virtuales, como los modelos 3D y otros medios, en relación con las imágenes del mundo real cuando éstas son vistas a través de una cámara de un dispositivo móvil. El objeto virtual rastrea la posición y orientación de la imagen en tiempo real de tal manera que la perspectiva del observador en el objeto corresponde con su perspectiva en la *Image Target*, de esta manera parece que el objeto virtual pertenece a la escena en el mundo real.

El presente proyecto trata de desarrollar una herramienta propia para la detección automática de elementos en la escena y el seguimiento de dichos elementos a lo largo del tiempo.

1.2. Estado del arte

Para identificar un objeto dentro de un escenario, así como para poder detectar el movimiento que ha sufrido el mismo, usaremos características visuales. En la literatura, las características visuales hacen referencia a ciertos elementos de la imagen con dos componentes: detector, que hace referencia a un punto en la imagen, y descriptor, que se refiere a la información extraída de la vecindad del punto. Tanto los detectores como los descriptores deben ser invariantes a escala, posición, luminosidad y rotación.

En 1999, David Lowe publicó por primera vez el detector y descriptor de características SIFT *(Scale Invariant Features Transform)* [4] . Este algoritmo realiza correspondencia entre puntos basada en los vectores de características de cada punto que componen el descriptor de la imagen y es capaz de detectar puntos característicos estables en una imagen. El detector propuesto por Lowe realiza una localización multiescala mediante un filtrado de diferencia de Gaussianas (DoG), que es una aproximación de una laplaciana de gaussianas, filtro definido por Mikolajczyk y Schmid. Este descriptor construye una pirámide de imágenes donde en cada nivel se aplica un filtrado más grueso para obtener invarianza con respecto a los cambios de escala. El descriptor que se obtiene finalmente consiste en un vector de características de 128 posiciones por cada punto de interés, que codifica valores del gradiente de la imagen en el entorno del punto de interés.

El algoritmo SIFT ofrece buenos resultados, pero no es competitivo en cuanto a velocidad y tiempo computacional. Por esto, en 2006, Herbert Bay, basándose en SIFT, desarrolla el descriptor SURF *(Speeded Up Robust Fea-*

tures) [3], que mejorará en velocidad y eficacia a su predecesor. Para la detección de puntos, SURF hace uso de la matriz Hessiana, concretamente de su determinante, con ello calcula tanto la posición como la escala de los puntos. Hace uso de filtros de tipo caja e imágenes integrales para que éstos puedan ser aplicados a cualquier tamaño a la misma velocidad sobre la imagen original. Es por esto que presenta un mejor rendimiento computacional que el algoritmo SIFT.

Otro detector y descriptor conocido es FAST, que obtiene tiempos mucho más competitivos a costa de obtener peores resultados. También encontramos alternativas como el detector MSER*(Maximally Stable Extremal Regions)*, que aunque no realiza la parte de descriptor, pude sacar información para la descripción. La novedad de este detector es que no se basa en procesar puntos, sino regiones. Como alternativa de descriptores destaca BRIEF *(Binary Robust Independent Elementary Features)*, únicamente descriptor que consiste en un string binario de 128 o 256, por lo que la memoria necesaria para almacenarlo es mucho menor que en los anteriores.

Actualmente, debido al equilibrio entre velocidad computacional y resultados, SURF, es utilizado como detector y descriptor estándar dentro de la familia de descriptores basados en características locales.

Sin embargo, también existen descriptores de tipo global. Ejemplo de ello son los descriptores HOG *(Histograms of Oriented Gradients)*, presentado por Navnnet Dalal y Bill Triggs en 2005. A diferencia de los descriptores anteriormente nombrados, este descriptor trabaja sobre una red de celdas uniformemente espaciadas y usa normalización de superposición local de contraste para mejorar la precisión. El descriptor HOG se basa en que la apariencia y forma local de un objeto dentro de una imagen puede ser descrita por la distribución de sus gradientes de intensidad o dirección de los bordes. La imagen es dividida en pequeñas regiones llamadas *celdas*, y se obtiene un histograma para cada una de éstas. Las celdas adyacentes, son englobadas en regiones mayores llamadas *blobs*, que servirán para hacer la normalización conjunta de las celdas dentro de un blob y finalmente estos histogramas normalizados forman el descriptor.

Los descriptores nombrados anteriormente son descriptores basados en características de apariencia, sin embargo, aunque normalmente son capaces de encontrar correspondencias entre puntos de manera eficiente, no son lo suficientemente robustos ante situaciones como oclusiones, ruido o distractores. Por este motivo, Nicole M. Artner y Walter G. Kropatsch, publican en 2013 un método que combina descriptores de apariencia con descriptores estructurales [1]. Tras hacer un tracking de los puntos, basándose en sus características de apariencia, hacen uso de las características estructurales,

como puede ser la forma de los bordes que unen los puntos de una malla de triángulos o las coordenadas baricéntricas proporcionadas por los triángulos de esa malla. En su estudio hacen una comparación de ambos criterios estructurales, concluyendo que el uso de triángulos con sus coordenadas baricéntricas es más robusto que el uso de la forma de los bordes de éstos. Para elegir el triángulo más adecuado para calcular las coordenadas baricéntricas de los puntos en base a este, hace uso de el coeficiente *confidence* de los triángulos. Este consiste en una ponderación entre la nota obtenida por el tracker, en su caso hacen uso de *Mean Shift*, y la deformación que han sufrido los triángulos tras escoger como vértices los puntos obtenidos por dicho el primero. Escogen como referencia el triángulo que obtiene un valor del coeficiente mayor y corrigen los puntos mediante las coordenadas baricéntricas de los puntos respecto a dicho triángulo.

1.3. Objetivos

Tras exponer los descriptores más utilizados actualmente en el campo de visión por computador, pretendemos hacer un algoritmo que supere las carencias que presentan estos, capaz de realizar el seguimiento de un objeto a lo largo de un vídeo de manera robusta y con un buen rendimiento.

Se busca un algoritmo altamente fiable, es decir, que sea robusto ante diferentes situaciones como pueden ser oclusiones, ruidos o distractores, elementos vecinos muy similares en apariencia a los puntos que buscamos. Para ello, basándonos en algoritmos ya implementados, plantearemos mejoras de éstos para aumentar la tasa de detección y reducir el coste computacional.

El primer objetivo es crear un detector y descriptor de características que presente mayor eficiencia, reduciendo los tiempos utilizados y favoreciendo así la reducción de tiempos de búsqueda de los puntos posterior. Este descriptor también debe ser robusto a cambios de escala, rotación, traslación, oclusiones y transformaciones afines.

Para poder optimizar el descriptor, antes implementaremos una solución para encontrar, dentro de una imagen, los puntos que serán más favorables para hacer su seguimiento, de esta manera se pretende incrementar el rendimiento del algoritmo.

A diferencia de los descriptores anteriormente nombrados, se pretende hacer un método no que sólo detecte los puntos, sino que haga un seguimiento de estos. Tras mejorar la implementación del descriptor, es necesario hacer un buen seguimiento de los puntos obtenidos, por ello, se pretende hacer un método capaz de encontrar un mismo punto entre fotogramas con

la mayor fiabilidad, pese a encontrarse con las posibles distracciones comentadas.

Todos estos conceptos se pretenden englobar dentro de un único método, que englobe detección de puntos óptimos, descriptor de características de estos puntos, algoritmo de seguimiento de estos puntos y combinación con el modelado estructural, para conseguir finalmente un método de detección y seguimiento de un objeto dentro de una secuencia de vídeo.

Con todo ello se pretende crear un único método capaz de realizar el seguimiento de un objeto plano que sufre transformaciones afines, así como oclusiones, minimizando el tiempo de búsqueda de los puntos representativos de dicho objeto.

Capítulo 2

Algoritmo de seguimiento basado en emparejamiento de patrones

Un patrón consiste en la recopilación de las características más significativas de un objeto para después poder ser reconocido dicho objeto en base a éstas. Ejemplo de ello en la vida diaria, son las huellas dactilares, cada huella dactilar consta de crestas, líneas o deltas que son únicos en cada persona. Estos identificadores se unen para crear patrones, de tal manera que una persona podrá ser identificada si el patrón de la huella dactilar coincide con la suya propia.

Como ya se ha nombrado en el Estado del arte (Sección 1.2) existen distintos detectores de características. Estos detectores, con capaces de identificar objetos a partir de sus patrones correspondientes. En el estudio nos centraremos principalmente en dos para comparar sus rendimientos en el caso de detección de un objeto dentro de un vídeo, y posteriormente hacer una selección para adaptarlo a nuestro método.

Un extractor de características es un algoritmo que obtiene, para cada punto, una descripción de sus características, es decir, un vector que contiene información acerca de las regiones alrededor de dicho punto. A estos vectores se les conoce como *features*.

Puesto que queremos detectar un objeto dentro de una imagen, nos centraremos en los *descriptores locales*, que son aquellos que dan información sobre regiones concretas de una imagen, en lugar de tomar la imagen es su totalidad como hacen los*descriptores globales*.

A partir de la descripción del objeto mediante sus características obte-

nidas por el extractor de características podremos buscar el mismo patrón en los frames siguientes, es por esto, que dicho extractor debe ser estable, es decir, invariante a luminosidad, transformaciones geométricas, ruido o distractores.

Nos centraremos en el comportamiento del método **SURF (Speeded-Up Robust Features)** y el método **SIFT (Scale Invariant Features Transform)**, ya que son los descriptores más extendidos. Los fundamentos de ambos se explican de forma más extensa en A y B. Además, será necesario hacer un emparejamiento de puntos característicos entre frames, para determinar el grado de similitud entre ellas. Tanto el algoritmo SIFT como el SURF hacen una correspondencia inicial de características, buscando aquél descriptor en la otro imagen cuya distancia con este sea menor. Pero este emparejamiento no es perfecto, se obtienen varios espurios, puntos con mala correspondencia, que distorsionan los resultados; para mejorar el sistema y filtrar aquellos puntos mal emparejados, se aplica el algoritmo **RANSAC** [Anexo D].

2.1. SIFT vs. SURF

El descriptor SURF, al igual que su predecesor SIFT, obtiene puntos de interés invariantes en escala, orientación e iluminación. Sin embargo, hay diferencias notables entre ellos por las que, según los autores del descriptor SURF, supera a su antecesor en rendimiento y fiabilidad.

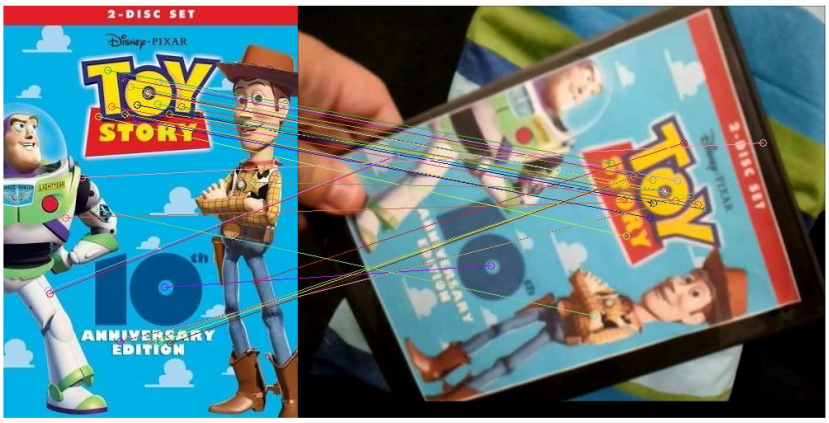
Las principales diferencias entre ambos son:

- La longitud del descriptor SURF es de 64, mientras que el de SIFT es de 128. Esto hace que SURF sea computacionalmente más rápido.
- El descriptor SURF utiliza siempre la misma imagen, la original. SIFT en cambio, necesita escalarla varias veces.
- Para obtener la escala, SURF utiliza el determinante de la matriz Hessiana para calcular tanto la posición, como la escala de los puntos de interés.
- Mientras que en SIFT se guarda la posición, la escala y la orientación, en SURF, en una misma posición solamente aparece un único punto de interés, por lo que no guarda ni la escala y ni la orientación.

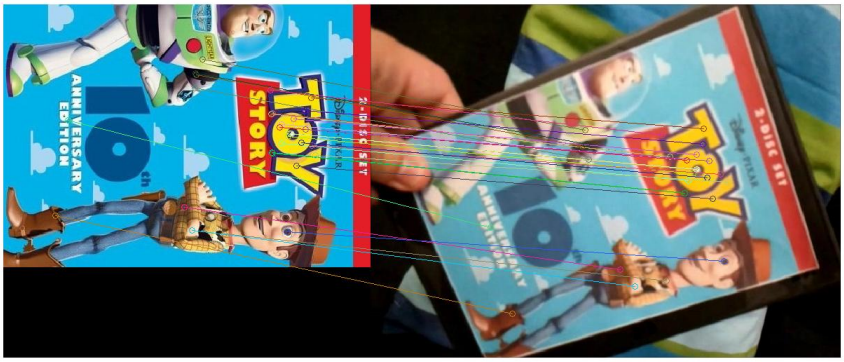
En el próximo análisis veremos si esto también se cumple para las situaciones a las que hará frente nuestro estudio.

Para el descriptor SIFT se hace una normalización de los histogramas para evitar espúreos fuera del objeto, y se hace la comparación entre histogramas mediante la distancia de Bhattcharyya (3.1). En cambio, el descriptor SURF, obtiene los descriptores de características normalizados entre [-1,1], de modo que se trabaja con esa normalización y se calcula la similitud entre histogramsa mediante la distancia Euclídea.

Además, el algoritmo SURF no es capaz de hacer una buena detección cuando el objeto ha sufrido una rotación o traslación grandes. Este problema nos lo hemos encontrado en la inicialización de la búsqueda mediante este descriptor, como se observa en 2.1, no es capaz de hacer un buen emparejamiento en caso que el tamaño y la rotación del template varíen mucho frente al del objeto dentro de la escena.



(a) Rotación abrupta



(b) Rotación similar

Figura 2.1: Detección objeto mediante descriptor SURF en situación de rotación.

Para poder analizar el comportamiento SURF a lo largo e la secuencia, se adecúa la inicialización a las carencias descritas, aunque será una factor a tener en cuenta.

En todas las situaciones analizadas se hace uso de los descriptores acotando la zona de búsqueda de puntos durante el emparejamiento, a la superficie que ocupa el objeto que queremos seguir, de tres alternativas diferentes, comparando siempre con el modelo del objeto, emparejando aquellos puntos que se emparejaron con los del template, acotando la zona delimitada por las esquinas estimadas del objeto. Puesto que se hace una mejora del emparejamiento mediante RANSAC, y este es computacionalemnte lento, se hace

una selección de los 40 puntos con mejor nota tras el emparejamiento inicial de alguno de los descriptores.

Para poder cuantificar el error de los resultados, se conoce la posición exacta de las esquinas del objeto que se está siguiendo en cada frame, por lo tanto, para cada fotograma calcularemos el error mediante la *distancia Euclídea* de las posiciones de las esquinas obtenidas por el método y la posición real de éstas. Además, la importancia de ese error varía en función del tamaño del objeto que se está detectando, por tanto, para comparar errores entre distintas imágenes, es necesario hacer una normalización de la distancia obtenida, que calcularemos a partir del perímetro total del objeto a seguir.

Durante el análisis realizado, se exponen los descriptores a las siguientes situaciones: búsqueda de puntos entre frames consecutivos y búsqueda de puntos con respecto al template, búsqueda de puntos con cambio de escala del objeto, búsqueda de puntos de un objeto sin oclusiones y búsqueda de puntos de un objeto con oclusiones.

Las oclusiones para las que se ha realizado el estudio, consisten en oclusiones sintéticas que cubre el primer cuadrante de la escena, con una región del fondo de la propia imagen, para exponer al algoritmo, a su vez, a situaciones donde los puntos vecinos presentan una apariencia similar.

Prueba 1. Búsqueda de puntos entre frames consecutivos sólo de los puntos dentro de la superficie del objeto. La búsqueda entre frames consecutivos se realiza de la siguiente forma: se parte del modelo del *template* del objeto que se quiere buscar, se obteienen los descriptores de éste y se emparejan con los puntos característicos más similares en el siguiente frame, se hace un filtrado de estos puntos mediante RANSAC y se obtiene la homografía de los puntos obtenidos para estimar la posición de las esquinas del objeto. A partir de aquí, sólo se hace la búsqueda de los puntos que se han situado en la superficie del objeto.

El problema que presenta esta solución para ambos descriptores es que el número de puntos característicos converge, ya que tanto RANSAC, como el emparejamiento previo de los descriptores hace una selección en la cual sólo se pueden descartar puntos. Para ninguno de los dos descriptores es posible terminar la secuencia como se observa en 2.2, ya que el número de puntos que se siguen converge. Sin embargo, el descriptor SIFT presenta un mejor comportamiento y es capaz de realizar el seguimiento en un período de frames más amplio.

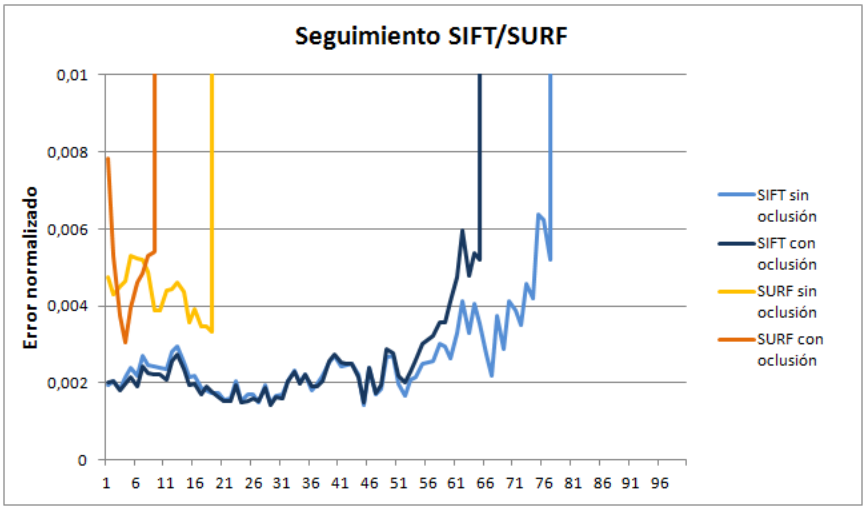


Figura 2.2: Error cometido en el seguimiento entre frames para SIFT y SURF.

El número de puntos que consigue emparejar entre cada pareja de frames consecutivos desciende a medida que avanza la secuencia como se observa en 2.3. Además el número de puntos que identifica desde el primer fotograma, es menor en el caso de oclusiones, y por ello converge antes que el caso en el que no existe oclusión.

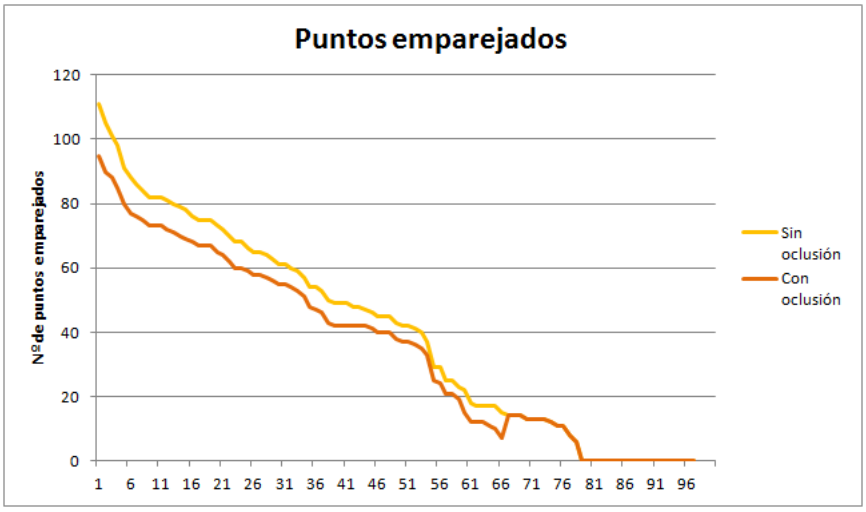
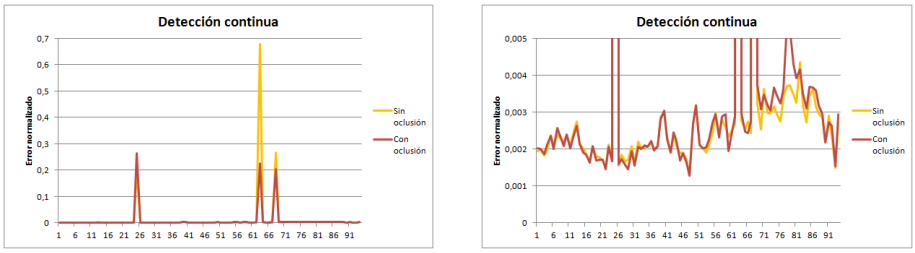


Figura 2.3: Puntos emparejados para cada par de frames consecutivos en el descriptor SIFT.

Puesto que el algoritmo SIFT presenta mejores resultados y es más robusto, las siguientes pruebas se harán en base a éste únicamente.

Prueba 2. Búsqueda de puntos entre frames con respecto al template. Detección continua. En el siguiente análisis, se pone a prueba la robustez del descriptor SIFT frente a la *detección continua*, es decir, en vez de realizar el seguimiento del objeto entre frames, donde existe una continuidad de movimiento y los cambios sufridos por el objeto se entienden que no son muy abruptos, se realizará la búsqueda del modelo aislado del objeto en cada uno de los frames. Esto nos dará una idea de la fiabilidad del descriptor frente a las distintas transformaciones que sufre el objeto.



(a) Error detección continua. (b) Zoom error detección continua.

Figura 2.4: Detección del template del objeto para cada frame.

Según los datos recogidos en 2.4, los resultados obtenidos son muy buenos a excepción de un par de picos puntuales, para el caso en el que no existe oclusión del objeto. Sin embargo, para el caso en el que existe oclusión los resultados son parecidos desde el comienzo de la secuencia, pero el algoritmo no es capaz de terminar la secuencia, probablemente en un fotograma donde el objeto está mayoritariamente ocluido, y el número de emparejamientos que realiza no es suficiente para determinar la homografía. Además, esta solución emplea mayor tiempo computacional que la anterior.

Prueba 3. Búsqueda de puntos entre frames con región de búsqueda acotada. Como alternativa a las anteriores, para poder realizar el seguimiento entre frames sin que converja el número de puntos que se estiman, se propone realizar el seguimiento entre frames que, a pesar de no haber sido emparejados con el frame anterior, se encuentran dentro de la superficie del objeto que se ha determinado mediante la homografía de las esquinas de éste. Para poder llevar esto a cabo, a partir de la estimación de las posiciones de las esquinas, se acota una superficie cerrada que debe corresponder con la superficie del objeto, y se mantienen para la búsqueda en el siguiente frame, aquellos que se encuentren dentro de esta región. Los resultados de esta alternativa se observan en 2.5.

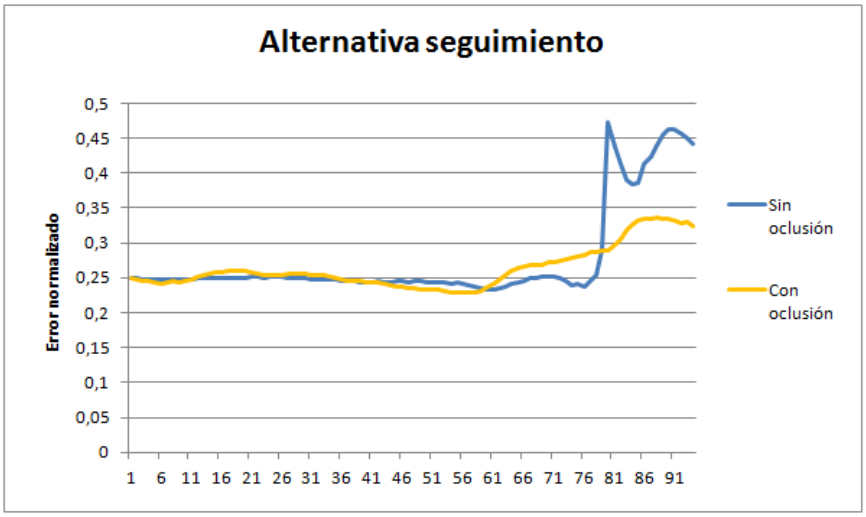


Figura 2.5: Alternativa al seguimiento del objeto entre frames mediante descriptor SIFT.

Esta alternativa no presenta buenos resultados, aunque sí que es capaz de mantener el número de puntos para evitar que el método converja y termine la secuencia. Sin embargo, esta solución arrastra el posible error cometido en fotogramas anteriores, ya que depende de la estimación de las esquinas hecha por el propio algoritmo en iteraciones anteriores.

Como conclusiones obtenemos que SIFT presenta un mejor comportamiento que SURF para los casos analizados, debido a que el primero es más robusto que el segundo en situaciones en las que el objeto presenta grandes cambios, como puede ser rotación o escala. Para todos los casos el comportamiento con oclusiones es peor, llegando incluso a perder el objeto y no ser capaz de realizar el seguimiento durante toda la secuencia. La detección continua del algoritmo SIFT presentaba muy buenos resultados para el caso en que el objeto no sufría oclusiones, aunque el tiempo computacional se incrementaba, sin embargo, esta opción no es válida para el caso con oclusiones ya que no era capaz de realizar el seguimiento completo. La alternativa en la que se acota la región de búsqueda era más robusta a oclusiones pero presentaba un peor comportamiento.

Como alternativa al seguimiento expuesto en este capítulo, optaremos por un seguimiento basado en características y estructura basado en las ideas de rastreo de Mean Shift, cuyos fundamentos se explican en 3.

Capítulo 3

Algoritmo de seguimiento basado en características y estructura

En el capítulo anterior se han expuesto algoritmos como solución al seguimiento de objetos, sin embargo, estos algoritmos no realizan un seguimiento propiamente dicho, sino que realizan una detección del objeto mediante la comparación de puntos.

Para llevar a cabo un seguimiento del objeto en la secuencia de vídeo, en lugar de únicamente detectarlo en cada fotograma, proponemos un algoritmo basado en el *tracker* Mean Shift, que realizará una seguimiento de los puntos característicos entre fotogramas. Además, para evitar las carencias que presentan los descriptores de características a la hora de hacer una comparación entre fotogramas donde existen contrariedades como oclusiones, ruido o puntos vecinos parecidos, combinaremos el algoritmo de seguimiento nombrado con los conceptos de modelado estructural basado en la triangulación de Delaunay. Con esta combinación, se consigue un algoritmo de seguimiento invariante a rotación y traslación debido a Mean Shift, y robusto ante oclusiones o movimientos rápidos debido al modelado estructural.

3.1. Mean Shift

El algoritmo Mean shift, basado en las ideas propuestas por Fukunaga and Hostetler, es un algoritmo *Hill Climbing*, o *subida de la colina*, de búsqueda del máximo local en densidad. Se trata de un método no paramétrico para localizar la función de densidad máxima en datos discretos.

Aplicado a nuestro algoritmo, el método Mean Shift tratará de buscar

el punto con mayor similitud al que estamos buscando, mediante la comparación de la matriz θ, μ descrita en 4.1.1 con el histograma de orientaciones de los gradientes del punto que se está buscando.

Como se explica más adelante, utilizaremos este método para dos situaciones distintas. Por un lado, se utilizará durante la selección de los puntos más favorables para hacer su seguimiento a lo largo de todos los frames (Sección 4.2). En este caso se utilizará Mean Shift para saber cuanto puede alejarse un punto de su posición inicial y poder volver a su punto de partida. Por otro lado, se hará uso del mismo método de seguimiento durante todo el algoritmo para detectar la posición de punto entre frames, partiendo de la posición en que se encuentra en el frame anterior al que se está examinando.

El procedimiento que sigue es el siguiente: se obtiene el histograma de orientaciones del punto a estudiar en el frame anterior, que consiste en un vector de longitud $128 bins$, que recopila la información de las orientaciones de los gradientes de los píxeles vecinos en los 360° alrededor, discretizado en 16 direcciones y 8 para las orientaciones, esto se explica detalladamente en 4.1.1. Según el mismo método se obtiene un mapa de orientaciones de todos los píxeles del nuevo frame. Lo que se quiere hacer es encontrar la posición donde se encuentra ese mismo punto en el nuevo fotograma, para ello, se parte de la misma posición en la que se encontraba en el anterior fotograma y a partir de ahí, con una ventana de 3x3$píxel$ alrededor, se comparan los histogramas de cada punto dentro de esta ventana con el histograma del punto que estamos siguiendo. Para saber el grado de similitud entre histogramas de orientaciones de gradientes se hace uso de la **distancia de Bhattacharyya**. Esta distancia mide la semejanza de dos distribuciones de probabilidad discretas y se define como

$$dBhatt(p, q) = \sum \sqrt{p(x)q(x))} \tag{3.1}$$

Siendo p y q las dos distribuciones que se quieren comparar.

Tras obtener las distancias de Bhattacharyya para la ventana de píxeles, se elige como nueva referencia para el centro de vecindad, el punto con la mayor distancia. Se coloca el punto estimado en esta posición y con ello se desplaza la ventana de vecindad en torno a este nuevo punto para repetir el paso anterior. Este proceso se repetirá sucesivamente hasta encontrar el máximo local, es decir, hasta que al comparar el histograma de un punto con los de sus vecinos, sea el punto donde está centrada la ventana el que tiene el máximo. Para evitar que el algoritmo tienda a infinito, se acota

el número de iteraciones a 15 desplazamientos, en caso que dentro de este número no se haya localizado el máximo, se da por perdido el punto que se estaba buscando de inicio. La pérdida del punto se puede deber a que estuviera ocluido o a que le movimiento fuera muy rápido y no local, como supone este algoritmo.

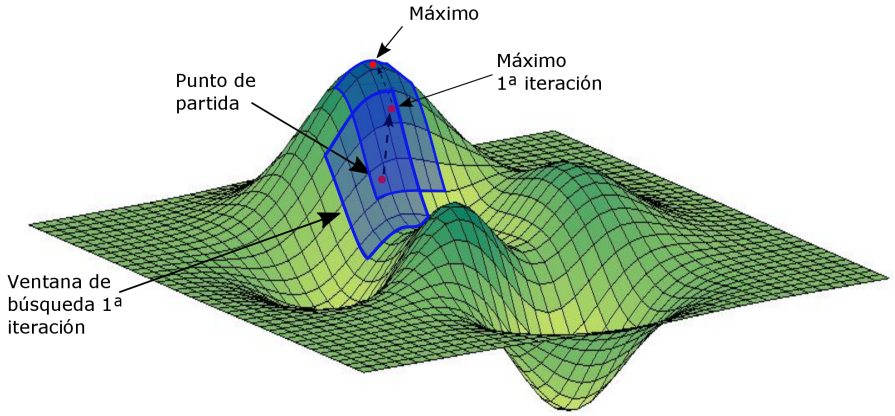


Figura 3.1: Recorrido de un punto hasta encontrar el máximo mediante el método Mean Shift.

El hecho de partir de la misma posición en que se encontraba el punto en el frame anterior y acotar la búsqueda del punto a una región, ahorrará tiempo de búsqueda. Esto es otra diferencia con los descriptores con los que se compara nuestro método, ya que tanto SURF como SIFT, hacen una comparación de cada punto con todos los de la imagen para encontrar el que tiene un descriptor más parecido al que se está buscando, esto incrementa el tiempo de procesado de ambos descriptores.

Por tratarse de un método de búsqueda del máximo local, no hace una buena estimación en caso que el movimiento entre fotogramas sea muy rápido, ya que el nuevo punto se encontrará demasiado lejos y el campo de búsqueda del algoritmo no llegará hasta ahí. Lo mismo pasa para el caso de las oclusiones, no es un buen método, puesto que la comparación de los histogramas no encontrará resultados, por este motivo, para cubrir estas necesidades, se combina el algoritmo de seguimiento expuesto con el modelado estructural que se explica en 3.2. En caso que Mean Shift no sea capaz de encontrar la nueva posición del punto que se está buscando, se localizará este punto mediante las características estructurales de ese punto dentro de un conjunto.

3.2. Modelado estructural

De los descriptores anteriormente descritos, así como del extractor de características implementado en este proyecto, (4.1.1), se obtiene una nube de puntos característicos. Para poder aprovechar la estructura que crean estos puntos y que reflejan la forma del objeto que queremos seguir, es necesario crear bordes que los unan. Un método para obtener el modelo estructural de los puntos es la triangulación, es decir, calcular la malla de triángulos que se forman al unir unos puntos con otros, siendo éstos los vértices de los triángulos. Hemos elegido la **Triangulación de Delaunay** para crear una malla que siga unas normas.

3.2.1. Triangulación de Delaunay

La triangulación de Delaunay consiste en una malla de triángulos donde todos los triángulos que la forman cumplen la condición de Delaunay, según la cual, la circunferencia circunscrita de cada triángulo de la red no debe contener ningún vértice de otro triángulo. Por lo tanto, todos los triángulos de la malla poseen las siguientes propiedades:

1. La frontera de la cara exterior de la triangulación de Delaunay es la frontera de la cubierta convexa del conjunto de puntos.
2. La circunferencia circunscrita de cualquier triángulo en la triangulación de Delaunay está vacía (no contiene ningún sitio de P).
3. Dos sitios pi y pj están conectados por una arista en la triangulación de Delaunay, si y sólo si hay un círculo vacío que pasa por p_i y p_j.
4. Los pares más cercanos de sitios en P son vecinos en la triangulación de Delaunay.

Por estas propiedades se aplicará la triangulación de Delaunay en dos situaciones dentro del método, primero en la elección previa de los puntos a seguir y posteriormente a lo largo de todos los fotogramas para determinar la posición de los puntos vecinos dentro de una malla.

En el primer caso, se hace uso de esta triangulación en el proceso de selección de los puntos que se van a seguir a lo largo de la secuencia de vídeo. Como se explica en la Sección 4.2, se hace un análisis previo de las características de apariencia de todos los píxeles de la imagen inicial, y de

entre todos, se escogen aquellos que presenten unas características visuales favorables para poder hacer una seguimiento y ser identificados en cada *frame*. Sin embargo, antes de poder utilizarlos, es necesario hacer una segunda selección puesto que el número de puntos que superan el primer análisis es elevado y no siempre útil, ya que muchos puntos están demasiado cercanos unos de otros. Por esto, en la segunda parte del análisis de los puntos, se tiene en cuenta las características estructurales que presentan, es decir, como se encuentran unos puntos de otros dentro de la imagen. Se quiere llegar a una situación en que los puntos seleccionados, además de ser favorables para hacer un buen seguimiento debido a sus características visuales, creen una malla uniforme, que abarque la mayor superficie del objeto y que no se encuentren unos demasiado cerca de los otros, puesto que la información sería redundante. La selección estructural se hace aplicando la Multi-escala de Delaunay que hemos desarrollado en 3.2.1, donde se crean iterativamente triangulaciones de Delaunay, que dan lugar a mallas de triángulos en la mayor medida regulares, y se van descartando aquellos nodos que corrompan esta regularidad, hasta encontrar una malla de triángulos uniforme que abarca la mayor parte de la superficie del objeto que se está siguiendo y con un número reducido de puntos, en nuestro caso hemos elegido 40 puntos.

En el segundo caso, se hace uso de la triangulación de Delaunay a lo largo de toda la secuencia de fotogramas para hacer una estimación del lugar donde se encuentran los puntos que estamos buscando en el nuevo *frame*. Como ya hemos dicho, la malla de triángulos que crea la triangulación de Delaunay nos proporciona información acerca de la estructura que forman los puntos en conjunto, esta estructura mantiene sus propiedades a lo largo de toda la secuencia siendo invariante a rotación, traslación y escalado, puesto que si el objeto sufre estas transformaciones, todos los puntos las sufrirán al mismo tiempo y la estructura de triángulos presentará una apariencia similar adaptada a las proporciones de las transformaciones. Para poder aprovechar estas propiedades se hace uso de las coordenadas baricéntricas, cuyos fundamentos se explican en la Sección 3.2.2, y que crean un sistema de coordenadas de puntos tomando como referencia un triángulo, por tanto, es posible saber la posición exacta de un punto si se conoce la posición del triángulo de referencia. Este sistema de coordenadas se mantendrá invariante a lo largo de toda la secuencia, por eso es necesario calcularlo únicamente en el primer fotograma, en el resto de iteraciones sólo será necesario hacer el paso inverso, es decir, obtener las coordenadas cartesianas a partir de las baricéntricas.

Puesto que las coordenadas baricéntricas se basan en el triángulo de referencia, será importante la fiabilidad del triángulo que se escoja para ello, di el triángulo que se elige no está correctamente situado en la imagen, todos los puntos serán desplazados en la misma proporción. Por ello, en cada fotograma se hace una selección del triángulo más favorable en base a los

puntos que forman sus vértices. Para conseguir esta medida, se hace uso del factor *confidence*, introducido por Nicole M. Artner y Walter G. Kropatsch en [1]

El factor *confidence* combina dos propiedades de los triángulos: cambio de forma y similitud en apariencia respecto a los calculados en el primer fotograma. Con esto, se busca aquel triángulo cuyos vértices ofrezcan la mejor similitud de sus descriptores de apariencia y, el triángulo haya sufrido la menos distorsión posible. El cálculo de este factor se hacer por tanto:

$$F(f,t_i) = \frac{1 - R(f,t_i) + min(A(a(v_j), I(p(v_j,t_i))))}{2} \tag{3.2}$$

donde A es la similitud en apariencia de sus vértices y $R(f,t_i)$ es el cambio de los ratios del triángulo, y se calcula como:

$$R(f,t_i) = \text{mín}(|1 - \frac{r_{12}(t_i)}{r_{12}(t_0)}| + |1 - \frac{r_{13}(t_i)}{r_{13}(t_0)}| + |1 - \frac{r_{23}(t_i)}{r_{23}(t_0)}|, 1) \tag{3.3}$$

En el primer fotograma se alamacenan las coordenadas baricéntricas de todos los puntos respecto a todos los triángulos de la malla, posteriormente, en cada fotograma de la secuencia, se escogerá aquel triángulo que presente un factor *confidence* mayor y se utilizarán las coordenadas baricéntricas respecto a este triángulo obtenidas en el primer fotograma.

3.2.2. Coordenadas Baricéntricas

Las coordenadas baricéntricas nos permiten conocer la posición de un punto respecto a un triángulo. Se definen como:

Consideremos una colección de puntos P_1, P_2,...P_n y tomemos la colección de puntos P que se pueden escribir en la forma:

$$P = \alpha_1 P_1 + \alpha_2 P_2 + ... + \alpha_n P_n \tag{3.4}$$

donde:

$$\alpha_1 + \alpha_2 + ... + \alpha_n = 1 \tag{3.5}$$

Los puntos P forman un espacio, y las coordenadas $(\alpha_1, \alpha_2, ..., \alpha_n)$ son llamadas las ***coordenadas baricéntricas*** *de los puntos del espacio*

Por tanto, cualquier punto puede ser descrito mediante las coordenadas baricéntricas. Además, estas coordenadas cumplen que son constantes a lo largo de la secuencia de vídeo. Esto quiere decir, que son invariantes a escala y rotación.

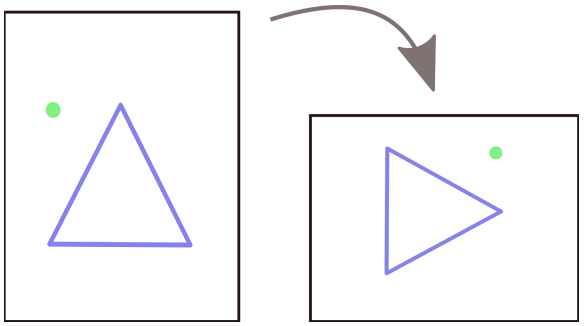


Figura 3.2: Las coordenadas baricéntricas de un punto respecto a un triángulo son invariantes a rotación y escala.

Para poder hacer uso de esta propiedad, calculamos las coordenadas baricéntricas de todos los puntos respecto a los triángulos seleccionados en el *frame* inicial, y en los *frames* sucesivos hacemos el paso inverso, es decir, calculamos las coordenadas cartesianas a partir de las baricéntricas, mediante alguno de los triángulos que forman la malla de Delaunay. De este modo, si el triángulo ha cambiado de tamaño, o ha rotado, las coordenadas cartesianas obtenidas también lo habrán hecho, puesto que se calculan respecto al triángulo de referencia, en cambio las coordenadas baricéntricas respecto al triángulo se mantendrán invariantes.

3.2.3. Multi-escala Delanuay triangulación basado en áreas

Se trata de un algoritmo en el que se descartan de manera iterativa puntos de una malla de triángulos en función de las propiedades estructurales con sus vecinos. El objetivo de este algoritmo es simplificar una malla de triángulos relativamente grande, la cual, pese a poseer gran cantidad de información en sus triángulos, no siempre nos permitirá conocer la estructura global del objeto de manera correcta.

Mediante el uso de coordenadas baricéntricas para cualquier punto respecto a cualquier triángulo de una malla de triángulos, se puede obtener un "structural cue"de dicho punto. Sin embargo, si el triángulo no es lo suficientemente grande, comparado con el entorno, o el punto del que se quiere obtener la información se encuentra lejos de éste, la exactitud de las coordenadas baricéntricas varía. Es por esto, que es recomendable que los triángulos de referencia sean del mayor tamaño posible y abarquen la mayor parte posible del objeto en cuestión y, por tanto, de los puntos a seguir.

Siguiendo esta idea se crea este algoritmo iterativo que hace una selección de puntos dentro de un mallado, en busca de aquellos que creen los

triángulos más adecuados para, posteriormente, usar como referencia en el cálculo de las coordenadas baricéntricas de los puntos a seguir en la imagen.

En primer lugar, para el conjunto de puntos seleccionado previamente mediante descriptores de apariencia, se crea una triangulación de Delaunay, que consiste en una malla de triángulos que cumplen la condición de Delaunay. Esta condición dice que la circunferencia circunscrita de cada triángulo de la red no debe contener ningún vértice de otro triángulo. Por tanto, aquellos puntos que no sean capaz de cumplir dicha condición quedan descartados.

De entre todos los triángulos que forman la malla se escoge el triángulo con menor área. Se hace un cálculo de cuánto ocupan, dentro de la malla, los dos segmentos a los que están unidos cada uno de los nodos del triángulo de menor área. Se entenderá que el nodo que menos contribuye es aquel cuya longitud de sus dos segmentos es menor y, por tanto, será este el nodo descartado.

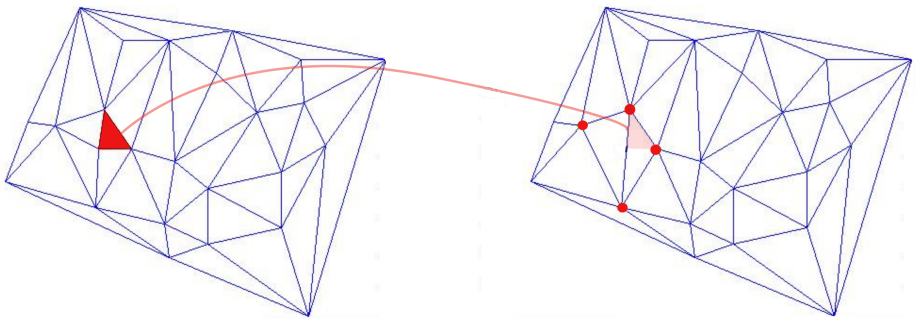


Figura 3.3: a) Triángulo a descartar por ser el de menor área. b)Nodos afectados tras haber eliminado el triángulo anterior

A continuación se generará una nueva triangulación de Delaunay, sin contar con el nodo descartado y se examinarán los nodos que se han visto afectados por la ausencia de éste. Se entiende por *nodo afectado* aquellos que compartían segmento con el nodo descartado y por tanto ya no pertenecen a un triángulo que antes existía y ya no, así como aquellos que ahora pertenecen a un triángulo nuevo que no existía en la etapa anterior.

Los pasos anteriores se repiten hasta que todos los nodos que no han sido descartados se consideran nodos afectados por alguna de las razones comentadas en el punto anterior.

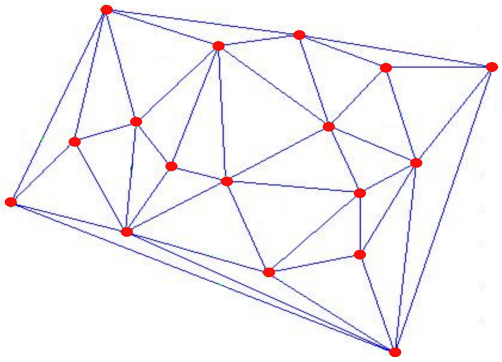
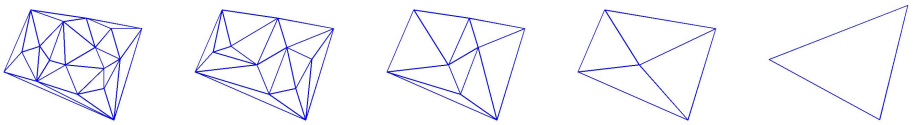


Figura 3.4: Fin de la escala. Todos los nodos son nodos afectados

Cuando todos los nodos que no han sido descartados cumplen alguna de las normas para convertirse en nodos afectados, se considera que se ha creado la primera escala de triángulos de Delaunay y se guarda para funciones posteriores. A partir de los puntos de esta nueva escala se reiniciará el proceso.

Se considera el proceso terminado cuando se obtiene la última escala de triángulos, esto es, la malla resultante se compone de un único triángulo. Esto está garantizado pues el método siempre converge ya que no se generan nuevos nodos, éstos sólo se pueden descartar. Sin embargo, esto da lugar a que el número de triángulos cambie.



(a) Escala 1 (b) Escala 2 (c) Escala 3 (d) Escala 4 (e) Escala 5

Figura 3.5: Multi-escala Delaunay

El número de escalas que se obtiene depende del conjunto de nodos que forman la malla inicial de triángulos.

Capítulo 4

Nuestra propuesta

En esta memoria se ha explicado el concepto de seguimiento de un objeto basado en emparejamiento de patrones, sin embargo, hemos comentado que esto no consiste en un seguimiento propiamente dicho, además que presentaba poca fiabilidad antes situaciones de oclusión o rotación notable. Como alternativa, proponemos un algoritmo de seguimiento que combina Mean Shift con modelado estructural para realizar un seguimiento y que además sea robusto ante oclusiones. Además, el método debe presentar invarianza a rotación, por ello, previo al seguimiento, proponemos un descriptor que sea capaz de presentar mayor fiabilidad que SIFT ante esta situación. Este descriptor creará un mapa de orientaciones de los gradientes de los puntos y hará una selección de los puntos característicos óptimos para su seguimiento.

Por tanto, la descripción del método propuesto en conjunto, donde se combinan los conceptos explicados anteriormente, se presenta a continuación.

4.1. Descriptor de características

En el análisis de descriptores de características realizado en 2, se obtiene que ambos métodos estudiados presentan mala fiabilidad ante oclusiones y rendimiento medio en situaciones de un objeto con transformaciones. Además, el descriptor SIFT, que ha obtenido mejores resultados que SURF, puede no presentar inviarianza ante la rotación del objeto, esto se debe a que su histograma de gradientes está normalizado respecto a la orientación dominante, como se explica en A. En caso que exista más de una orientación dominante para un mismo punto, la normalización de su histograma se verá afectado por el ruido, ya que en función del ruido, variará la elección de la orientación elegida como dominante, y con ello, presentar un mal rendimiento ante la rotación del objeto.

Con motivo de evitar esas carencias, y en busca de un descriptor más robusto y eficiente se escoge como descriptor de características un mapa de orientaciones con coordenadas angulares, detallado en 4.1.1.

4.1.1. Matriz θ, μ

Como ya hemos dicho, un feature es un vector o matriz de características de un punto determinado. Como alternativa a los features SIFT o SURF,se propone un mapa de orientaciones que dará invarianza respecto a la rotación.

Un mapa de orientaciones consiste en una matriz de las mismas dimensiones que la imagen principal, donde la tercera dimensión recoge, para cada punto, un vector donde se indican los valores de las orientaciones de los gradientes de los puntos que rodean a éste.

A diferencia de los otros descriptores, para obtener la matriz θ, μ se hace un barrido de los 360° alrededor de un punto, para conseguir las orientaciones de los contornos en esos ángulos.

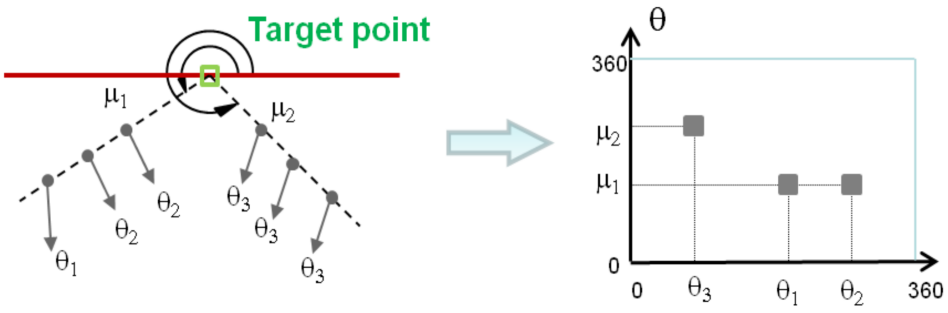


Figura 4.1: Feature de un punto

En la Figura 4.1 se observa como los 360° alrededor del punto, μ, están discretizados en 16, mientras que las direcciones de los gradientes, θ, está discretizado en 8 posibles direcciones.

Para obtener los ángulos discretizados de μ se hace uso de 16 máscaras filtradas que darán robustez al sistema. Estas máscaras son de tamaño 16x16, y cada una de ellas cubre un ángulo μ distinto, teniendo un total de 16 máscaras, ya que aunque se podría cubrir todo el rango con sólo 8 de esas máscaras, existe solapamiento entre ellas para evitar el posible error en los límites de las máscaras. Además, estas máscaras están filtradas por un filtro gaussiano para dar mayor valor a las zonas más cercanas al punto de origen.

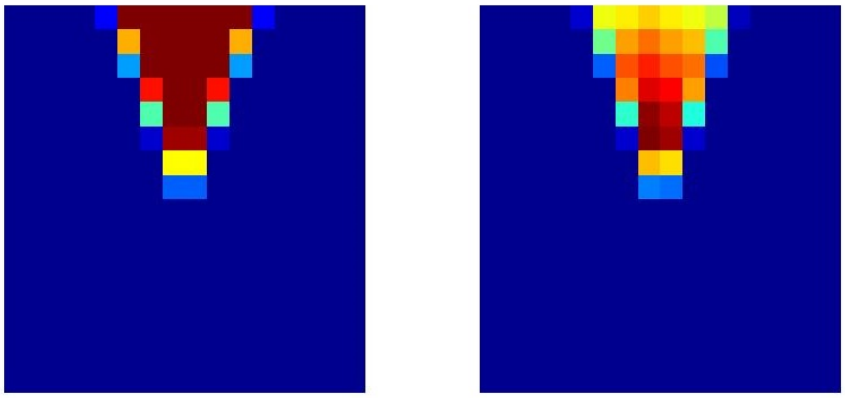


Figura 4.2: Máscaras con un filtro gaussiano

Previo a buscar las orientaciones de los gradientes de los puntos es necesario normalizar la imagen y se filtrarla con dos filtros gaussianos separados en filas y columnas para obtener los bordes.

La novedad de este mapa de orientaciones, respecto a los que utilizan los descriptores anteriores, es que la información que contienen la matriz son coordenadas angulares, es decir, a la hora de hacer la comparación entre matrices para buscar el punto que queremos encontrar, no se hará una búsqueda de matriz similar, sino que se buscará la rotación de los valores de dicha matriz. Aunque la comparación es un poco más complicada, esto reduce la búsqueda considerablemente, ya que se podrá hacer una con tan sólo 16 movimientos, 16 porque la matriz no es simétrica, ya que la dirección μ está discretizada en 16 sectores de 22,5° cada uno, mientas que θ se discretiza en 8 sectores de 45° cada uno. Con esto, se obtiene el ángulo que sufren todos los puntos del histograma, para evitar así el error que presenta SIFT, ya que al normalizar, basa todas las rotaciones en función de la orientación principal, y en caso de fallar ésta debido al ruido, falla el algoritmo ya que no es capaz de identificar dicho histograma.

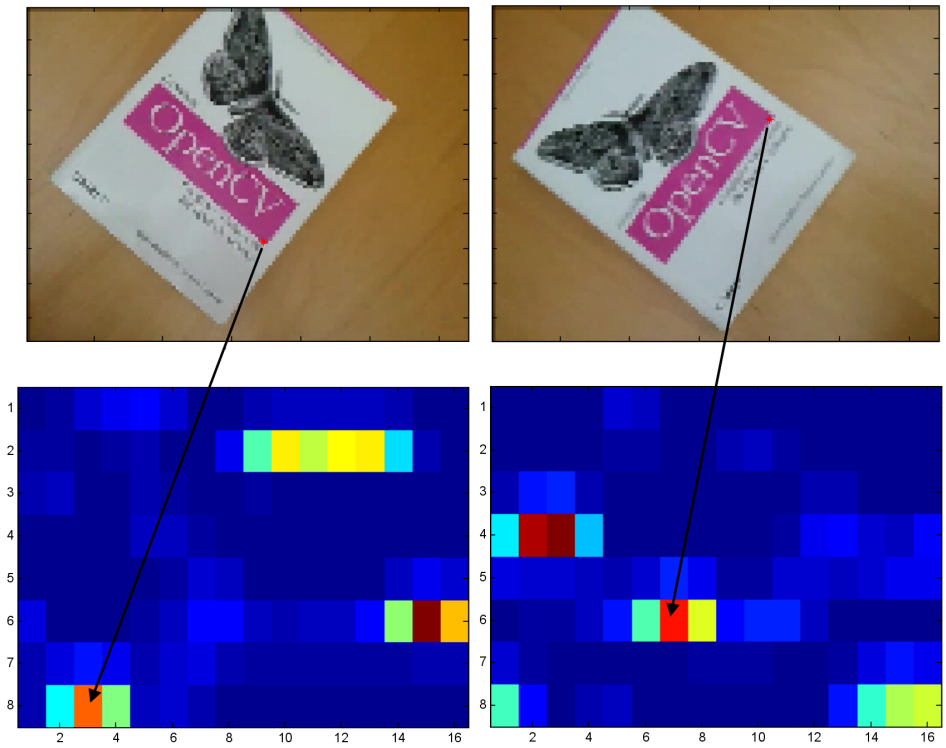


Figura 4.3: Rotación de un punto y su matriz de orientaciones

Este número de movimientos reducido se debe a que en de una imagen, si un punto rota dentro de ésta con cierto ángulo también lo hacen sus orientaciones con un ángulo igual, es por esto que se dice que la búsqueda se hace en *"diagonal"* en lugar de uno a uno. Un ejemplo de esto se puede ver en la Figura 4.3, en él se observa como una imagen es rotada aproximadamente 90° y con ello, el punto que se está siguiendo, y los gradientes de sus vecinos, también lo hacen. Esto se ve reflejado en las matrices de orientaciones de dichos puntos. El eje y de la matriz representa el ángulo θ, que son las orientaciones de los gradientes de sus vecinos, el eje x representa los ángulos discretizados de μ, es decir, el sector alrededor del punto en el que está buscando los gradientes. Tomando como referencia alguno de los valores más altos de gradientes, se observa como el desplazamiento que sufren en μ es de 4 sectores, lo que equivale a 90°, exactamente el desplazamiento que sufren las orientaciones en θ, dos sectores de 45° cada uno. Por tanto es un movimiento diagonal, ya que se desplaza lo mismo en ambas direcciones. Además, también hay que destacar que, puesto que se trata de coordenadas angulares, la rotación de la matriz es circular, es decir, los 360° equivalen a 0°. Se puede observar una representación gráfica de la búsqueda diagonal descrita en la Figura 4.4.

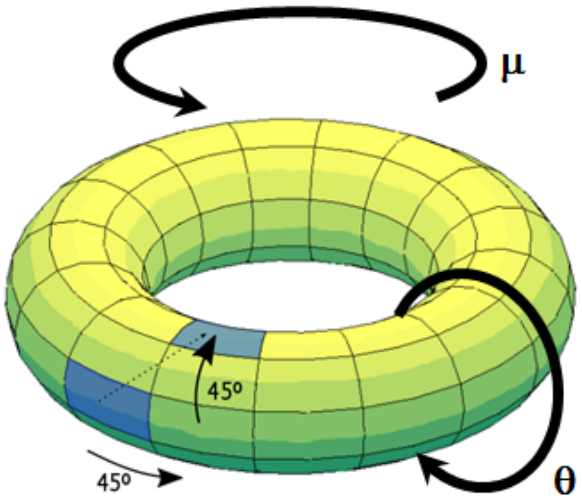


Figura 4.4: Desplazamiento diagonal de 45° sobre el toroide de las coordenadas angulares de la matriz θ, μ

De modo que para crear el mapa de orientaciones de toda la imagen, se hace una búsqueda de las orientaciones de los gradientes mediante el método descrito para cada punto, adaptando las máscaras utilizadas a la escala en la que se encuentra la imagen.

4.2. Elección puntos óptimos

Es importante escoger los puntos más adecuados del objeto a estudiar. Éstos serán la referencia que determinará la posición y forma del objeto que queremos localizar. Por ello, es necesario escoger un punto que nos dé fiabilidad a la hora de hacer su tracking y se pueda identificar con la máxima exactitud.

No todos los puntos son igual de robustos ante el ruido en la imagen, debido, principalmente al entorno en el que se encuentran. Es por esto, que para cada punto de la imagen se observa tanto las características propias del punto, como las de la región en la que se encuentra. Para poder seleccionar los puntos más adecuados y hacer un tracking fiable, haremos un estudio de todos los puntos que forman el primer *frame*, donde se expondrán a tres factores basados en su apariencia.

Previamente a examinar punto por punto, se ha construido un mapa de orientaciones de la imagen completa. En él, se hace un estudio de las orientaciones de los gradientes de cada punto y sus proximidades, del que se obtiene un histograma de gradientes orientados para cada punto de la imagen. El procedimiento para crear el mapa de orientaciones se explica en la Sección 4.1.1. Este mapa nos servirá para poder ver el rendimiento de los puntos frente a los factores que se explican a continuación.

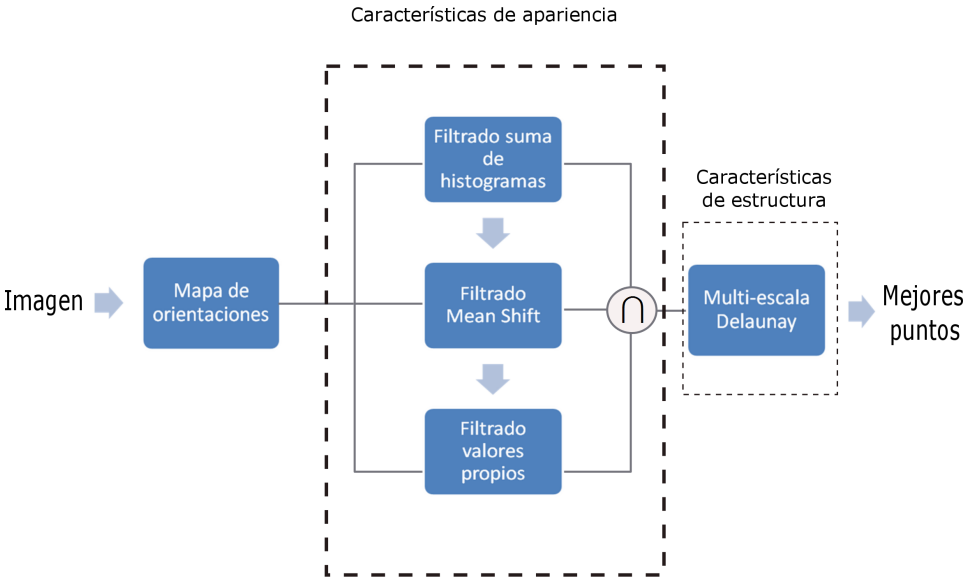


Figura 4.5: Proceso de selección de puntos óptimos.

En primer lugar, el píxel en cuestión no debe pertenecer a una zona de textura uniforme. Es posible que el punto por sí solo dé un valor de gradiente alto y, sin embargo, debido a los píxeles vecinos, sea irrelevante para hacer su tracking, ya que forma parte de una zona uniforme y eso podría hacer que se perdiera difuminado entre su vecinos en los frames posteriores. Además estas texturas son muy sensibles a ruido en la imagen. Para saber si un punto pertenece a una zona de textura uniforme, hacemos uso del **histograma de gradientes orientados**, que obtendremos a partir del mapa de orientaciones, donde cada histograma tiene una longitud de 128*bins*. En caso que la suma de los valores del histograma dé un valor bajo, está indicando que se trata de un punto dentro de una zona uniforme, pues los gradientes de los píxeles de la región de alrededor del punto apuntan mayoritariamente en una dirección y, por tanto, la apariencia es similar entre vecinos, es necesario descartarlo.

En la Figura 4.6 se aprecia la diferencia entre un punto perteneciente a una zona uniforme, a la izquierda, donde destaca una orientación de gradiente sobre las demás, y por tanto la suma de todas da una valor muy bajo, y otra que se encuentra en una zona de textura no uniforme, a la derecha, donde hay valores altos para casi todas las orientaciones posibles.

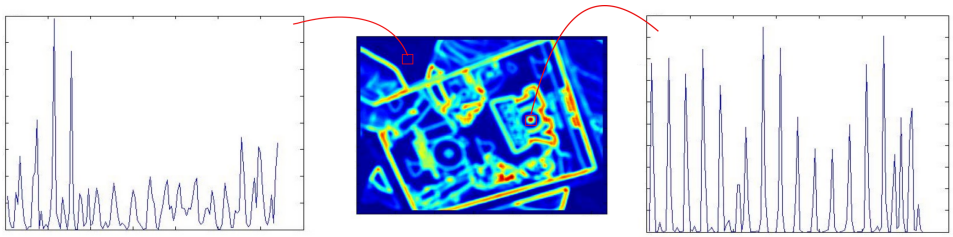
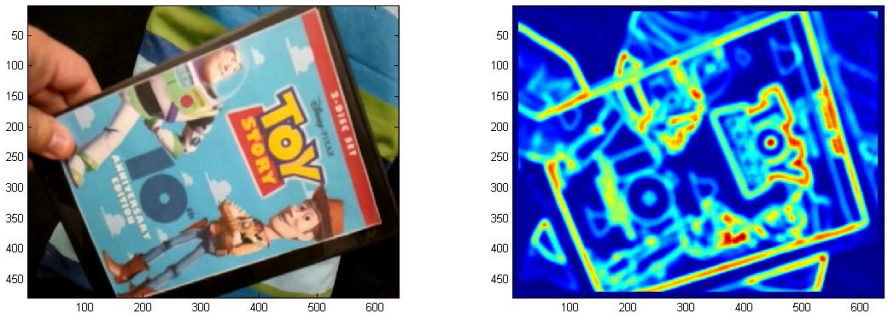


Figura 4.6: Histograma gradientes orientados. Izquierda: textura uniforme, derecha: textura no uniforme.

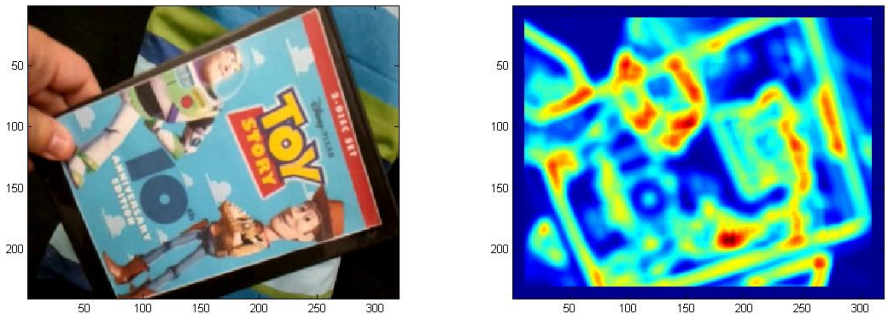
Tras realizar distintas pruebas, que se detallan en 4.2.1, donde se tiene en cuenta la sensibilidad de este factor, junto con los otros dos, se obtiene un valor umbral de 2 para la suma de los historiales.

Una vez que se ha examinado el primer paso, sometemos al punto a una búsqueda del mismo ante un hipotético desplazamiento en sus proximidades mediante el método **Hill Climbing**. Éste método hace una búsqueda iterativa del máximo local. En nuestro caso, mediante la **distancia de Battacharyya** haremos una búsqueda de cuánto se puede alejar el punto de su posición inicial y poder regresar al punto de partida. Este método está detallado en el Capítulo 3.

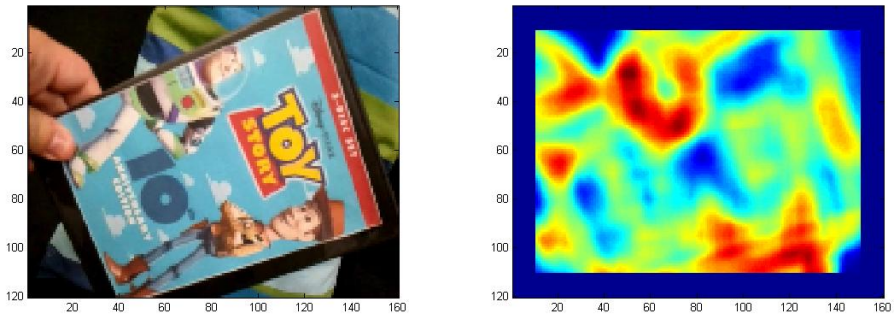
De nuevo, a partir del mapa de orientaciones de los gradientes, se obtiene el histograma del punto a estudiar. Además, se crea un mapa de orientaciones de la región acotada por el Hill Climbing, escogiendo a partir del primero, todas las posibles rotaciones del punto. Después, se compara el histograma de gradientes orientados del punto con cada uno de los histogramas de las rotaciones alrededor y se guardan las distancias de Battacharyya obtenidas en cada caso.



(a) Escala 1



(b) Escala 2



(c) Escala 4

Figura 4.7: Suma histogramas para cada escala.

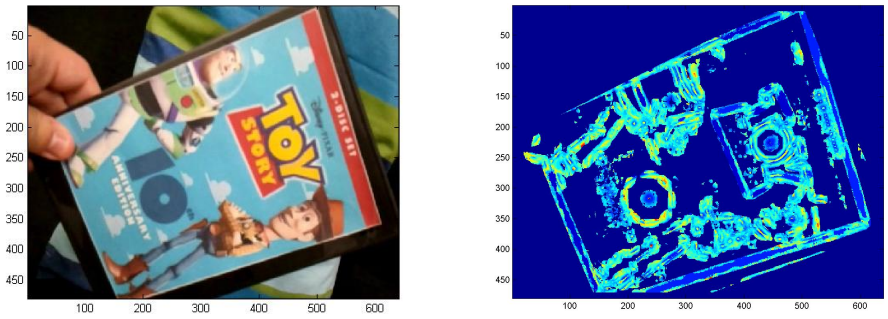
También es posible que dicho punto no converja, para evitar esto, se hace un límite de 11 pasos, si en ese intervalo no ha encontrado el máximo local, se interpreta que el punto se ha perdido.

Finalmente, de entre las mayores notas obtenidas, se asigna al punto la nota mayor más próxima a él.

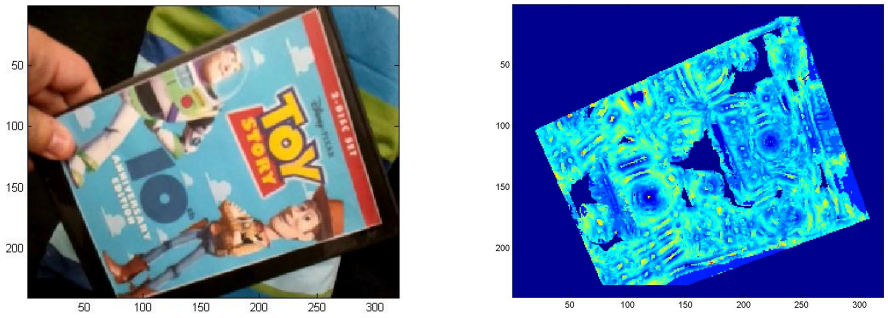
En esta etapa, seleccionamos los puntos que hayan obtenido las notas más altas. El umbral estimado para este factor es de 6 para la escala real, 5 para la escala 2 y 4 para la escala más grosera. Como se han estimado estos valores se explica en 4.2.1. Los puntos que superan este factor como consecuencia del umbral elegido se pueden observar en la Figura 4.8

El segundo factor carece de valor si el punto pertenece a un borde, ya que éste podrá ser fácilmente reconocido si el desplazamiento es fuera del contorno y, sin embargo, si el desplazamiento es a lo largo del mismo, éste podrá ser confundido con sus puntos vecinos, también pertenecientes al mismo borde. Para descartar estas posiciones se hace uso de la convolución de dos funciones gaussianas, una perpendicular a la otra.

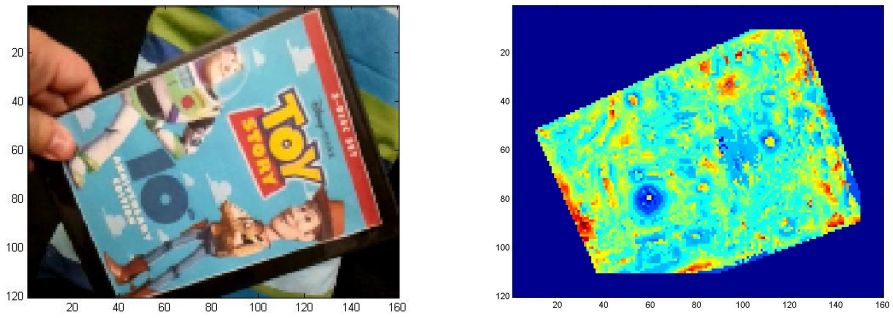
Para llevar a cabo este proceso, se escoge una máscara de 5x5 alrededor del punto y se obtiene la covarianza de las filas por las columnas, de este modo pretendemos saber en qué relación varían los gradientes de los píxeles vecinos separados por filas y columnas. Se obtienen los valores propios, de modo que, si la covarianza obtenida tiene un aspecto circular, es decir, sus valores propios son similares, podemos considerar el punto adecuado. En cambio, si la covarianza tiene un aspecto ovalado, sus valores propios son dispares, significa que se encuentra en un borde y no resulta adecuado. Éste es un principio en el que ya se basa el detector SIFT, sin embargo, a la hora de elegir el umbral, no sólo escogeremos aquél que elimine los puntos con valores propios dispares, sino que pondremos un umbral más restrictivo que elimine también los puntos que sean sensibles al ruido. Tras las pruebas realizadas para la estimación de los umbrales (4.2.1) se obtiene que se trata del factor más crítico de los tres y su valor umbral para conseguir los mejores puntos es de 1.98.



(a) Escala 1

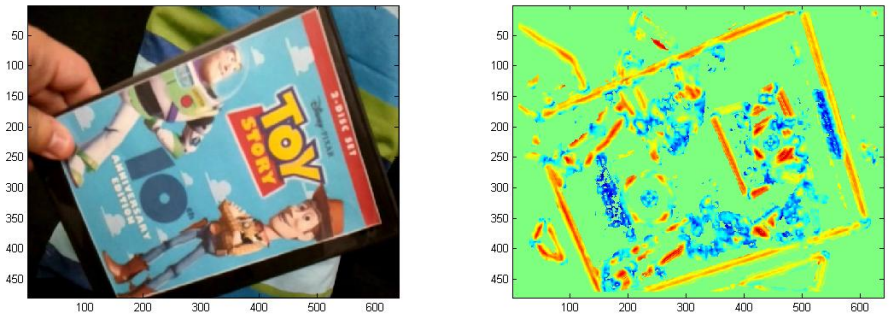


(b) Escala 2

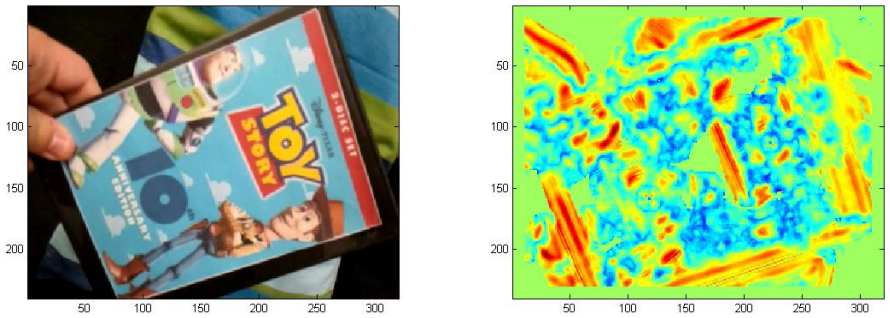


(c) Escala 4

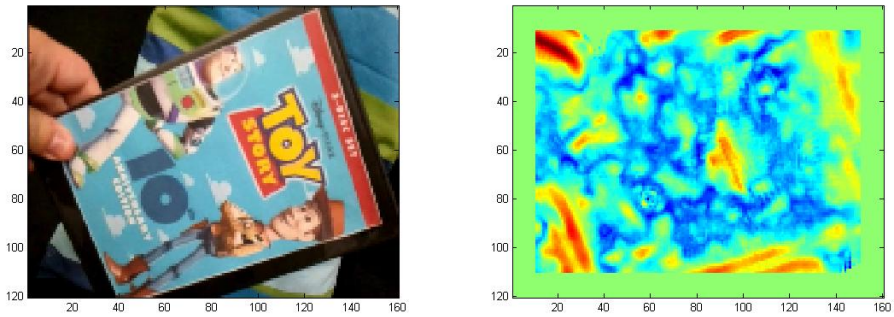
Figura 4.8: Notas obtenidas median Hill Climbing para cada escala.



(a) Escala 1



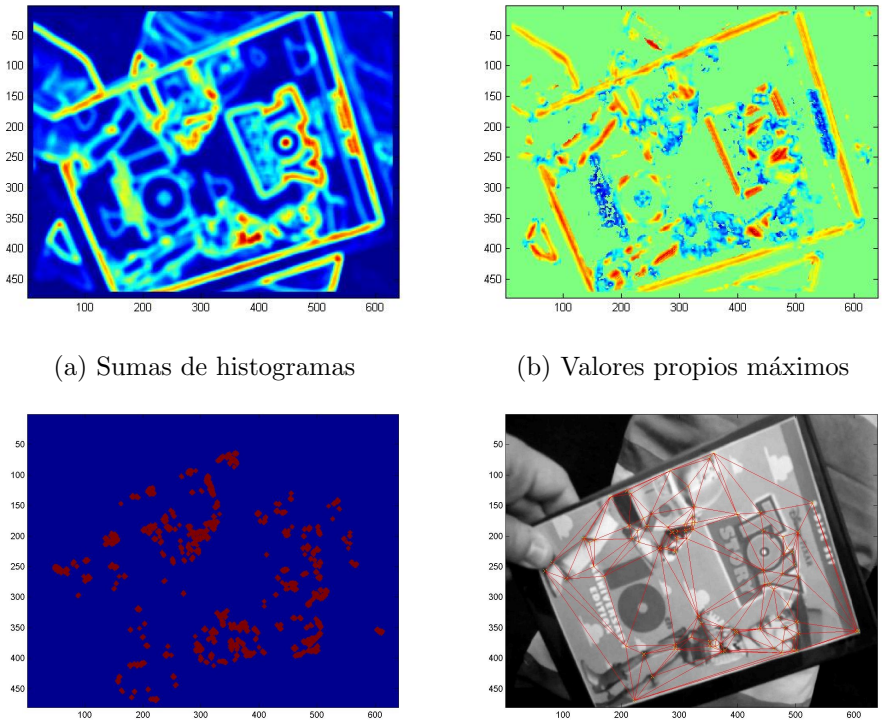
(b) Escala 2



(c) Escala 4

Figura 4.9: Valores propios máximos en cada escala.

Hasta ahora se ha explicado y mostrado la selección de los puntos por cada uno de los factores individualmente. En cambio, el algoritmo implementado, busca aquellos puntos que superen los tres factores simultáneamente. En la Figura 4.10, se muestra el proceso completo de filtrado de los puntos mediante los tres factores descritos, así como los puntos que finalmente cumplen estos tres para la escala 1:



(a) Sumas de histogramas

(b) Valores propios máximos

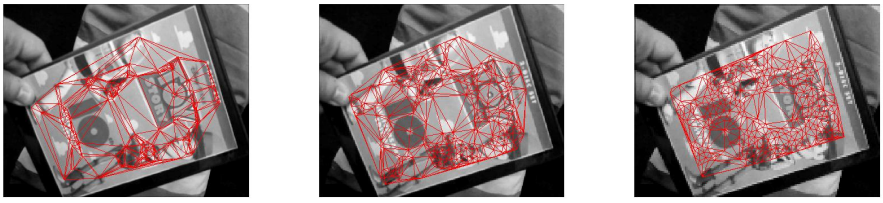
(c) Valores máximos notas

(d) Puntos finalmente seleccionados

Figura 4.10: Pasos para la selección de puntos óptimos.

Tras realizar esta selección de puntos, obtenemos como resultado distintas posiciones favorables para hacer su seguimiento en función de la escala. Es decir, un punto por ser considerado favorable en una escala, no tiene por qué serlo en las demás. Esto se debe a que cada punto no sólo depende de sí mismo, sino también de las características de sus vecinos y, puesto que para obtener escalas menores se ha muestreado la imagen original, descartando píxeles, las características de un mismo punto, de una escala a otra, no siempre se mantienen. Se trata de imágenes distintas, de modo que se obtiene una selección de puntos individual para cada una de las tres escalas en las que trabajaremos. Los puntos obtenidos para cada una de las escalas se pueden observar en la Figura 4.11. En ella se observa que ni el número

de puntos, ni las posiciones de estos coinciden entre escalas.



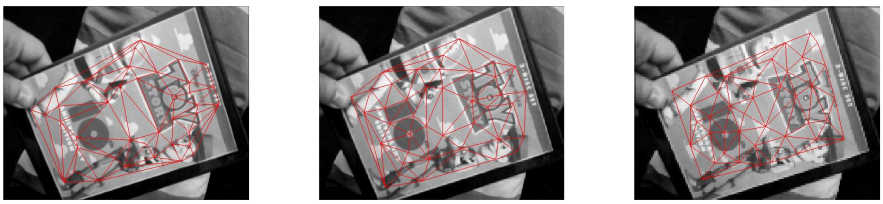
(a) Escala 1 (b) Escala 2 (c) Escala 4

Figura 4.11: Puntos favorables para los tres factores.

Además, tras esta criba, es posible que aún nos quede un gran número de puntos recomendables para hacer su seguimiento en cada una de las escalas. Sin embargo, puesto que queremos converger hacia un método que se pueda optimizar en el menor tiempo posible, no será necesario, ni favorable, para nuestro objetivo hacer el seguimiento de todos y cada uno. Nos quedaremos con un conjunto reducido del orden de 40 puntos para cada escala, que consideramos más que suficiente para el correcto funcionamiento del algoritmo.

La selección final de estos puntos no se basará en las características de apariencia de los puntos sino en características basadas en la estructura de estos. Para ello hemos implementado un método que aplica el uso de la triangulación de Delaunay. Este procedimiento se explica detalladamente en 3.2.3. Se trata de un proceso iterativo, que detendremos cuando obtengamos el número de puntos objetivo.

Por tanto, los 40 puntos seleccionados finalmente en cada una de las escalas utilizadas cumplen una serie de características de apariencia y estructura ideales para poder hacer un buen seguimiento de ellos en los siguientes frames, y con ello poder identificar la posición, rotación, traslación y transformación que ha sufrido el objeto con respecto al inicio. Estos puntos son representados en la Figura 4.12



(a) Escala 1 (b) Escala 2 (c) Escala 4

Figura 4.12: Puntos finales tras los factores de apariencia y estructura.

4.2.1. Sensibilidad de los factores de selección

Para elegir los umbrales que optimicen la selección de puntos ha sido necesario hacer un estudio de la sensibilidad de estos. Para ello, se ha hecho una estimación inicial de en qué rango de valores debían encontrarse los tres por individual, se muestran en la Tabla 4.1, y posteriormente se han hecho pruebas con todas las combinaciones posibles, un total de 27. Estas pruebas consisten en la obtención de los 40 puntos para cada combinación de factores y se observa el comportamiento de nuestra propuesta para estos puntos. Para poder ver la respuesta del método frente a los umbrales elegidos, se hace un cálculo del error medio que se ha estimado de cada una de las cuatro esquinas del objeto. Para saber la importancia del error obtenido, se ha normalizado la distancia Euclídea que indica el número de píxeles, con respecto al perímetro real del objeto en ese frame.

	Umbral 1	Umbral 2	Umbral 3
Suma historiales	2.0	1.5	1.0
Notas H.C.	8 (7) (6)	7 (6) (5)	6 (5) (4)
Valores propios	2	1.99	1.98

Tabla 4.1: Valores de estudio para la sensibilidad de los factores

Como se ha comentado antes, los valores de las sumas de los historiales son muy bajos, por lo que se hace una pequeña normalización para trabajar con ellos en el orden de las unidades, como los otros factores. Con esto, los valores medios de la suma de historiales para las tres escalas está entre 1.4-2.3. Hay que tener en cuenta que si este valor es demasiado restrictivo quitará píxeles que no estén en zonas de textura uniforme, por ello, se hace un estudio de los valores 2, 1.2 y 1.

Para elegir los umbrales de las notas obtenidas por Hill Climbing hay que tener en cuenta que este factor se ve alterado por la escala. Esto se debe a que, para obtener las escalas menores se ha muestreado la original y, por ello, es lógico que sea más difícil encontrar un punto cuanto más se aleje en una escala más grosera. Por este motivo se hace un pequeño ajuste en las pruebas con los umbrales para las escalas. La escala real se prueba con los valores 7, 8 y 9, para la escala 2 se reducen estos valores en una unidad, es decir, vares 6, 7 y 8, y por último, para la escala 4 se vuelven a reducir estos valores en otra unidad, por tanto, 5, 6 y 7 para la escala más grosera.

Para el factor de la covarianza para detectar puntos dentro de contornos, se observa que los valores propios obtenidos de todos los punto se encuentran en un rango muy pequeño que va de 1.92 a 2.07. Mirando la paridad de

los valores propios, se obtiene que sólo en caso que uno de los valores tenga una puntuación superior a 2, el otro valor propio es notablemente dispar, del orden de 1.94. Por este motivo, ya que sólo se han guardado los valores propios mayores, con un umbral de 2 se desecharían aquellos valores propios dispares, que corresponden a los puntos dentro de un contorno, esto se puede observar en la Figura 4.9. Sin embargo, como se ha comentado en 4.2, elegiremos un umbral más restrictivo para eliminar también aquellos puntos que pueden ser sensibles al ruido, y que tiene los valores propios más atenuados, por ello se harán pruebas con los valores 2, 1.99 y 1.98.

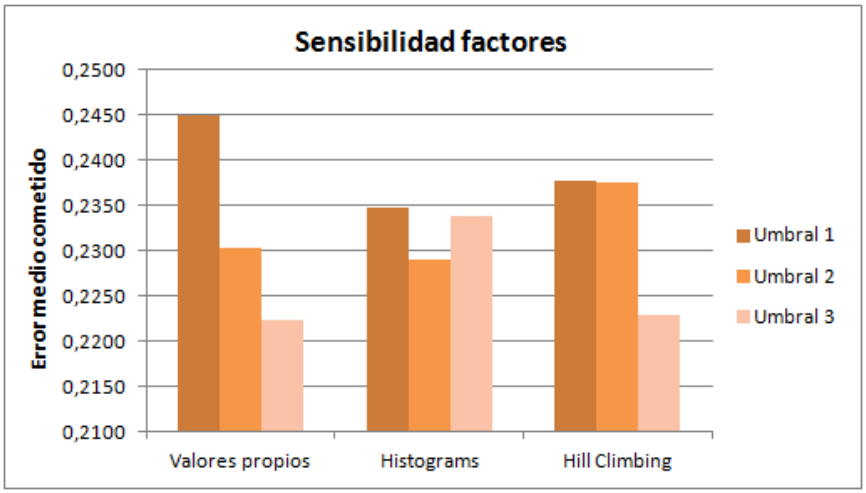


Figura 4.13: Notas medias obtenidas en función de los tres umbrales de cada factor.

Tras realizar todas las combinaciones posibles hemos obtenido como valores umbrales óptimos 2 para la suma de los histogramas, 6 para las notas en la escala real, y por consiguiente 5 y 4 para las escalas menores, y 1.98 para los valores propios de las covarianzas. Sin embargo, se deduce de estas pruebas que los factores de notas de Hill Climbing y valores propios de la covarianza son más críticos que el umbral de la suma de histogramas (Figura 4.13) y, puesto que los umbrales obtenidos se encuentran en los extremos de los rangos elegidos para estos dos se han comprueba hasta que valores de éstos dos umbrales mejor el rendimiento. De este modo, se obtiene un mejor rendimiento cuando el umbral del algoritmo de seguimiento es de 5 para la escala real, 4 y 3 para las menores, y 1.97 para el factor de los valores propios. Sin embargo, ésta combinación es crítica, en caso que se produzca un cambio rápido en el objeto o exista oclusión, los puntos con las notas de seguimiento menores se perderán y el rendimiento del algoritmo principal disminuirá. Tras realizar una prueba de estos casos se confirma esta hipótesis. Para hacer frente a estos casos, se escoge la siguiente combina-

ción de umbrales en cuanto a buen rendimiento. Se concluye que los valores umbrales óptimos para los tres factores son:

- Suma de histogramas: 2
- Notas de seguimiento: 6 escala 1, 5 escala 2 y 4 escala 4.
- Valores propios covarianza: 1.98

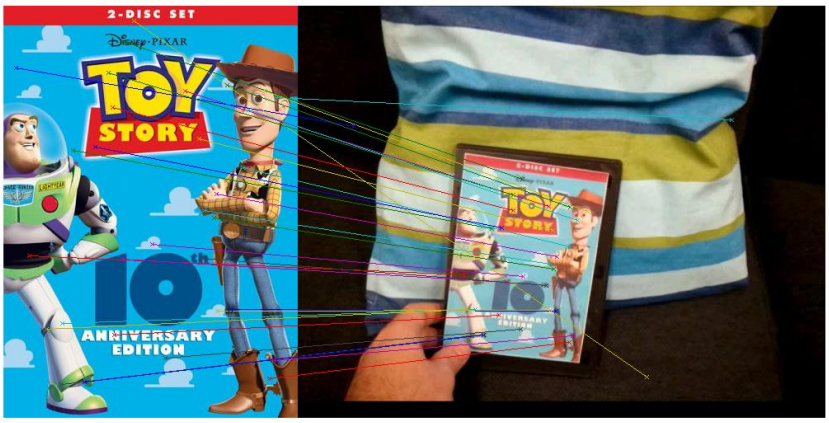
4.3. Inicialización del algoritmo

Antes de poder hacer el seguimiento de un objeto dentro de una secuencia de fotogramas es necesario identificar la posición del objeto dentro de la escena.

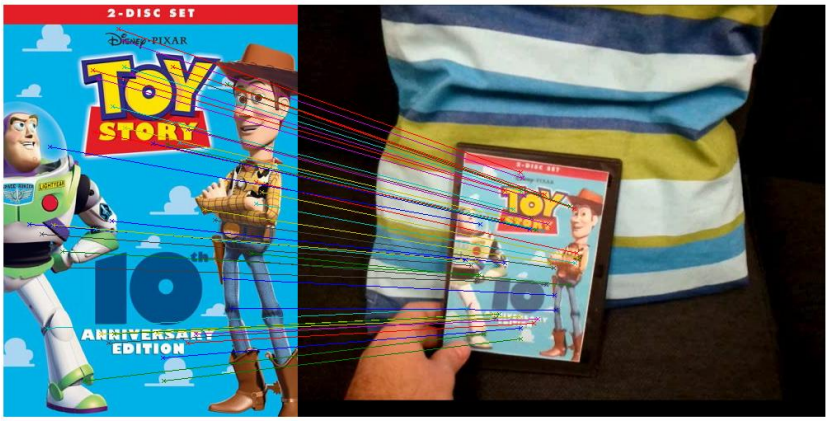
Para ello, partiremos del modelo del objeto aislado, *template* que contiene descriptores de características de los puntos más significativos dentro del objeto. Hemos elegido un modelo de tipo SIFT, ya que presentaba mejor rendimiento que el tipo SURF, como se detalla en 2.1.

Consiste en situar el modelo dentro de la imagen, buscar los puntos con mayor similitud, y a partir de ellos, crear una homografía para situar los puntos que hemos seleccionado como más favorables mediante el procedimiento descrito en la sección anterior. Es necesario hacer un emparejamiento de los descriptores en el *template* y los pertenecientes a la escena completa, para ello, compararemos la similitud entre histogramas de ambos sitios.

Al realizar el emparejamiento, puede hacerse una mala asignación debido a que dos zonas tengan valores similares por ser éstos muy grandes, por ello, es necesario hacer una normalización de los histogramas previa al emparejamiento, de este modo es posible cuantificar la similitud entre puntos. Se hace uso de la distancia de Bhattcharyya (3.1), para cuantificar la similitud entre los histogramas normalizados, siendo buen emparejamiento aquel que tenga valor 1 y mal emparejamiento los que no tienen valores próximos a 1.



(a) Histogramas sin normalizar y con distancia Euclídea



(b) Histogramas normalizados y con distancia Bhattcharyya

Figura 4.14: Emparejamiento en función de la distancia escogida.

En la Figura 4.14a se observa como aparecen 2 espúreos claros, situados a bastante distancia de la posición del objeto, además de otros que se encuentran dentro del objeto. El emparejamiento no tiene nada que ver, ya que los puntos situados fuera del objeto se encuentran en zonas de textura uniforme. Sin embargo, gracias a la normalización de los histogramas, estos errores no aparecen en 4.14b.

Para evitar los espúreos intrínsecos al objeto, que deforman la transformación que calculamos mediante homografía, se hace un segundo análisis de los puntos resultantes mediante modelado estructural. De nuevo, hacemos uso de la triangulación de Delaunay con los puntos obtenidos. A partir de

un conjunto de triángulos obtenidos en la triangulación formada por la nube de puntos en el *template*, se calculan las coordenadas baricéntricas de todos los puntos. Puesto que estas son invariantes a transformaciones, se hace el cálculo de las coordenadas cartesianas en el objeto dentro de la escena que se está buscando. Se hace una ponderación de las coordenadas cartesianas obtenidas por todos los triángulos seleccionados para evitar el caso en que alguno de los puntos incluidos en el triángulo de referencia sea un espúreo. Por último, aquellos puntos que disten en una medida considerable de las coordenadas cartesianas obtenidas, son descartados para realizar la homografía.

(a) Sin modelado estructural (b) Con modelado estructural

Figura 4.15: Estimación posición objeto dentro de la primera escena.

4.4. Cuerpo principal

Tras la elección de los puntos óptimos el algoritmo hace uso de todos los conceptos explicados a lo largo de esta memoria para obtener una secuencia de vídeo donde se hace el seguimiento de un objeto que sufre distintas transformaciones geométricas.

La imagen original, que consiste en el primer *frame* de la secuencia de vídeo a estudiar, se muestrea para conseguir la imagen en escala 2 y 4, éstas serán las escalas sobre las que se trabajará durante todo el algoritmo. Para cada escala se genera un modelo donde se obtienen para cada punto un mapa de orientaciones con todas sus posibles rotaciones, siguiente el procedimiento explicado en 4.1.1.

También en este primer fotograma, únicamente para la escala más grosera, se crea una triangulación de Delaunay y se almacenan las coordenadas baricéntricas de todos los puntos respecto a todos los triángulos, ya que co-

mo se detalle a lo largo del Capítulo 3.2, estas son invariantes a lo largo de la secuencia, y se utilizarán para estimar las coordenadas cartesianas de los puntos en los *frames* siguientes.

Figura 4.16: Imágenes escaladas tras muestrear por factores 2 y 4.

A partir de la información recopilada del primer fotograma,se podrá hacer el seguimiento a lo largo de toda la secuencia, mediante la combinación de tres factores: algoritmo de seguimiento Mean Shift, modelado estructural con coordenadas baricéntricas, homografía de los puntos que han obtenido un buen resultado de Mean Shift.

Para cada fotograma a lo largo de la secuencia, se trabaja primeramente en la escala más grosera para reducir los tiempos de búsqueda, y posteriormente se hace un cambio a la escala real con los resultados obtenidos para posteriormente afinar esta búsqueda al detalle que proporciona la escala real y se había perdido en la más grosera.

El procedimiento que sigue el algoritmo a partir de este momento es, para cada fotograma, se muestrea por un factor 4 para obtener la misma imagen en una escala menor que nos permita trabajar de una manera más ligera. En esta escala se buscan los mismos puntos que teníamos en el *frame* predecesor a partir del algoritmo de seguimiento Mean Shift. Durante este algoritmo, a cada punto se le ha asignado una nota correspondiente a la distancia de Bhattcharyya del punto que se ha elegido como máximo. Utilizaremos estas notas para determinar si la detección realizada por Mean Shift es fiable o no. Con esto, se diferencian los puntos en *buenos* y *malos* en función de dicha nota, considerando como buenos aquellos que están por encima de la mediana de las notas de conjunto y tienen un valor superior a 0.9. Se hará una corrección de los puntos malos a partir de la homografía creada por los puntos considerados buenos.

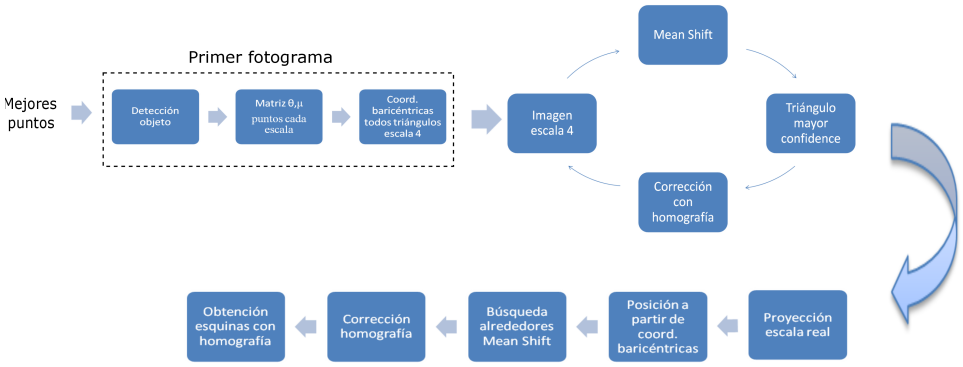


Figura 4.17: Proceso completo del algoritmo principal.

Como se ha explicado en 3, en caso que el movimiento del objeto sea muy rápido o existan oclusiones, Mean Shift no es suficiente para hacer una buena detección, y es por ello, que hacemos uso del modelado estrutural. Por ello, a partir de la malla de triángulos de Delaunay, se obtiene aquel triángulo con mayor *confidence* (3.2), se proyecta este triángulo en las escalas superiores y se calculan las coordenadas baricéntricas de los puntos en el fotograma inicial de las escalas superiores teniendo como referencia el triángulo obtenido en la escala 4 proyectado. Antes de hacer uso de las coordenadas obtenidas, se ajusta la escala en que se está trabajando a la escala real del objeto en la imagen actual. Tras este ajuste, se obtienen las coordenadas cartesianas de los puntos en la escala real del frame actual, a partir de las coordenadas baricéntricas obtenidas respecto al mejor triángulo para los puntos del frame inicial.

Las posiciones obtenidas las consideramos una primera aproximación a la posición real en que se encuentra por obtenerse a partir del triángulo en la escala más grosera, de modo que, para encontrar la posición exacta, se vuelve a hacer uso del algoritmo Mean Shift, para encontrar el máximo en las proximidades de las coordenadas obtenidas. Al igual que en la escala 4, se hará un ajuste de los resultados obtenidos mediante la homografía de los puntos considerados buenos.

Finalmente, se obtienen las posiciones de las esquinas del objeto a partir de una homografía con los puntos obtenidos.

Capítulo 5

Resultados y conclusiones

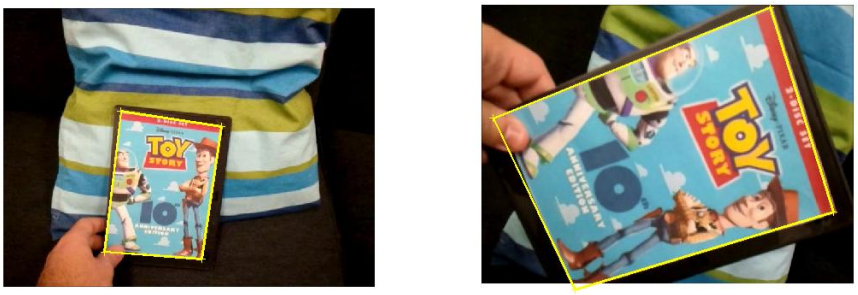
Con lo explicado a lo largo de la memoria y siguiendo el procedimiento descrito, se ha hecho una serie de pruebas para analizar el rendimiento del algoritmo. Las pruebas que se han realizado, son similares a las que se realizaron en 2.1 para los descriptores SIFT y SURF. Las bases de datos empleadas, así como las oclusiones sintéticas que se añaden son las mismas.

Por tanto, las pruebas que se han hecho son las siguientes: seguimiento de un objeto en dos bases de datos diferentes, seguimiento de un objeto con cambio de escala, seguimiento de un objeto en una secuencia con oclusiones.

Seguimiento de un objeto en dos bases de datos diferentes. Las bases utilizadas son las mismas que se emplearon en el Capítulo 2. La base de datos nº 1 consiste en una escena con un objeto que sufre principalmente rotación, mientras que en la base de datos nº 2 el objeto sufre escalado, rotación, traslación, transformación afín.

Seguimiento de un objeto con cambio de escala. En el algoritmo se tiene en cuenta el cambio de escala mediante el modelado estructural, proyectando triángulos entre escalas y empleando las coordenadas baricéntricas en base a estos. Por tanto, el algoritmo debe ser invariante al cambio de escala.

En la Figura 5.1, se observa el cambio de escala que sufre el objeto, modificando su tamaño en un factor de 2. En ella también se observa, que a pesar del cambio de escala, el algoritmo es capaz de realizar su seguimiento y situar su estructura dentro de la escena. Las líneas amarillas que encuadran el objeto son la estimación de la estructura del objeto que hace el algoritmo.



(a) Escala menor (b) Escala mayor

Figura 5.1: Cambio de escala de un objeto

En la Gráfica 5.2 se presentan las dos bases de datos para examinar el efecto que tiene que exista un cambio de escala del objeto en la secuencia o no. La base de datos nº 1, en color naranja, aunque presenta un ligero cambio de escala, no es significativo y se puede hacer su seguimiento sin tener en cuenta este factor. En cambio, en la segunda base de datos, en color azul, el objeto que se está siguiendo llega a reducir su tamaño en un factor 2. Se corresponde con las imágenes de 5.1.

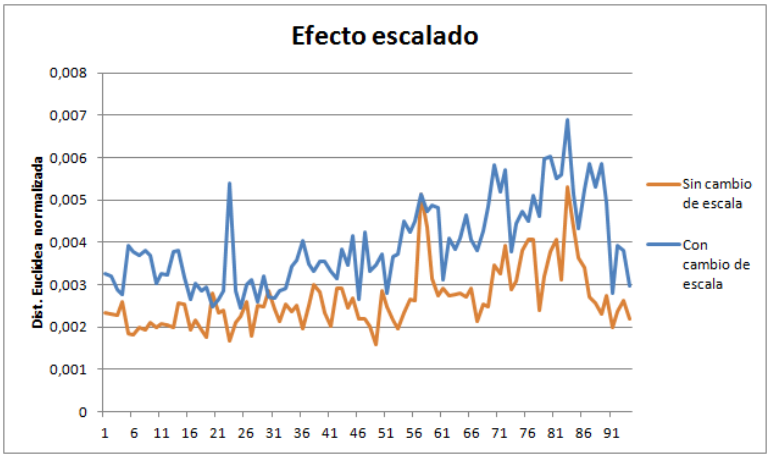


Figura 5.2: Resultados obtenidos con nuestra propuesta en el caso que exista cambio de escala

Como es de esperar, presenta un peor comportamiento en el caso de existir cambio de escala, aunque éste no es significativo, ya que ambas bases de datos presentan valores de error similares. Además, puesto que se trata de base de datos reales y no es posible aislar las transformaciones que sufre el objeto, estas no tienen por qué ser necesariamente debido a este factor, y

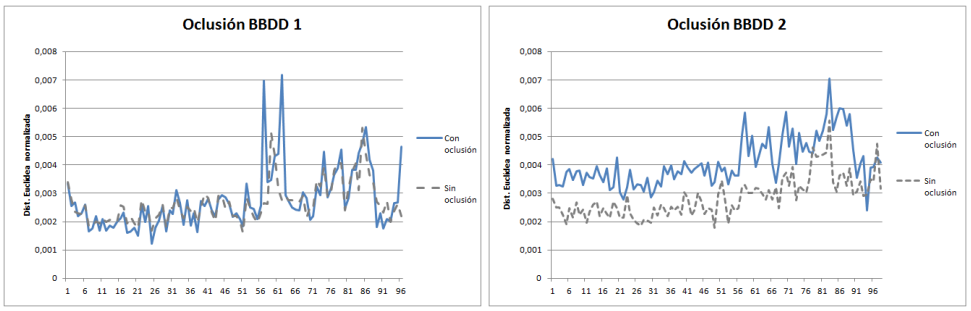
la diferencia de errores puede ser debido a que se están eligiendo dos bases de datos distintas con características diferentes. Este error no es apreciable por el ojo ya que se trata de unos pocos píxeles, por tanto, el algoritmo es prácticamente invariable a escala.

Seguimiento de un objeto en una secuencia con oclusiones. Debido al uso de las coordenadas baricéntricas, el algoritmo debe ser invariante a movimientos rápidos u oclusiones. Para comprobar el funcionamiento del algoritmo frente a situaciones de oclusión, se ha creado sintéticamente una oclusión a partir del fondo de la propia imagen, y que ocupa un cuarto de la imagen. Las oclusiones máximas que llegan a tener los objetos para las dos bases de datos se observa en 5.3.

(a) Máxima oclusión base de datos 1. (b) Máxima oclusión base de datos 2.

Figura 5.3: Oclusión máxima durante la secuencia de vídeo.

.



(a) Error base de datos 1. (b) Error base de datos 2.

Figura 5.4: Error durante la secuencia con oclusión.

.

Las gráficas correspondientes a los errores producidos por esas oclusiones, se observan en 5.4. En ella se observa que el algoritmo presenta un

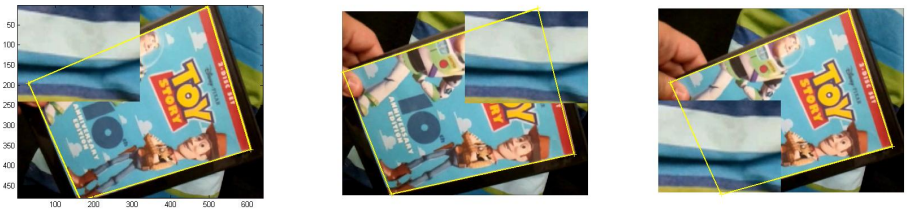
comportamiento muy similar para los casos con y sin oclusión en ambas bases de datos. Como es de esperar, el comportamiento con oclusiones es ligeramente peor, pero no es significativo, en ninguno de los casos, ni siquiera para el caso en que la secuencia además de presentar oclusión, presenta transformaciones, rotación y escalado significativo.

El tamaño y posición de la oclusión del objeto es determinante, pues se entiende, que a mayor oclusión, más difícil es la visibilidad del objeto y con ello su detección en la imagen. Por ello, se examina el método para tres tipos de oclusiones sintéticas, cada una de ellas ocupa una región de un cuarto de la imagen, y se sitúan en los cuadrantes 1, 2 y 3 respectivamente, siendo la del cuadrante 2 la que se ha examinado previamente.

(a) Oclusión 1. (b) Oclusión 2. (c) Oclusión 3.

Figura 5.5: Oclusiones sintéticas base de datos 1.

.

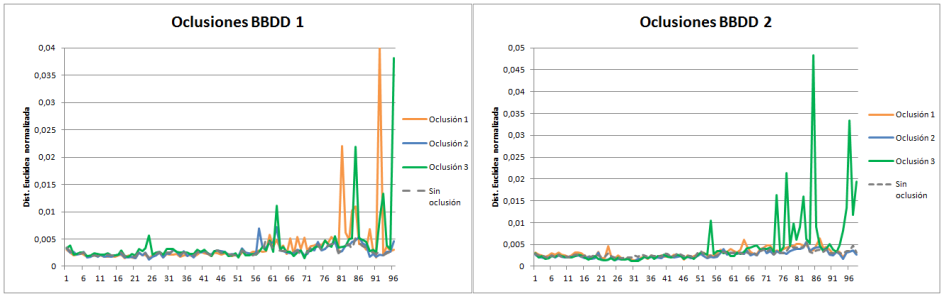


(a) Oclusión 1. (b) Oclusión 2. (c) Oclusión 3.

Figura 5.6: Oclusiones sintéticas base de datos21.

.

En 5.7 se observan los errores obtenidos en función del tipo de oclusión a la que se expone la secuencia de vídeo.Se observan picos correspondientes a fotogramas en que la oclusión es alrededor del 50 % de la superficie del objeto. Como es lógico, la posición de la oclusión no afecta de la misma manera a las dos base de datos, ya que la posición del objeto en la escena varía entre ambas.



(a) Error base de datos 1. (b) Error base de datos 2.

Figura 5.7: Error producido para cada oclusión en cada base de datos.

.

Eludiendo estos picos, se observa como para ambas bases de datos se presenta un comportamiento muy similar en las situaciones con y sin oclusión. Por tanto, se puede decir que el algoritmo supera en invarianza ante oclusiones a los descriptores con que se comparaban en el principio de esta memoria.

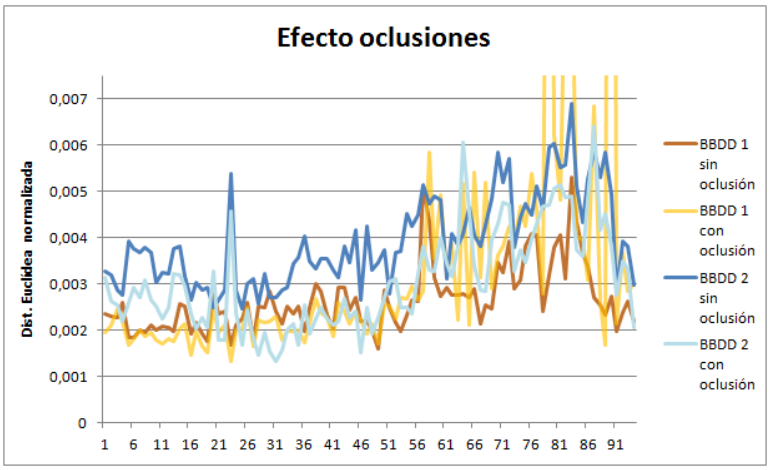
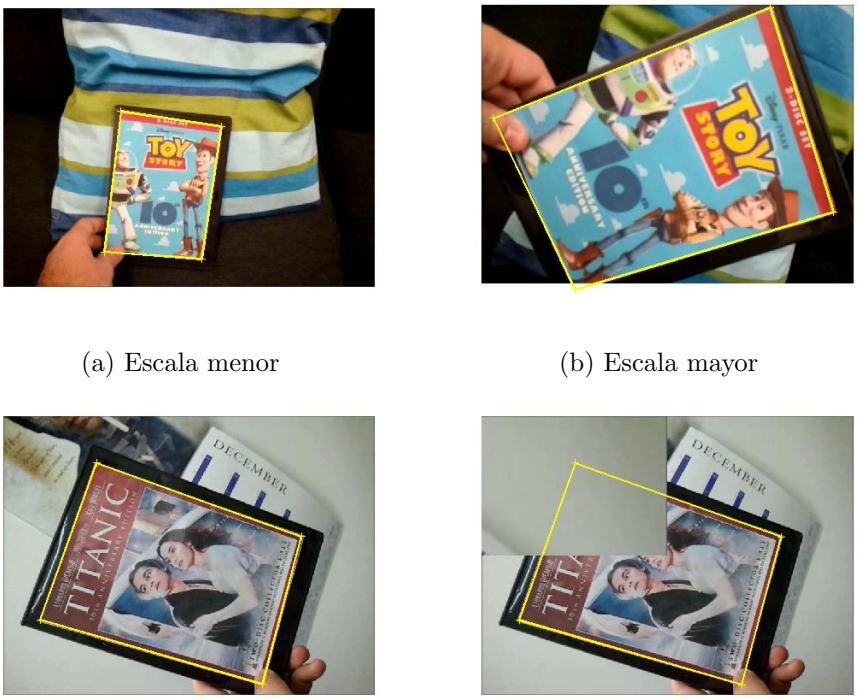


Figura 5.8: Comparación efecto oclusión en las dos bases de datos.

Por otro lado, se observa en 5.8, que el error de ambas bases de datos es similar con y sin oclusiones y entre ellas. En esta figura, se ha hecho un zoom de las zonas donde no existe el pico de error, ya que este depende de la posición de la oclusión.

Por tanto, el algoritmo es capaz de identificar un objeto y realizar su seguimiento a pesar de estar parcialmente ocluido durante toda la secuencia de vídeo. Esto es una mejora notable con respecto a los descriptores analizados en 2.1, ya que ninguno era capaz de hacer una buen seguimiento y en caso que existiera oclusiones, la tasa de error era muy elevada.



(a) Escala menor

(b) Escala mayor

(c) Sin oclusión

(d) Con clusión

Figura 5.9: Tracking de objeto ante distintas situaciones

Tras los resultados obtenidos, se concluye que nuestro algoritmo es más fiable que sus predecesores para el seguimiento dentro de una secuencia de vídeo, ya que los primeros están orientados al reconocimiento de un objeto dentro de una imagen aislada y, en cambio, nuestro algoritmo, gracias a la combinación de descriptor de características con algoritmo de seguimiento, modelado estructural y homografía, crea un método robusto, capaz de identificar la posición del objeto en cada frame y pudiendo estimar su estructura.

En la tabla 5.1 se ve como los valores medios del error mejoran notablemente frente a la detección continua mediante SIFT descrita en 2, que era quien presentaba las mejores respuestas de las tres alternativas que se analizaron, así como frente al descriptor SURF.

	Detección continua SIFT	Nuestra propuesta
Sin oclusión	0,0145	0,0028
Con oclusión	0,0098	0,0041

Tabla 5.1: Error medio normalizado obtenido para cada situación. Comparación con los métodos analizados en el Capítulo 2.

Planteando un descriptor basado en mapa de orientaciones de histogramas con coordenadas angulares, se ha conseguido dotar de invarianza a rotación del objeto incluso existen ruido alrededor del punto, superando así las carencias del desriptor SIFT. Además, por trabajar con coordenadas angulares, el rango de búsqueda de la rotación está acotado.

Mediante el método de seguimiento Mean Shift y la homografía entre puntos dentro del objeto, hace del algoritmo, invariante a transformaciones y traslaciones. Es cierto que Mean Shift puede no ser útil en caso que exista un movimiento rápido o haya oclusiones, por tanto, para que el método sea invariante a éstos se combina el método de seguimiento con el modelado estructural del objeto. Y es precisamente, este modelo estuctural, del que hace uso entre escalas de una misma imagen, el que dota al algoritmo de inviarianza ante un cambio de escala del objeto.

5.1. Trabajos futuros

Para trabajos futuros, uno de los aspectos más importantes a mejorar es el tiempo de computación. Hay un amplio margen para reducir tiempos de ejecución y poder dar uso del algoritmo en aplicaciones en tiempo real.

Tras realizar estas mejoras, se plantea la implementación del código en lenguajes de programación que consuman menos tiempo computacional en el tratamiento de imágenes y que sean soportados por las tecnologías y dispositivos móviles que inundan el día a día en la actualidad.

Por otro lado, nuestro algoritmo es robusto a oclusiones, en el caso que exista una oclusión prácticamente íntegra del objeto, detectamos que existe dicha oclusión, pero para poder hacer una estimación de la posición del objeto mediante homografía de los puntos detectados, es necesario que el número de puntos sean al menos 4. Para poder dar continuidad a la secuencia a partir de una oclusión en la que no se encuentren al menos esos 4 puntos, se podrá hacer la detección mediante el algoritmo de inicialización explicado en 4.3, de este modo, se podrá hacer una nueva búsqueda del objeto mediante el modelo del template y continuar con el seguimiento.

Este algoritmo, aunque hace seguimiento de los puntos característicos del objeto, no hace seguimiento del objeto propiamente dicho, ya que Mean Shift no hace un tracking global, sino local y en caso de que el movimiento sea mayor, no es capaz de encontrar el punto. Esta carencia se ha solucionado mediante el modelado estructural, sin embargo, sería posible combinar el algoritmo Mean Shift con un filtro de Kalman, el cual utiliza la dinámica de movimiento, lo que puede conseguir que el algoritmo converja más rápidamente.

Parte II

Anexos

Apéndice A

Scale Invariant Features Transform. SIFT

En 1999, David G. Lowe desarrolló el descriptor Scale Invariant Features Transform [4] como algoritmo capaz de detectar puntos característicos en una imagen. Las características son invariantes a escala, rotación y parcialmente invariantes al cambio de iluminación y punto de vista de una cámara 3D. Son localizados en el dominio de espacio y frecuencia reduciendo la probabilidad de confusión por oclusión, ruido o pequeñas distorsiones Realiza la correspondencia entre puntos basada en los vectores de características de cada punto que componen el descriptor de la imagen. El algoritmo se divide en 4 etapas fundamentales descritas a continuación.

1. Detección de Extremos en el Espacio Escala:

   La primera etapa del algoritmo realiza una búsqueda sobre las diferentes escalas y dimensiones de la imagen identificando posibles puntos de interés, invariantes a los cambios de orientación y escalado. Esto se lleva a cabo mediante la función DoG (Difference-of-Gaussian).

   La función Diferencia de Gaussianas, $D(x, y, \sigma)$,se define de la siguiente manera:

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma) \tag{A.1}$$

   donde:

$$L(x, y, \sigma) = I(x, y) * G(x, y, \sigma)$$

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{A.2}$$

Siendo (x, y) las coordenadas horizontal y vertical, $I(x, y)$ la imagen de entrada al sistema y σ la desviación estándar del filtro Gaussiano.

Se trata de una función eficiente de calcular, que proporciona una buena aproximación a la normalización en escala Laplaciana de Gaussianas , $\sigma^2 \bigtriangledown^2 G$, que estudió Linerberg (1994) y, según la cual, demostró la necesidad de normalizar la Laplaciana con el factor σ^2 para obtener invarianza respecto de la escala. Posteriormente, en 2002, Mikolajczyk, descubrió que los máximos y mínimos de $\sigma^2 \bigtriangledown^2 G$ produce las características locales más estables comparadas con de otras posibles funciones como el gradiente, la función Hessiana o la función de detección de esquinas de Harris.

Siendo

$$\frac{\partial G}{\partial \sigma} = \sigma^2 \bigtriangledown^2 G \qquad \sigma^2 \bigtriangledown^2 G = \frac{\partial G}{\partial \sigma} \approx \frac{G(x, y, k\sigma) - G(x, y, \sigma)}{k\sigma - \sigma} \tag{A.3}$$

La diferencia de Gaussianas se puede aproximar como

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \approx ((k-1) sigma^2 \bigtriangledown^2 G) * I(x, y) \tag{A.4}$$

Con esto, se pueden identificar los puntos de interés invariantes a escala y rotación, comparando cada punto con sus 8 vecinos en la escala en la que se encuentra, y con los 18 vecinos de la escala superior e inferior. Será necesario, por tanto, ajustar la frecuencia de muestreo tanto del espacio como de la escala, teniendo en cuenta que los extremos muy cercanos son inestables cuando hay ruido en la imagen.

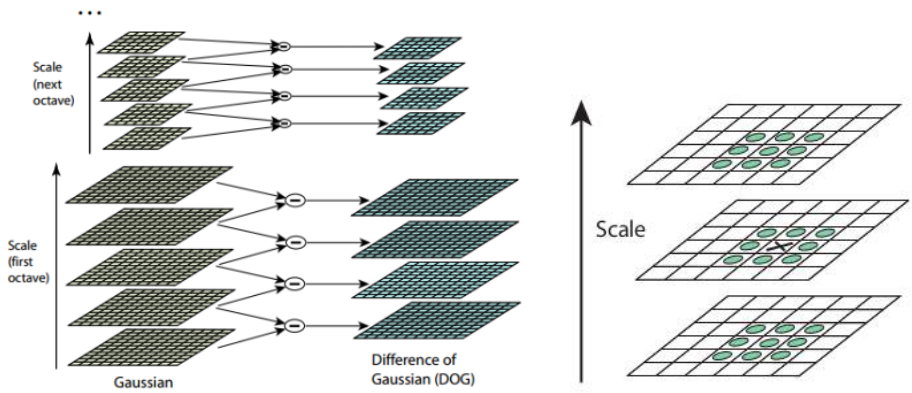


Figura A.1: Detección de puntos

2. Localización de puntos clave

Tras realizar el primer paso, existen múltiples candidatos, no todos igual de favorables. L os puntos no firmemente situados sobre los bordes o aquellos con bajo contraste son bastante vulnerables al ruido y por lo tanto no podrán ser detectados bajo pequeños cambios de iluminación o variación del punto de vista de la imagen. Es por esto, que en esta etapa se hace una segunda selección para seleccionar los puntos clave, también llamados puntos de interés o keypoints, de forma precisa.

Para eliminar los puntos con bajo contraste, se aplica un proceso de umbralización del siguiente modo: primero, se estima la diferencia de Gaussianas descrita anteriormente, entorno a uno de los puntos seleccionados en la primera etapa, mediante el desarrollo de Taylor de orden 2.

$$D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^T\frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x} \tag{A.5}$$

donde $x = (x, y, \sigma)^T$ es el offset del punto clave. La localización del extremo, $\hat{x}$ es determinado mediante la derivada de la función respecto a x en el punto 0.

Se calcula el valor del desarrollo en serie de Taylor, con el offset $\hat{x}$ en caso que el valor sea menor que 0.03, se descarta, si no la localización espacial final del descriptor será $y + \hat{x}$ donde y es la posición inicial del punto.

Una vez descartados los puntos con bajo contraste, es necesario eliminar los puntos situados sobre bordes de manera difusa, que conllevan un alto grado de inestabilidad incluso ante pequeños ruidos. Para ello, se utiliza la propiedad de la función DoG atendiendo a la gran curvatura que presenta en la dirección paralela al borde y la pequeña curvatura que se observa en la dirección perpendicular. Estas respuestas se pueden estudiar mediante el cálculo de la matriz del Hessiana sobre la localización y escala del punto en estudio:

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{yx} & D_{yy} \end{bmatrix}$$

$$Tr(\mathbf{H}) = D_{xx} + D_{yy} = \alpha + \beta \quad Det(\mathbf{H}) = D_{xx}D_{yy} - D_{yy}^2 = \alpha\beta \tag{A.6}$$

Sea H la matriz Hessiana de $D(x, y, \sigma)$, siendo α y β sus valores propios, dicho punto se encontrará en un borde si éstos son dispares, uno grande y otro pequeño. Si $\alpha = r.\beta$, la condición se reduce a:

$$\frac{Tr(\mathbf{H})^2}{Det(\mathbf{H})} < \frac{(r+1)^2}{r} \tag{A.7}$$

Finalmente, Lowe propone un umbral r=10. Tras descartar los puntos inestables, se les asignará una orientación a los keypoints obtenidos.

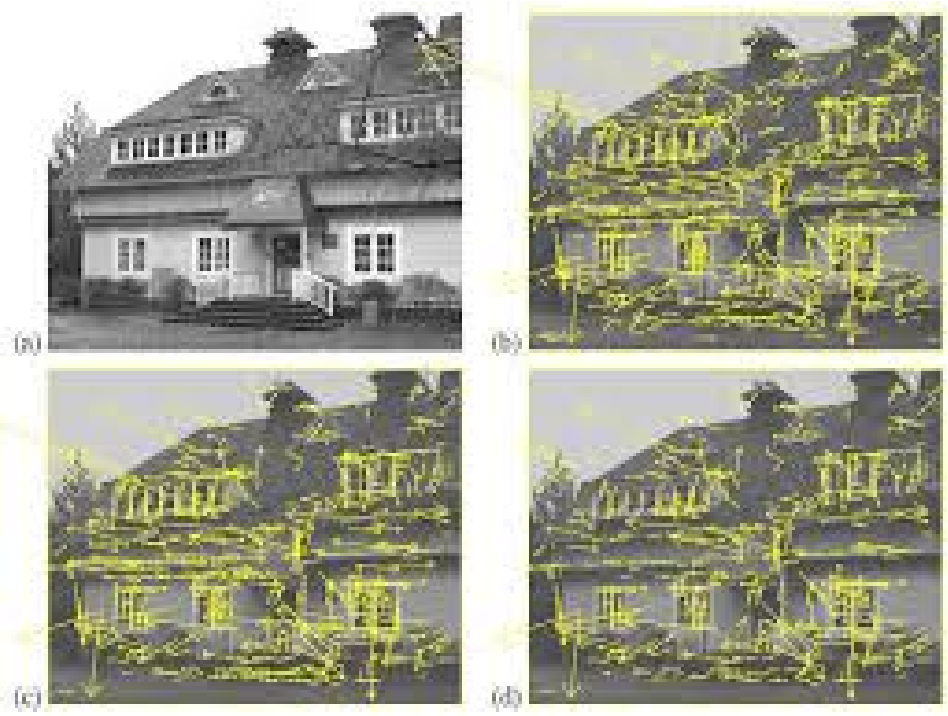


Figura A.2: Localización de puntos

3. Asignación de la orientación

Una de las principales características de los puntos SIFT es que éstos son invariantes a una serie de transformaciones sobre la imagen.

La invarianza respecto a la rotación se obtiene mediante la asignación, a cada uno de los puntos clave obtenidos en el paso anterior, de una orientación basada en las propiedades locales de la imagen. Para cada punto clave, se calcula la magnitud, m, y orientación, θ, del gradiente mediante

las ecuaciones:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \tag{A.8}$$

$$\theta(x,y) = \tan^{-1}\left(L(x,y+1) - L(x,y-1)\right)/(L(x+1,y) - L(x-1,y)) \tag{A.9}$$

Se crea un histograma de orientaciones de gradiente con 36 posibles direcciones, cada uno de ellos con 10° de amplitud para cubrir el rango de los 360°. Se escogerá como orientación dominante aquella con respuesta mayor y el descriptor quedará representado respecto de éstas.

4. Descriptor del punto clave

   Tras realizar las etapas anteriores, se han seleccionado como puntos de interés aquellos que presentan mayor invarianza respecto a la orientación, escala y localización. En este último paso se determinará un descriptor basado en el entorno del mismo.

   Para cada keypoint, se estudiará su entorno en un área de 16x16 píxeles. Esta región se divide en bloques de 4x4, donde para cada uno de ellos se obtendrá un histograma de orientaciones de gradiente, como en la etapa anterior, en este caso con 8 orientaciones. Por tanto, tras obtener los histogramas de orientaciones de todo el área, se obtendrá un descriptor de 128 valores. Estos 128 son porque la región que es de 256 píxeles (16x16) se subdivide en 16 bloques de 4x4 píxeles, y para cada uno de esos 16 bloques, se obtiene un histograma de 8 posiciones, por tanto, 16 bloques x 8 orientaciones son 128 valores para cada keypoint.

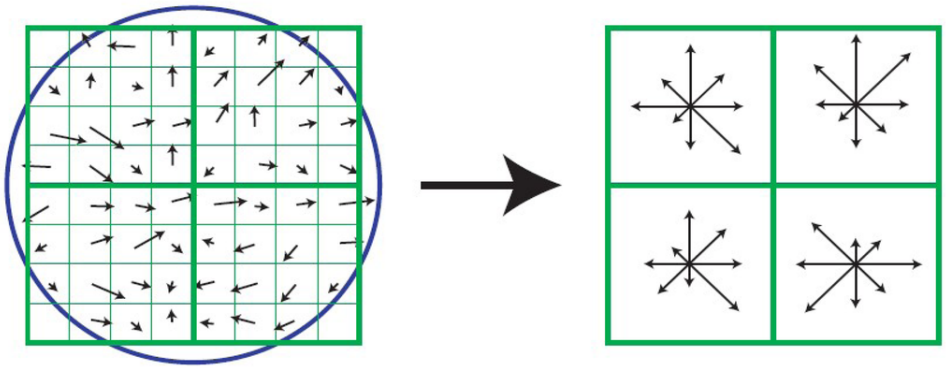


Figura A.3: Descriptor de puntos

Finalmente, se limita el valor de cada componente de magnitud de gradiente a un valor máximo para que tenga un mayor peso la orientación frente a la magnitud del gradiente. El valor optimizado del umbral según Lowe es de 0,2. Por último, se vuelve a normalizar de nuevo a una amplitud unidad.

Apéndice B

Speeded-Up Robust Features. SURF

El descriptor SURF, fue publicado en 2006 por Herbert Bay et al. como detector y descriptor de puntos de interés robusto. Está basado en el descriptor SIFT, tratando de mejorar la velocidad computacional de este. Los autores afirman que SURF presenta principalmente dos mejoras: velocidad de cálculo considerablemente superior sin ocasionar pérdida del rendimiento, y mayor robustez ante posibles transformaciones de la imagen. Sin embargo, posteriormente se comprobó que sin tener en cuenta el tiempo computacional, los puntos SIFT dan un mejor rendimiento.

Son diferencias claves con su predecesor las siguientes:

- El descriptor SURF utiliza siempre la imagen original, escalando no ésta sino los filtro empleados.
- Hace uso del determinante de la matriz Hessiana para calcular posición y escala de los puntos de interés
- La longitud de los vectores de características de los puntos de interés que obtiene es de 64 posiciones en lugar de las 128 del descriptor SIFT.

Para conseguir esto, el proceso del descriptor SURF se divide en los siguientes tres pasos.

1. Detección de puntos de interés

   Como ya hemos adelantado, SURF hace uso de la matriz Hessiana para la localización y escala de los puntos de interés. No hace uso de diferentes medidas de la imagen, sino que utiliza el valor del determinante de la matriz Hessiana.

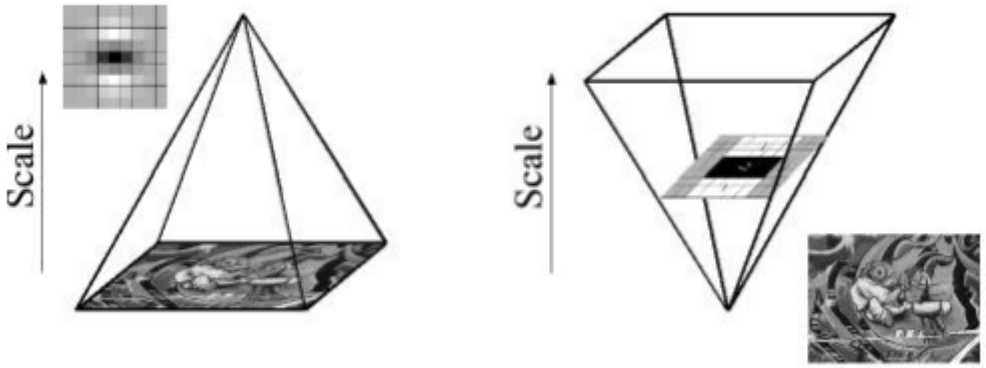


Figura B.1: Comparación métodos de búsqueda descriptores SIFT y SURF

Además, se ha implementado una alternativa a los filtros gaussianos, los filtros tipo caja (box-filters). Estos filtros pueden ser evaluados de manera rápida hacienod uso de imágenes integrales, independientemente del tamaño de éstas.

Debido a la utilización de filtros de tipo caja e imágenes integrales, no es necesario aplicar el mismo filtro iterativamente a la salida de una capa filtrada previamente, sino que se pueden aplicar dichos filtros de cualquier tamaño a la misma velocidad directamente sobre la imagen original. De este modo resulta que el espacio escala es analizado mediante la elevación del tamaño del filtro, en vez de reducir el tamaño de la imagen como hace el descriptor SIFT.

Finalmente para calcular la localización de todos los puntos de interés en todas las escalas, se procede mediante la eliminación de los puntos que no cumplan la condición de máximo en un vecindario de 3 x 3 x 3. De esta manera, el máximo determinante de la matriz Hessiana es interpolado en la escala y posición de la imagen. En este punto se da por concluida la etapa de detección de los puntos de interés.

2. Asignación de la orientación

   Para conseguir invarianza en la rotación, al igual que hace SIFT, es necesario asignar una orientación a cada punto de interés obtenido en el paso anterior. Para ello, se debe calcular la respuesta de las funciones Haar-wavelet en ambas direcciones, x e y, mediante las máscaras de Haar.

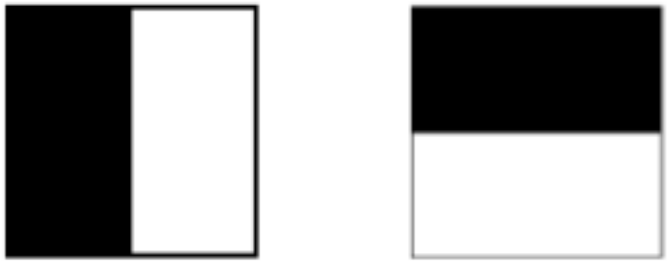


Figura B.2: funciones Haar-wavelet en ambas direcciones

Para cada punto, se calcularán estas respuesta en un área circular de $6s$, siendo s la escala en la que el punto de interés. Como se ha comentado, se hace uso de imágenes integrales proceder al filtrado mediante las máscaras de Haar y obtiene así las respuestas de manera eficiente.

Tras hacer el cálculo de las respuestas onduladas, se pondera cada respuesta por una Gaussiana de valor $\sigma = 2{,}5s$ centrada en el punto de interés. Las respuestas son representadas como vectores en el espacio colocando la respuesta horizontal y vertical en el eje de abscisas y ordenadas respectivamente. Finalmente, se obtiene una orientación dominante por cada sector mediante la suma de todas las respuestas dentro de una ventana de orientación móvil cubriendo un ángulo de $\pi/3$.

3. Creación del descriptor

   Primero se construye una región cuadrada de tamaño $20s$ alrededor del punto de interés orientada en relación a la orientación calculada en la etapa anterior. Esta región se divide en regiones cuadradas de 4x4, de las que se calculan las respuestas de Haar de puntos equiespaciados. Se consideran dx y dy las respuestas de Haar en las direcciones horizontal y vertical respectivamente relativas a la orientación del punto de interés. A continuación, se obtiene la suma de éstas sobre cada subregión y su valores absolutos, proporcionando así un vector descriptor de 4 dimensiones

$$V = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|) \tag{B.1}$$

   Los valores absolutos dan información sobre la polaridad de los cambios de intensidad.

El resultado es un vector de longitud 64 para cada subcelda de 4x4, en comparación con el descriptor SIFT cuya longitud era 128; por lo tanto, al tener menor número de elementos, también se conseguirá aceleración del tiempo computacional en el matching.

Las respuestas de la wavelet son invariantes a los desplazamientos en la iluminación (offset) y la invarianza al contraste se obtiene convirtiendo el descriptor en un vector unitario.

Apéndice C

Histograms of Oriented Gradients. HOG

En 2005, Navneet Dalal y Bill Triggs, presentaron el descriptor Histograms of Oriented Gradients (HOG). El descriptor se basa en la idea que la apariencia local de un objeto, así como su forma, dentro de una imagen, pueden ser descritas por la distribución de la intensidad de sus gradientes o la dirección de sus bordes. Aunque si que parte de la idea de los descriptores anteriormente tratados, este algoritmo trabaja sobre una red de celdas uniformemente espaciadas y usa normalizaciones de superposición local de contraste para mejorar el rendimiento. Los pasos que sigue el método para ello son los siguientes:

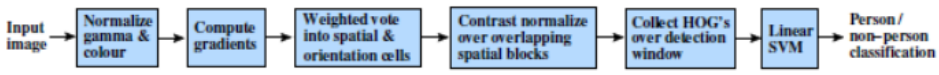


Figura C.1: Esquema algoritmo HOG

1. Obtención gradientes

   En otros detectores de características, lo primero que se debe hacer es una normalización de color y luminosidad para tener un conocimiento de los gradientes correspondientes o las posiciones de los contornos. Sin embargo, para el detector HOG no es necesario, ya que se obtendrá resultados similares, mediante la división de la imagen en pequeñas regiones llamadas *celdas*.

   El primer paso necesario es la obtención de los valores de gradientes. Para ello, Dalal y Triggs optan por el método 1-D centrado. Es decir, utilizan el operador derivador simple unidimesional ([-1 0 1]) para el cálculo del gradiente.

2. Orientación de las celdas

   Como ya hemos dicho, la imagen se divide en regiones más pequeñas, uniformemente distribuidas llamadas celdas. Dentro de cada celda, cada pixel procesa un voto ponderado para el histograma de orientación de borde basado en la orientación del gradiente de cada elemento, cada voto se acumula en los contenedores (bins) de la región espacial local donde fue calculada.

   Las celdas pueden ser rectangulares o radiales y los gradientes pueden normalizarse entre 0° y 360° o entre 0° y 180° en caso que se considere el gradiente con o sin signo respectivamente. Los contenedores pueden cubrir un amplio rango de orientaciones, sin embargo, el mismo valor puede no ser efectivo para el reconocimiento de todos los objetos, por ejemplo, Dalal y Triggs determinaron que para el caso de detección de personas, 9 *bins*.

3. Descriptor de bloques

   Para hacer frente a los cambios de iluminación y contraste, las fuerzas de gradiente deben normalizarse a nivel local, lo que requiere la agrupación de las celdas en conjuntos más grandes llamados bloques, que se encuentran conectados espacialmente. El descriptor HOG es un vector concatenado de los componentes de los histogramas de celdas normalizados para toda la región del bloque. Estas regiones más grandes están superpuestos, de manera que cada celda contribuye más de una vez en el descriptor final.

   Según Dalal y Triggs, la mejor opción es 4 celdas de 8x8 píxeles por bloque, es decir, cada bloque es de 16x16 pixeles, con histogramas de 9 posiciones.

   Además, para obtener una cierta mejora sobre el rendimiento, se hace uso de una ventana espacial Gaussiana dentro de cada bloque antes de hacer la tabulación de votos en el histograma con el fin de ponderar los píxeles alrededor los bloques. Los bloques pueden tener forma rectangular (R-HOG) o forma circular (C-HOG). En cualquiera de los dos existe superposición entre bloques de manera que una celda aporta información en más de un bloque.

4. Normalización de bloques

Para una tener una invarianza más robusta ante iluminación o contrastes se hace uso de la normalización de contraste local. Se presentan cuatro posibles normalizaciones:

L2-norm

$$f = \frac{v}{\sqrt{\|v\|^2 + e^2}} \tag{C.1}$$

L2-Hys: consiste en L2-norm limitando los valores máximos de v a 0.2 y depués normalizando de nuevo.

L1-norm

$$f = \frac{v}{(\|v\|_1 + e)} \tag{C.2}$$

L1-sqrt

$$f = \sqrt{\frac{v}{(\|v\|_1 + e)}} \tag{C.3}$$

Según Dalal y Triggs L2-norm, L2-Hys y L1-sqrt tienen un rendimiento similar, mientras que L1-norm proporciona un rendimiento ligeramente menos fiable, aproximadamente un 27 %. A pesar de ello, las cuatro normalizaciones muestran una mejor muy significativa.

5. Clasificador SVM lineal

   Finalmente, se introduce el descriptor en un sistema de reconocimiento basado en el aprendizaje supervisado.

Apéndice D

Random sample cosensus. RANSAC

En este PFC hemos mencionado el uso del algoritmo RANSAC para eliminar emparejamientos erróneos entre puntos característicos de dos imágenes que han podido verse afectados por transformaciones paramétricas como el escalado, rotación y afinidad. Bien, en este anexo se explica el funcionamiento de dicho algoritmo.

RANSAC es un algoritmo introducido por Martin L. Fischer y Rober C. Bolles en 1981, [2] de mecánica recursiva, se utiliza para estimar los parámetros de un modelo matemático de un conjunto de datos en los que pueden contenerse outliers (valores atípicos). Este algoritmo se utilizó para estimar una homografía a partir de puntos clave obtenidos de las características de una imagen. Los algoritmos que determinan los puntos clave cometen dos tipos de errores, de clasificación y de medida. Los primeros ocurren cuando el algoritmo de detección identifica incorrectamente una zona de la imagen como lugar donde se da una característica. Los errores de medida se dan cuando el algoritmo detecta la característica pero calcula erróneamente su ubicación. Mientras que los errores de medida tienen una distribución normal y se cancelan al tomar muchos puntos, los de clasificación no siguen ninguna distribución, y dan lugar a resultados indeseados en la obtención de la homografía.

Básicamente el algoritmo sigue 4 pasos que se describen a continuación:

1. Dado un modelo que requiere un mínimo de n para determinar sus parámetros, y un conjunto de datos P tal que el número de puntos en P es mayor que n, se sortea un subconjunto S1 de n puntos de P para instanciar el modelo. Con este modelo instanciado M1 se determina el subconjunto S1' de puntos de P que están a menos de una distancia t de M1.

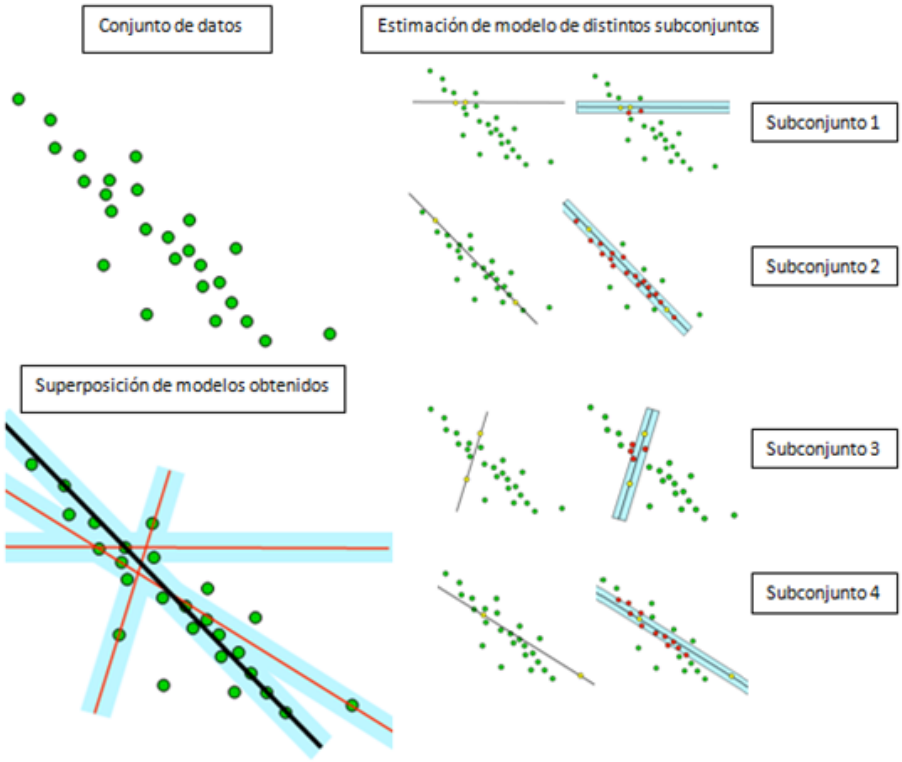


Figura D.1: Representación del proceso RANSAC en imágenes, dónde los puntos amarillos son los puntos utilizados para generar el modelo lineal y los puntos rojos son los puntos que encajan bien con el modelo obtenido. En el caso representado en la imagen, RANSAC nos devolvería como modelo el subconjunto 2.

2. Si la cantidad de puntos que contiene S1' es mayor que un determinado umbral T entonces dicho subconjunto de decisión S1 es elegido para computar el nuevo modelo M1'.

3. Si por el contrario la cantidad de puntos en S1' es menor que el umbral establecido T, se sortea un nuevo subconjunto S2 y se repite el proceso. Si tras la ejecución de un numero establecido de iteraciones N, no se obtiene un subconjunto que supere el umbral T, se resuelve el modelo con el subconjunto de decisión más grande obtenido, o se termina sin devolver modelo.

En primer lugar hay que determinar el umbral el cual determina si un punto es considerado inlier u outlier. Se debe elegir un umbral t talque la probabilidad de que un punto es un inlier es α. Para ello es necesario conocer la distribución de la probabilidad de la distancia de un inlier del modelo. Se

asume que el error es gaussiano de media cero y varianza σ. Para un α de 0,95 el umbral utilizado para calcular homografías es $t = \sqrt{5,99}\sigma$.

Otro parámetro a determinar es el número de iteraciones N. Este valor se calcula en función de la probabilidad p de que por lo menos un subconjunto que se elige aleatoriamente no tenga outliers.

$$1 - (1 - (1 - e)^S)^N = p \tag{D.1}$$

Siendo:

- **e**: probabilidad de que un punto sea un outlier.
- **s**: número de puntos en una muestra.
- **N**: número de muestras.
- **p**: probabilidad deseada de conseguir una muestra buena.

Además si analizamos esta ecuación detenidamente:

- **1-e**: probabilidad de elegir como punto un inlier.
- $(1 - e)^S$: Probabilidad de elegir **s** inliers en una fila.
- $1 - (1 - e)^S$: Probabilidad de que uno o más puntos de la muestra fueran outliers.
- $1 - (1 - e)^{S^N}$: Probabilidad de que N muestras fueran contaminadas.
- $1 - (1 - (1 - e)^S)^N$: Probabilidad de que al menos una muestra no fue contaminada.

Por lo tanto se elige N para que, con probabilidad p, al menos una muestra aleatoria esté libre de outliers.

$$1 - (1 - (1 - e)^S)^N = 1 - p \tag{D.2}$$

$$N = \frac{\log(1 - p)}{\log(1 - (1 - e)^S)} \tag{D.3}$$

Una vez tenemos varios subconjuntos, hay que determinar el de consenso. Como hemos visto, sólo necesitamos N subconjuntos elegidos aleatoriamente. Sin embargo, puede que antes de obtener los N subconjuntos obtengamos la solución. Para ello nos fijaremos si el ratio de inliers alcanza al ratio esperado de inliers. Por lo tanto se toma T=(1-e)n donde n es la cantidad de puntos del conjunto de datos.

Algoritmo RANSAC: líneas de ajuste utilizando random sample consensus

Utiliza los siguientes parámetros:

s: Menor número de muestras requerido.

N: Número requerido de iteraciones.

t: umbral que determina si un punto en inlier u outlier.

T: Número requerido de puntos cercanos para determinar si el modelo encaja bien.

Hasta que se llegue a N iteraciones

    Dibujar muestra aleatoria de s puntos del conjunto de datos

    Calcular modelo lineal a los s puntos

    Para cada punto fuera del modelo

        Compara distancia al modelo con d

            Si la distancia es menor que d el punto pertenece al subconjunto

        fin

    Si hay T o más puntos pertenecen al subconjunto se considera correcto.

        Recalcular modelo utilizando los puntos del subconjunto

    fin

Apéndice E

Emparejamiento de descriptores

Una vez hemos obtenido los descriptores de las imágenes a comparar, será necesario hacer un emparejamiento entre puntos característicos de ambas fotografías, para determinar el grado de similitud entre ellas. Tanto el algoritmo SIFT como el SURF hacen una correspondencia inicial de características, buscando aquél descriptor cuya distancia sea menor en la otra imagen. Pero este emparejamiento no es perfecto, se obtienen varios espurios, puntos con mala correspondencia, que distorsionan los resultados; para mejorar el sistema y filtrar aquellos puntos mal emparejados, se aplica el algoritmo RANSAC [Anexo D].

Este algoritmo trata de ajustar un conjunto de datos de entrada a un modelo que puede ser propuesto o estimado a partir de los propios datos. RANSAC utiliza el conjunto de datos más pequeño posible y procede a extenderlo mientras esta agrupación siga siendo consistente con los puntos dados. Es decir, a partir del emparejamiento inicial entre descriptores, buscará un conjunto mínimo de puntos que cumplan un modelo, y tratará de extenderlo hasta alcanzar el máximo número de puntos que siguen ese modelo establecido.

Vamos a comprobar gráficamente su funcionamiento. Primero, calculamos los descriptores SIFT, o SURF, de dos imágenes y hacemos la correspondencia inicial.

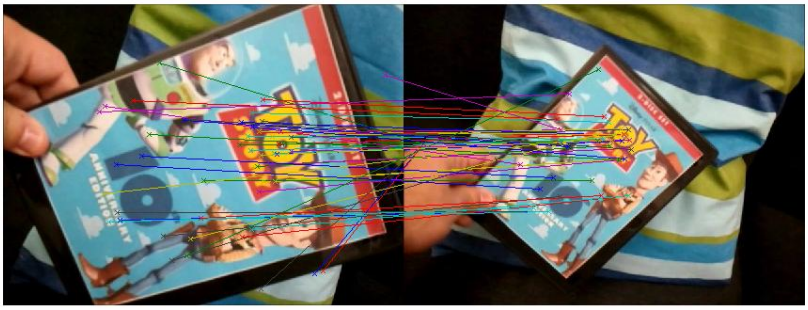


Figura E.1: Descriptores con emparejamiento inicial tras SURF.

Como podemos comprobar en la figura (número figura), hay muchos descriptores con emparejamiento correcto entre ambas imágenes, pero también hay líneas que no siguen el modelo y se cruzan con el resto, estos serán los espurios que habrá que eliminar para asegurar el correcto funcionamiento del sistema.

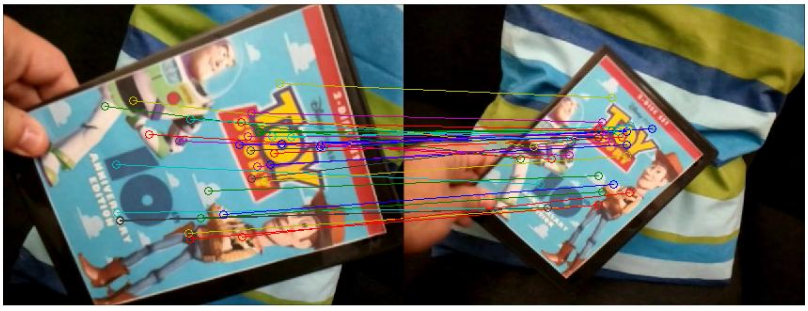


Figura E.2: Descriptores con correspondencia tras el filtrado por RANSAC.

Ahora se han eliminado los espurios y solo tenemos los puntos con una correspondencia óptima entre ambas imágenes.

Bibliografía

[1] Artner1, N. M., and Kropatsch1, W. G. Structural cues in 2d tracking: Edge lengths vs. barycentric coordinates. *Vienna University of Technology Pattern Recognition and Image Processing Group, Vienna, Austria* (2013).

[2] Derpains, K. G. Overview of the ransac algorithm.

[3] Herbert Bay, T. T., and Gool, L. V. Surf: Speeded up robust features. *ETH Zurich, Switzerland, and Katholieke Universiteit Leuven, Belgium* (2006).

[4] Lowe, D. G. Distinctive image features from scale-invariant keypoints. *University of British Columbia, Vancouver, B.C, Canada* (2004).